Fast Rate-Distortion Optimal Packetization of Embedded Bitstreams into Independent Source Packets

FAST RATE-DISTORTION OPTIMAL PACKETIZATION OF EMBEDDED BITSTREAMS INTO INDEPENDENT SOURCE PACKETS

BY

JIAYI XU, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2011) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Fast Rate-Distortion Optimal Packetization of Embedded Bitstreams into Independent Source Packets
AUTHOR:	Jiayi Xu B.Sc., (Biomedical Engineering) Fudan University, Shanghai, China
SUPERVISOR:	Dr. Sorina Dumitrescu
NUMBER OF PAGES:	xi, 66

To my family

Abstract

This thesis addresses the rate-distortion optimal packetization (RDOP) of embedded bitstreams into independent source packets for the purpose of limiting error propagation in transmission over packet noisy channels. The embedded stream is assumed to be an interleaving of K independently decodable basic streams. The goal is to partition these basic streams into $N(N < K)$ groups, which later are used to form N independent source packets. The streams within each group are interleaved according to their relative order in the original embedded stream to generate the source packet. Error protection can be furthered applied across source packets to produce the channel packets to be transmitted.

The RDOP problem previously formulated by Wu *el al.* focused on finding the partition that minimizes the distortion when all packets are decoded. The authors proposed a dynamic programming algorithm which worked under both high bit rate and low bit rate scenarios. In this thesis, we extend the problem formulation to finding the partition which minimizes the expected distortion at the receiver for a wide range of transmission scenarios including unequal/equal error/erasure protection and multiple description codes. Then we show that the dynamic programming algorithm of (Wu *et al.*, 2001) can be extended to solve the new RDOP problem.

Furthermore, we propose a faster algorithm to find the globally optimal solution based on the divide-and-conquer technique, under the assumption that all *basic* streams have convex rate-distortion curves. The proposed algorithm reduces the running time from $O(K^2LN)$ achieved by the dynamic programming solution to $O(NKL \log K)$. Experiments performed on SPIHT coded images further validate that the speed up is significant in practice.

Acknowledgements

"Don't let the noise of other's opinions drown out your own inner voice " was an insightful quote from Apple co-founder Steve Jobs that inspired me to continue my study at McMaster University as a Master's student.

Everyone has a story and my story is if I didn't decide to become a graduate student, I would've been an accountant or a salesman. Not that I was good at calculating numbers or selling toothbrushes, it's because they were popular among new graduates in town. Although life would be different if I chose either of them, I knew I would be thinking the same thing 30 years later, "What was I thinking ?!". I've always known that technology/science/engineering is my Fountain of Youth. And for that, I need to thank Steve Jobs, or the guy who put his video on Youtube.

For the past two years, Dr. Sorina Dumitrescu, my supervisor, has set up an exceptional example of how important devotion and hard work is in professional work. I wouldn't be half of what I am now if it wasn't her to help me start a new chapter in the research work. I want to thank for her patience and valuable input during my journey here and wish her continue success on her work.

I am also very grateful to Dr. Shahram Shirani and Dr. Alexandru Patriciu for being part of my thesis committee and providing me with valuable input. I would like to thank my friends, Li, Yinhan, Dong, without whom the taste of food would

be a mission impossible. Yang and Ge are guys who make my life so easy in a foreign country. Special thanks to Jason, Xiao and Daguanren, whose ingenious opinions are always inspiring and philosophical.

Last but not least, I want to thank Richard Feynman, Donald Knuth, Claude Shannon and all other greatest minds of our time. I have not gotten the privilege to meet them in person, but it is their stories and words that motivate me to continue my career in technology and never look back.

List of Abbreviations

D&C: the proposed Divide-and-Conquer algorithm for RDOP

DP: the Dynamic Programming algorithm for RDOP

EBCOT: Embedded Block Coding with Optimal Truncation

EEP: Equal Error/Erasure Protection

EZW: Embedded zerotree wavelet

FMUEP: MUEP with Fixed redundancy

IP: Internet Protocol

MUEP: Unequal Error/Erasure Protection for progressive Multi-streams

OPE: Optimal Partitioning with Equal Coefficients

OPUF: Optimal Partitioning with Unequal coefficients and Fixed across packets

OPUV: Optimal Partitioning with Unequal coefficients and Variable across packets

PZW: Partitioning algorithm proposed in (Rogers and Cosman, 1998)

RCPC: Rate Compatible Punctured Convolutional codes

RDOP: Rate Distortion Optimal Partitioning

SMAWK: Fast matrix search algorithm introduced in (Aggarwal *et al.*, 1987)

SPIHT: Set Partitioning in Hierarchical Trees

UEP: Unequal Error/Erasure Protection

Contents

List of Figures

Chapter 1

Introduction

1.1 Robust Data Transmission over Noisy Networks

Packet-switching networks used in multimedia communication systems such as the Internet and ATM have to deal with the problem of packet loss and/or packet corruption during transmission. Internet Protocol (IP) networks are such packet-switching networks used for relaying datagrams (packets) over the Internet. For the benefit of reducing network complexity, the IP only provides best-effort delivery, i.e., the network guarantees neither that data is received nor that a user is given a certain quality of service level or priority. In consequence, its service can be characterized as unreliable and data corruption and packet loss might occur during the transmission. Both fault events can be caused due to 1) network congestion, when a router is over-filled with datagrams, in which case it must discard some of the incoming packets, 2) bit errors, when the packets arrived contain uncorrectable bit errors, or 3) transmission delay, when the delay from the transmitter to the receiver unfortunately fails the real-time constraints of the application, making the packets useless to the receiver.

As far as the robust data transmission is concerned, techniques of error control such as automatic repeat request (ARQ) and forward error correction (FEC) have been developed in order to tackle such problems. In ARQ, the receiver is asked to send an acknowledgement message to the transmitter to indicate a successful transmission. If the transmitter fails to receive the acknowledgement message before timeout, it will re-transmit the packet or frame until the acknowledgement message is received or the maximum number of re-transmission is exceeded. Under circumstances where re-transmission is costly, FEC is used instead, in which case the transmitter includes redundant information along with the data. The receiver is able to detect and correct errors by exploiting this redundant information without asking the sender to re-transmit the data. The ability to detect and correct error varies with the specific code designed. However, both systems have deficiencies in the sense that requesting re-transmission causes delay which is not acceptable in many real-time applications while using FEC will consume the valuable bit budget in order to provide redundant information for the decoder to correct errors.

1.2 Embedded Codestreams

Progressive image or video coders, such as SPIHT (Said and Pearlman, 1996), EBCOT (Taubman, 2000) and 3DSPIHT (Kim *et al.*, 2000), have become increasingly popular due to their ability to easily adapt to rate variations. Such encoders generate embedded or successively refinable bitstreams, i.e., bitstreams in which any prefix can reconstruct the signal to a certain fidelity. In other words, these algorithms provide progressive content to represent images and videos. The compressed data is allowed to be chunked at any point with reconstruction quality proportional to the length of

the decoded chunk.

Image coders that produce embedded bitstreams have been broadly studied over the years. Since natural images are mixed with low-frequency components and high-frequency edges, wavelet transforms have become an ideal and efficient tool for image compression. Shapiro's (Shapiro, 1993) embedded zero-tree wavelet algorithm was a pioneering work in the design of wavelet-based encoders for generating embedded bitstreams. The SPIHT algorithm (Said and Pearlman, 1996) developed by Said and Pearlman greatly improved the zero-tree based EZW albeit the slight difference in the tree structure (see Fig 1.1).

The state-of-art JPEG2000 compression standard (Taubman and Marcellin, 2001) uses Taubman's embedded block coding with optimized truncation (EBCOT) algorithm (Taubman, 2000). Unlike the EZW and SPIHT, JPEG2000 is block-based encoded in the sense that nonoverlapping blocks are composed of neighboring wavelet coefficients, which are later encoded without zero-tree coding. The context-adaptive approach selected in the JPEG2000 allows the encoder to generate scalable bitstreams. Moreover, the use of block-based coding strategy enables JPEG2000 to be implemented under low-memory environment and also has the benefit of confining error propagation inside individual blocks.

While embedded bitstreams are attractive due to their rate scalability property, they are also prone to error propagation due to their successively refinable nature. This is because in order to decode a particular bit, all previous bits have to be available at the decoder, i.e., correctly received. This fact marks the significant importance of the initial bits. In fact, the reconstruction quality does not depend on the total bits received at the decoder, but rather on the length of the bits correctly decoded up

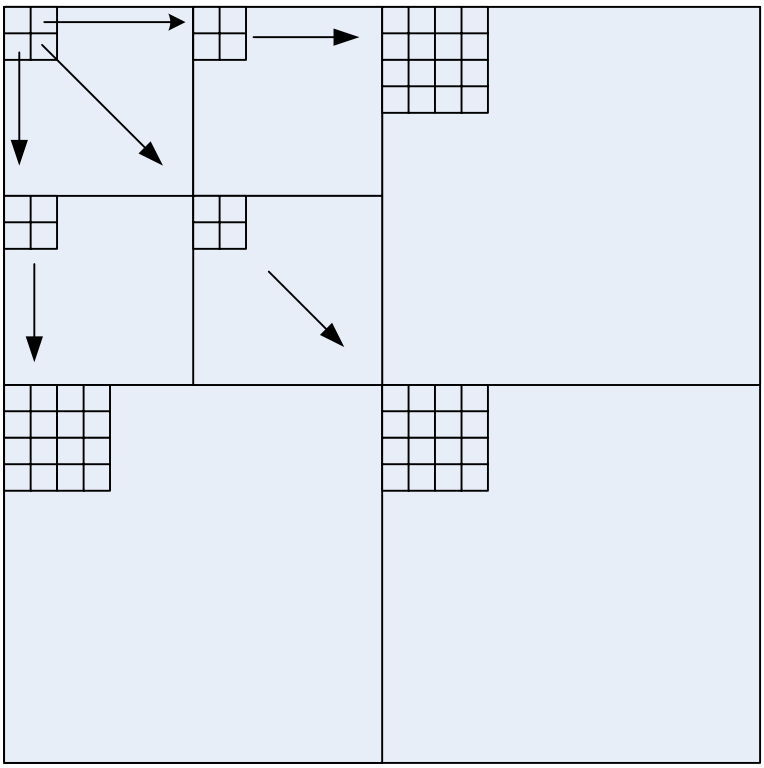


Figure 1.1: Parent-child relationships in SPIHT

to the first uncorrectable error. As a result, if the bitstream is simply sequentially chunked to form the packets, loss of a single packet will degrade image quality if the packet contains the early bits from the embedded bitstream.

1.3 Unequal Error Protection (UEP) for Embedded Bitstreams

Since embedded bitstreams are highly susceptible to errors, various FEC-based strategies have been developed to alleviate the impact of packet loss or bit errors. Specifically, Unequal Error Protection techniques (UEP), that give more protection towards more important bits (prefix in the embedded bitstream) and less protection or even no protection to less important bits, are very popular.

Many UEP frameworks have been proposed over the years. In (Chande and Farvardin, 2000), the authors provide channel protection with a finite family of block codes, where each member of the family has error correction and error detection capability. Their strategy employs a concatenation of rate compatible punctured convolutional (RCPC) codes for error correction and cyclic redundancy check (CRC) codes for error detection. The strength of error correction is decreasing to achieve unequal protection among data blocks. The layout of this type of strategy contains the same number of source symbols but the number of channel symbols varies. Another type of strategy was proposed in (Stankovic *et al.*, 2003) where the channel code block length is fixed and the number of source symbols varies in each code block. Both strategies can be optimized in rate-distortion sense by minimizing the expected distortion at the receiver end given fixed rate budget.

Another UEP strategy to combat packet loss uses Reed Solomon (R-S) codes of decreasing strengths to protect subsequent segments of the code stream(Mohr *et al.*, 1999; Puri and Ramchandran, 1999). Unlike those in (Chande and Farvardin, 2000) and (Stankovic *et al.*, 2003), where each channel codeword forms a packet, in this

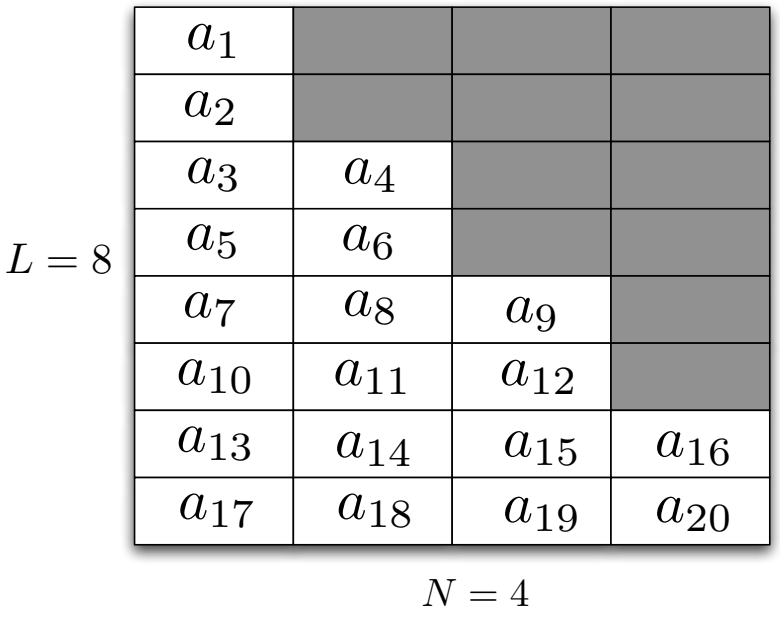


Figure 1.2: An example of UEP packetization array with $N = 4$ and $L = 8$, where L denotes the number of symbols in a packet. White boxes represent source symbols and grey boxes represent redundant symbols.

technique, the packets are formed across the channel codewords. The source symbols protected by an (N, i) R-S code, where N is the number of packets, are said to be in layer i. Such a code ensures that all channel symbols can be recovered from at most $N - i$ erasures. Thus, if only $k(k < N)$ packets are received at destination, all source symbols in the first k layers are recovered using the R-S erasure protection. They form a prefix of the embedded bitstream, which will be entirely decoded. Fig 1.2 presents an example of such UEP packetization array for transmitting an embedded bitstream $M = \{a_1, a_2, a_3, \cdots\}$. Rate-distortion optimization for this UEP framework has been studied extensively in (Mohr *et al.*, 1999; Puri and Ramchandran, 1999; Mohr *et al.*, 2000; Stockhammer and Buchner, 2001; Stankovic *et al.*, 2004; Dumitrescu *et al.*, 2004; Thie and Taubman, 2005; Hamzaoui *et al.*, 2005; Dumitrescu *et al.*, 2007).

1.4 Independent Source Packets

The UEP technologies mentioned in the previous section attempt to combat error, but do not address the issue of error propagation. A natural way to limit error propagation is by confining the error only within the packet it occurs by breaking the decoding dependence between packets. This can be achieved by partitioning the data samples into groups, then encoding each group independently and transmitting the resulting progressive stream within a packet. Many researchers have taken this approach to generate independent source packets. Man *el al.* (H. Man and Smith, 1997) proposed an improved SPIHT algorithm to categorize the output bit sequence into subsequences, which later receive different amounts of rate compatible punctured convolutional codes (RCPC) protection based on their significance and robustness. Creusere (Creusere, 1996) partitioned the wavelet transform coefficients into groups and independently processing each group using an embedded coder. Applying similar idea to videos, the video codec proposed in (Crump and Fischer, 1997) combined entropy coded scalar quantization and subband coefficient trees with run-length and Huffman coding to isolate spatially related trees and thus encapsulating the encoded bitstream in independent variable-length packets. In (Tillo *et al.*, 2007, 2010; Subbalakshmi and Somasundaram, 2002; Jiang and Ortega, 1999; Akyol *et al.*, 2007), MDC frameworks were proposed where packets (i.e., descriptions) are composed of subsets of samples from the source bitstream encoded at the different rates. Thus, any packets that are able to be received and correctly decoded can reconstruct subsets at the corresponding rate.

On the other hand, progressive coders already employ partitioning of the set of samples into basic subsets, which are independently encoded producing embedded

sub-streams. These sub-streams are further interleaved in nearly optimal rate distortion sense to generate a single coded stream for the entire data. Specifically, in SPIHT, each basic subset is a spatial orientation tree of coefficients, while in JPEG2000 each basic subset is a block of coefficients of fixed dimension, within a subband. We will use the term *basic* or *primary* streams for these embedded bitstreams encoding each basic subset of samples. This feature can be exploited in the construction of independent source packets. In particular, in the case when the number K of basic streams is higher than the required number N of packets, the primary streams can be grouped into N pairwise disjoint groups. Next the streams within each group are interleaved to generate a separate source packet. Adopting this procedure reduces the task of partitioning the whole set of samples to the simpler task of partitioning the smaller set of basic subsets, or in other words, partitioning the set of basic streams. Many researchers have taken this approach to generate independent source packets, however most have used ad-hoc methods to choose the partition. For instance, Rogers and Cosman (Rogers and Cosman, 1998) developed a scheme for robust transmission of SPIHT coded images over packet lossy channels, where no channel coding is applied to the independent source packets, but error concealment is used at the decoder to recover from losses. The basic streams (i.e., the code streams corresponding to individual trees) are assumed to be in a fixed order known to both the encoder and decoder. The groups are formed with consecutive basic streams until approximately reaching the capacity of the fixed size packet. To fit exactly the size of the packet, the interleaved stream is either truncated or extended, as necessary.

The first work which clearly addresses the problem of optimal packetization of embedded streams, into independent source packets, is the work of Wu *et al.* (Wu

et al., 2001). The construction of source packets is also based on partitioning the set of basic streams into groups, but the packetization constraints are more relaxed than in (Rogers and Cosman, 1998) by also allowing a group which contains only one basic stream, to occupy more than one packet. On the other hand, the condition that each group is formed only from consecutive packets in the fixed ordering, is maintained in order to reduce the amount of side information needed to specify the composition of each group. The goal of the optimization problem is, given a fixed budget of N packets, each containing L symbols, to minimize the distortion when all the packets are decoded. The authors of (Wu *et al.*, 2001) propose a globally optimal dynamic programming solution. However, as their simulations show, this solution is impractical due to the high overload of precomputations.

1.5 Contribution and Organization of the Thesis

This thesis reexamines the problem of rate distortion (R-D) optimal packetization of embedded streams into independent source packets based on partitioning of the set of basic streams. In our formulation we impose the constraint of (Rogers and Cosman, 1998) that each group is formed out of consecutive primary streams according to a fixed ordering, and that the interleaved stream of each group occupies a single packet. We consider a wide range of transmission scenarios where the source packets may be further protected against bit errors and/or erasures with equal or unequal error protection, but without error concealment at the decoder. For such a general transmission scheme, with fixed bit budget and fixed redundancy allocation, we formulate the optimization problem as the problem of finding the partition which minimizes the expected distortion at the receiver. We refer to this problem as the

problem of rate-distortion optimal packetization or rate-distortion optimal partitioning (RDOP). Mathematically, this problem is equivalent to maximizing a weighted sum of decrements in distortion corresponding to individual transmitted source symbols. The weight assigned to a symbol represents the probability that the symbol is recovered and decoded and depends on the channel protection (if any) assigned to that symbol. Moreover, we distinguish between several cases of the RDOP problem, referred to as OPUV, OPUF and OPE. This categorization is relevant from the point of view of the solution algorithm complexity. In OPUV the weights are unequal within each packet, and they may vary from packet to packet, while in OPUF, they are fixed from packet to packet. We show that OPUV and OPUF are relevant for schemes employing unequal erasure protection across packets, which are very popular for robust transmission of embedded data streams. Finally, in OPE, all the weights are equal. It is interesting to note that OPE is equivalent to minimizing the distortion when all the packets are received, in other words, it has the same objective as the optimal packetization problem of (Wu *et al.*, 2001). Moreover, solving OPE also minimizes the expected distortion for schemes with equal error protection and channels with independent losses and/or bit errors.

Due to similarities in the problem formulation, the dynamic programming algorithm of (Wu *et al.*, 2001) can be extended to solve the new RDOP problem. Our main contribution is a more efficient globally optimal solution to the problem. To make possible the accelerated solution, we assume that the rate-distortion curve of each basic stream is a convex function. This assumption has been widely used in previous work and is a good approximation for practical situations. Based on this assumption we show that the problem can be reduced to a series of matrix search

problems in totally monotone matrices[1]. This nice property enables a reduction of the time complexity from $O(K^2LN)$ (via dynamic programming) to $O(NKL\log K)$, based on the divide and conquer strategy. Furthermore, having a simpler form than OPUV, both OPUF and OPE can be solved faster in practice than predicted by the asymptotical complexity evaluation of $O(NKL\log K)$ running time.

The thesis is organized as follows. In Chapter 2 the RDOP problem is formulated for a general transmission scenario. Chapter 3 presents explicit examples of transmission schemes where the above formulation is motivated, including schemes with equal/unequal error protection, product codes and multiple description schemes. The solution based on dynamic programming and the fast divided and conquer algorithm based on total monotonicity are revealed in Chapter 4. Experimental results are presented in Chapter 5 and Chapter 6 concludes the thesis.

[1]This notion was introduced in (Aggarwal *et al.*, 1987) and will be defined precisely in Chapter 5.

Chapter 2

Problem Formulation

In this Chapter we present the mathematical formulation of the problem of rate distortion optimal packetization (RDOP) or partitioning into source packets. Section 2.1 introduces the notations and the problem setting. The cost function we consider is further explained in section 2.2. Finally, the definition of partitioning and final problem formulation are presented in section 2.3.

2.1 Problem Setting

Let us first explain the setting of the problem. Assume that the set of samples is divided into K pairwise disjoint groups. Each group is encoded using a successively refinable (i.e., progressive or embedded) coder. Thus, K independently decodable, embedded basic streams, labeled $B_1, B_2, \cdots, B_K$, are generated. Each B_k is a sequence of l_k symbols (a symbol consists of a fixed number of bits - 8 bits in our tests). Each basic stream B_k is associated a sequence $\Delta_k = (\Delta_k(1), \Delta_k(2), \cdots, \Delta_k(l_k))$ of non-increasing non-negative numbers. Specifically, $\Delta_k(i)$ represents the decrease in

distortion due to decoding the i-th symbol of B_k, given that all previous $i-1$ symbols have been decoded. The fact that Δ_k is non-increasing is equivalent to the assumption that the rate distortion curve associated to B_k is convex.

We consider a general transmission scenario where first N ($N < K$) independent source packets are produced $S_1, \cdots, S_N$. Then necessary headers are appended and error protection may be applied across and/or along source packets to generate the channel packets, which are further transmitted over the channel/network. Each source packet S_n contains L_n symbols. To form the source packets, the set of basic streams is first partitioned into N groups. The basic streams in the n-th group are interleaved and the prefix of size L_n of this interleaving constitutes the n-th source packet S_n. The interleaving is performed in optimal rate distortion manner, in other words such that the symbols are ordered in non-increasing order of their associated distortion reductions. Moreover, we impose the following constraint on the partitioning strategy: any group must consist of consecutive basic streams. This requirement is motivated by the desire to reduce the amount of side information needed at the decoder to specify the composition of each source packet. In particular, the decoder is informed of the labeling of basic streams. Therefore, in order for it to know what are the basic streams contained in each received packet, it is sufficient to specify the labels of the first and last basic streams in the group, using a header of $2\lceil \log_2 K \rceil$ bits.

2.2 Cost Function

The aim of the optimization problem will be to minimize the expected distortion at the receiver. In order to evaluate the expected distortion at the receiver, we need to describe first the operations at the decoder. They proceed as follows. From the

received data (all transmitted data or only part of it, corrupted or uncorrupted by errors), part or all of the transmitted source symbols are recovered based on the channel error protection employed by the scheme. Then for each source packet S_n all recovered source symbols forming a prefix of S_n are decoded. In other words, recovered symbols from S_n are decoded, in order, until the first missing symbol. Now let us denote by $\gamma(n, r)$ the probability that the r-th symbol of source packet S_n, is decoded (i. e., the probability that this symbol is recovered and all previous symbols in S_n are recovered as well). Also, let $\Delta D_n(r)$ denote the distortion decrease incurred by decoding the r-th symbol of source packet S_n. Then clearly, the expected distortion $\bar{D}$ at the receiver equals

$$\bar{D} = D_0 - \sum_{n=1}^{N} \sum_{r=1}^{L_n} \gamma(n, r) \Delta D_n(r), \tag{2.1}$$

where D_0 is the distortion when no information is available. Clearly, the value $\gamma(n, r)$ depends on the channel statistics, the level of channel error protection assigned to the r-th symbol of source packet S_n, and on specifics of the transmission scheme. Nevertheless, in all cases, these values obey a monotonicity property which will prove itself crucial (together with the assumption of convexity of the rate-distortion curves) to obtaining efficient solutions to our optimization problem. In particular, because the $(r + 1)$-th symbol cannot be decoded without decoding first the r-th symbol, the following relation holds

$$\gamma(n, r) \geq \gamma(n, r + 1), \text{ for all } n, r, 1 \leq n \leq N, 1 \leq r \leq L_n. \tag{2.2}$$

The objective of rate distortion optimization in our work is to minimize the expected distortion (2.1) at the receiver. Notice that in many transmission schemes covered by previous work, there is some flexibility in the redundancy allocation. Then the rate distortion optimization of the packetization would comprise joint optimization of the construction of source packets and of the redundancy allocation. This problem is very complex, and it is dependant on the particular scheme employed. In this work we consider the simplified problem case where the redundancy allocation is fixed, thus the coefficients $\gamma(n, r)$ are fixed, and we are concerned only with the construction of the source packets such that (2.1) to be minimized. This problem is meaningful in transmission scenarios where the redundancy allocation is decided independently of the partitioning into source packets, as is the case of FMUEP framework introduced in (Dumitrescu *et al.*, 2010), which will be discussed in the next section. For other scenarios where the redundancy allocation depends on the partitioning, the algorithm to optimize the partitioning can be incorporated into the joint optimization of the redundancy allocation and partitioning by iteratively optimizing one component while keeping the other one fixed.

2.3 Optimization Problem

In order to formally introduce the optimization problem we need to formulate mathematically the condition that each group in the partitioning is formed only out of consecutive basic streams. Notice that such a partition can be defined by an $(N+1)$-tuple $\mathbf{p} = (p_0, p_1, \ldots, p_N)$, with $0 = p_0 < p_1 < p_2 < \cdots < p_N = K$. The bitstreams in group n are $B_{p_{n-1}+1}, B_{p_{n-1}+2}, \ldots, B_{p_n}$. The sequence of symbols obtained by interleaving these bitstreams in optimal rate distortion manner, will be referred to as

the *composite stream* $B(p_{i-1}+1, p_i)$. Thus, source packet S_n consists of the first L_n symbols of $B(p_{i-1}+1, p_i)$. Furthermore, let us associate to each composite stream $B(j,k)$, $j<k$, and each packet index n, the weight $w_n(j,k)$ defined as

$$w_n(j,k) = \sum_{r=1}^{L_n} \gamma(n,r)\Delta_{j,k}(r), \tag{2.3}$$

where $\Delta_{j,k}(r)$ denotes the distortion decrease incurred by decoding the r-th symbol in $B(j,k)$. With these notations, the expected distortion corresponding to partition $\mathbf{p}$ becomes

$$\bar{D}(\mathbf{p}) = D_0 - \sum_{n=1}^{N} w_n(p_{n-1}+1, p_n), \tag{2.4}$$

We are ready now to formulate the optimization problem. Assume that the values $N, L_1, \cdots, L_N$ and coefficients $\gamma(n,r)$, for $1 \le n \le N$, $1 \le r \le L_n$, are given. The problem of rate distortion optimal partitioning (RDOP) of the set of basic streams is the problem of finding the partitioning $\mathbf{p}$ which minimizes the expected distortion $\bar{D}(\mathbf{p})$ defined in (2.4). Notice that this problem can be equivalently cast as maximizing the decrease in distortion denoted by $\Delta D(\mathbf{p})$, in other words finding

$$\max_{\mathbf{p}} \Delta D(\mathbf{p}) = \max_{\mathbf{p}} \sum_{n=1}^{N} w_n(p_{n-1}+1, p_n). \tag{2.5}$$

In the sequel we will distinguish between several cases of this problem formulation. As we will see this categorization is meaningful from the point of view of the complexity of the solution algorithm. The first case is when all source packets have the same length, i.e., $L_1 = L_2 = \cdots = L_N$, and all coefficients $\gamma(n,r)$, $1 \le n \le N$, $1 \le r \le L_1$, are equal. We will refer to this problem as OPE (OP with Equal coefficients). As

it will be revealed in the next section, this case is relevant to equal error protection scenarios. Moreover, due to the equivalent formulation (2.5), all coefficients in OPE can be set to 1. Thus, OPE is equivalent to minimizing the distortion when all source packets are decoded, which is the objective considered in (Wu *et al.*, 2001).

The second case is when $L_1 = L_2 = \cdots = L_N$ and $\gamma(1, r) = \gamma(2, r) = \cdots = \gamma(N, r)$ for each $1 \leq r \leq L_1$. This problem will be referred to as OPUF (OP with Unequal coefficients, but Fixed from packet to packet). This case is encountered in unequal error protection, where the redundancy allocation is fixed across packets. Finally, the most general case is when the source packets lengths may be different and coefficients $\gamma(n, r)$ may be different within each packet and across packets. This case is referred to as OPUV (OP with Unequal coefficients, and Variable from packet to packet).

Chapter 3

Motivation and Applications for RDOP

In this chapter we present specific examples of transmission scenarios motivating the formulation of the RDOP problem. In section 3.1, equal error protection and retransmission schemes are identified as motivation of the OPE problem. Section 3.2 discusses in detail the MUEP framework proposed in (Dumitrescu *et al.*, 2010). RDOP associated with product code with MUEP and multiple description codes are discussed in sections 3.3 and 3.4, respectively.

3.1 Motivation for OPE

The main motivation for the OPE problem is the case of equal error/erasure protection (EEP), symmetric channels, and independent bit errors. The OPE problem corresponds to the case where in the formula of the expected distortion (2.1), all source symbols have the same weight and hence the same error protection is applied

to all source symbols. Interestingly, the OPE problem is meaningful also in scenarios where the packet loss is dealt with by using retransmission. In such a case, transmitting independently decodable source packets can reduce the decoding delay at the receiver. For instance, consider a naive packetization, where the embedded data stream is divided into consecutive segments and each segment forms a packet. If the user receives all packets except for the first one, it cannot start decoding of received packets until the first packet is retransmitted. On the other hand, if packets are independently decodable, then the receiver can start decoding any packet as soon as it arrives. The natural optimization objective is minimizing the distortion when all packets are received, in other words, solving the OPE problem.

3.2 Unequal Erasure Protection for Progressive Multi-streams (MUEP)

MUEP (Dumitrescu *et al.*, 2010) is a transmission framework for progressive multi-streams (i.e., for code streams which can be decomposed into independently decodable embedded streams), over packet lossy channels. The main features of this scheme are independent source packets and uneven erasure protection across packets. In the MUEP framework, first the source packets S_n, $1 \leq n \leq N$, are produced. Each source packet is an independently decodable embedded stream. In order to explain how the erasure protection is applied we will describe the MUEP packetization array (PA). Column n of the array represents the n-th channel packet. All channel packets have the same size L'. Channel packet n consists of source symbols from source packet S'_n (S'_n is S_n with the header appended) interleaved with redundancy

symbols coming from the channel codes applied across packets. Each row i of the array is a *permuted* systematic $(N, m(i))$ Reed-Solomon (R-S) codeword. The term permuted refers to the fact that the $m(i)$ systematic bits are not necessarily at the beginning or end of the codeword, as is assumed usually in the UEP scenarios, but are interleaved with the redundancy bits, according to some specified permutation. Such a code is able to recover all symbols from at most $N - m(i)$ erasures. The strength of the R-S codeword decreases as i increases, i.e., $m(i) \leq m(i+1)$. Following (Dumitrescu *et al.*, 2010) let us refer to the rows consisting of (N, m) R-S codewords as layer m. Further, let x_m denote the number of rows in layer m, and let $x_m^{(n)}$ be the number of source symbols from source packet S_n, which are situated in layer m, $1 \leq n, m \leq N$. Note that the rows belonging to any layer are consecutive, due to the constraint which imposes non-increasing strength of the RS codes in the row index. As in UEP, some layers m may be empty, hence with $x_m = 0$. As an example, Fig 3.1 represents an MUEP packetization array for four independent bitstreams, $A = \{a_1, a_2, a_3, \dots, \}$, $B = \{b_1, b_2, b_3, \dots\}$, $C = \{c_1, c_2, c_3, \dots\}$ and $D = \{d_1, d_2, d_3, \dots\}$. Note that the rate distortion optimal interleaving of the four streams is $M = \{a_1, a_2, b_1, c_1, b_2, d_1, c_2, d_2, a_3, c_3, b_3, c_4, d_3, d_4, a_4, a_5, c_5, d_5, b_4, c_5\}$. Similar to UEP, if the number of received packets is k, where $k < N$, all source symbols in layer 1 through layer k can be successfully decoded. On the other hand, unlike UEP, all the symbols in the received packets can also be decoded since within one packet, the source symbols are a prefix of the same bit stream.

The expected distortion at the receiver is

$$\bar{D}_M = D_0 - \sum_{m=1}^{N} \sum_{n=1}^{N} C_M(n,m) \left(\sum_{r=1+\sum_{\ell=1}^{m-1} x_\ell^{(n)}}^{\sum_{\ell=1}^{m} x_\ell^{(n)}} \Delta D_n(r) \right), \tag{3.1}$$

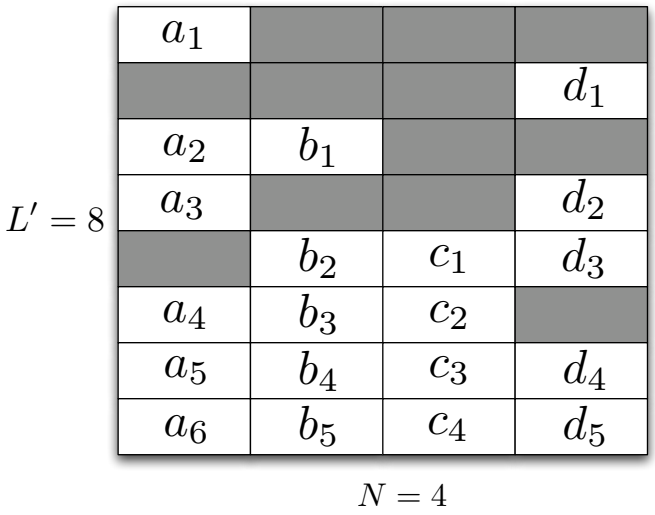


Figure 3.1: An example of MUEP packetization array for $N = 4$ and $L' = 8$. White boxes indicate source symbols and grey boxes indicate

where $C_M(n, m)$ denotes the probability that a source symbol in packet n situated in layer m is recovered and decoded at the receiver. Clearly, (3.1) has the form of (2.1), where $\gamma(n, r)$ equals $C_M(n, m)$ if the r-th symbol of S_n resides in layer m of PA. Due to non-increasing strength of the R-S code applied to consecutive source symbols in each packet, once a source symbol is recovered, it is guaranteed that all previous source symbols in the packet are recovered. Thus, $C_M(n, m)$ equals the probability that one of the following events occurs: 1) channel packet n is received at the destination, or 2) packet n is lost and at least m other packets are received. It is clear now that, given fixed budget and redundancy allocation, the problem of finding the partitioning which minimizes the expected distortion, reduces to solving the RDOP problem. Because the redundancy allocation for different symbols inside a source packet is uneven, the values $\gamma(n, r)$ can be different for different symbols r within each packet. Moreover, the erasure protection level assigned to the r-th symbol in different packets may be

different. Thus, the optimization problem, in the most general case, is an instance of OPUV. On the other hand, it becomes an instance of OPUF problem when the redundancy allocation is the same across packets and the channel is *symmetric*. The concept of symmetric channel is as considered in (Dumitrescu *et al.*, 2010), i.e., a channel where the probability that the packets in some subset $\mathcal{I} \subset \{1, 2, \cdots, N\}$ are lost, while the rest of packets are received, is the same for all subsets $\mathcal{I}$ of equal size. As shown in Dumitrescu *et al.* (2010), in such a case, the following relation holds

$$C_M(n, m) = 1 - \mu + \sum_{k=0}^{N-m} \frac{k}{N} P_N(k), \tag{3.2}$$

where μ denotes the mean packet loss rate, and $P_N(k)$ denotes the probability that out of the N transmitted packets, k are lost, while $N - k$ are received. Consequently, $C_M(n, m)$ does not depend on the particular packet n. Thus, when for each m, the number of source symbols in layer m, is the same for each packet, i.e., $x_m^{(1)} = x_m^{(2)} = \cdots = x_m^{(N)}$, we have $L_1 = L_2 = \cdots = L_N$ and $\gamma(1, r) = \gamma(2, r) = \cdots = \gamma(N, r)$ for each $1 \leq r \leq L_1$. In conclusion, optimizing the partitioning for this transmission scenario, reduces to solving the OPUF problem. The latter scenario, termed FMUEP, was considered in (Dumitrescu *et al.*, 2010), in order to decrease the amount of side information needed at the decoder. The redundancy optimization for FMUEP is performed in (Dumitrescu *et al.*, 2010) independent on the partitioning. Specifically, the optimal values x_m, $1 \leq m \leq N$, are computed only based on the rate distortion curve information of the coded stream for the whole data set (i.e., the interleaving of the basic streams). Then the number of source symbols in each layer m is set to be the same for all packets: $x_m^{(n)} = mx_m/N$. If the division mx_m/N is not exact, then some $x_m^{(n)}$'s are set to $\lfloor mx_m/N \rfloor$ and the others to $\lceil mx_m/N \rceil$, according to a fixed rule.

3.3 Product Code with MUEP

The MUEP transmission scheme can be extended to incorporate bit error protection within each packet in the case when the channel incurs packet losses and bit errors, or only bit errors. For this, after constructing the MUEP PA as described above, each column is applied an error correction code of some length $L" > L'$. This codeword will form the channel packet. Upon receiving a channel packet, the error correction code is used to correct bit errors. If this channel code fails to correct all errors[1], then the packet is declared unavailable. The packets which are corrected enter the MUEP PA decoder. Thus, in this case the expected distortion still has the form of (3.1), where $C_M(n, m)$ equals the probability that one of the following events occurs: 1) channel packet n is received at the destination and is recovered from bit errors, or 2) packet n is either lost or not recovered from bit errors, but at least m other packets are correctly recovered. Thus, optimizing the partitioning for this scenario reduces to solving the RDOP problem. Notice that the corrected packets which enter the PA decoder can be regarded as the output of a second packet lossy channel, which is the cascade of the initial transmission channel followed by the intra packet decoder. If the initial transmission channel is symmetric, bit errors are independent, from packet to packet and bit error statistics is the same for every packet, then this second channel is symmetric as well. Thus, optimizing the partitioning is a case of OPUF. Such a product code scheme was used in (Thomos *et al.*, 2006), with the only difference that

[1]It is assumed that the decoder of the intra packet channel code either corrects all errors or declares inability to correct all errors. This can be realized in practice via a concatenated code with an outer error detection and inner error correction codes.

the R-S codes used across packets are *strictly* systematic codes, i.e., the redundant symbols appear only after the information symbols.

3.4 Multiple Description Codes

RDOP problem can also be used for optimizing the partitioning in several multiple description scenarios specifically tailored for progressive embedded coders, such as ((Jiang and Ortega, 1999; Subbalakshmi and Somasundaram, 2002; Tillo *et al.*, 2007; Baccaglini *et al.*, 2007)). Consider first the following general MDC framework. To produce N descriptions these schemes divide the set of samples into N subsets, and each subset is encoded using a successively refinable coder. Thus N independently decodable embedded substreams are produced. We can regard them as the N source packets. Each source packet constitutes the basis for each side description. However to completely construct the descriptions, redundancy is added in order to create correlations between descriptions. In particular, description n is constructed by interleaving the source symbols of source packet S_n with redundancy symbols. Each redundancy symbol is a function of source symbols from all or only some source packets. If some description n does not arrive at the destination, then the decoder recovers part of the source symbols in S_n based on the redundancy symbols from received packets. Thus, the RDOP problem can be solved to optimize the partitioning. Notice that the MUEP scheme also conforms to this general MDC framework. Each channel packet (i.e., column in PA) can be regarded as a description. The redundancy symbols included in the descriptions are the R-S parity check symbols, which are functions of source symbols from all packets, situated in the same row of the PA. The MDC techniques proposed in ((Jiang and Ortega, 1999; Subbalakshmi

and Somasundaram, 2002; Tillo *et al.*, 2007; Baccaglini *et al.*, 2007)), also fit into this general MDC scenario, where redundancy symbols are copies of source symbols from other source packets. Indeed, their common feature is the partitioning of the set of samples into N subsets $P_1, \cdots, P_N$. Then each subset is encoded at different rates. Finally, the n-th description is formed from the source bitstream encoding P_n at the highest rate and a bitstream for each of the remaining P_j's at some lower rate. We can regard the source source bitstream encoding P_n at the highest rate as the source packet S_n and the remaining added symbols as redundancy. When the source coder is a progressive coder, any bitstream encoding P_j at lower rate is a prefix of S_j, thus the redundant symbols are actually copies of source symbols from other packets. Moreover, the symmetry in the construction of redundancy symbols for different packets, renders the associated RDOP problem an instance of the OPUF problem.

Chapter 4

Solution Algorithms

In this chapter we discuss globally optimal solutions to the RDOP problem. First we present in section 4.1 a dynamic programming algorithm, which is an extension of the algorithm of (Wu *et al.*, 2001). The other two solution algorithms exploit the property of total monotonicity to expedite the computations. One of them uses the fast matrix search algorithm of (Aggarwal *et al.*, 1987) to speed up the dynamic programming, but the improvement in speed is modest. Finally, the proposed divide-and-conquer algorithm accelerates the globally optimal solution significantly and constitutes the main contribution of this thesis.

The dynamic programming algorithm is discussed in section 4.1. Section 4.2 presents an equivalent formulation of the RDOP problem as a series of matrix search problem in totally monotone matrices. Finally, section 4.3 and 4.4 discuss the two faster solution algorithms, respectively.

4.1 Dynamic Programming Solution

We will present the solution for the general OPUV problem and specify when considering the special cases of OPUF or OPE leads to computational savings. The dynamic programming solution follows immediately from Wu *et al.* (2001). Specifically, let $\Delta D_{opt}(n,k)$ denote the maximum distortion decrease achieved by partitioning the subset of basic streams $B_1, B_2, \ldots, B_k$ into n source packets. Then the following recursion holds

$$\Delta D_{opt}(n,k) = \max_{1\le j<k} \{\Delta D_{opt}(n-1,j) + w_n(j+1,k)\}. \tag{4.1}$$

for $1 \le n \le N$, $1 \le k \le K$, where $D_{opt}(0,j) = D_{opt}(n,0) = 0$ for $0 \le j \le K-1$, $1 \le n \le N$. The dynamic programming algorithm computes the values $\Delta D_{opt}(n,k)$ for all $1 \le n \le N$ and $1 \le k \le K$, in lexicographical order of pairs n,k. At the end, the quantity $\Delta D_{opt}(N,K)$ is the maximum $\Delta D(\mathbf{p})$ sought of. If all values $w_n(j,k)$ are already known, then the number of operations to solve (4.1) for fixed n,k is $O(K)$, amounting to a total of $O(K^2N)$ operations over all pairs n,k.

On the other hand, the computation of the N upper triangular matrices W_n with elements $w_n(j,k)$, $1 \le j \le k \le K$, $1 \le n \le N$, has to be accounted for as well. For this let us denote $L = \max\{L_1, \cdots, L_n\}$ and let us discuss first the OPUF problem, where a single matrix W of weights has to be computed (let $W = W_1$). Recall that in OPUF all source packet lengths are equal, and thus equal to L. To calculate the elements of matrix W, one proceeds as follows. The matrix columns are filled from left to right. For each column k, its elements are computed starting at the main diagonal ($j = k$) and progressing up (i.e., by decreasing j). For each pair j,k with

$j < k$, the basic stream B_j is merged with the previously computed composite stream $B(j+1,k)$, to form the composite stream $B(j,k)$. Then $w(j,k)$ is computed, the memory for $B(j+1,k)$ is deallocated, while $B(j,k)$ is stored to be used at the next step. Notice that only the length L prefix of each composite stream $B(j,k)$ needs to be evaluated and the merge procedure to accomplish this goal requires only $O(L)$ running time, while the number of multiplications and additions used to evaluate $w(j,k)$ amounts to $O(L)$ operations as well. Thus, the whole upper triangular matrix is computed in $O(K^2L)$ operations. Finally, for OPUV, after having constructed the length L prefix of the composite stream $B(j,k)$, all N entries $w_n(j,k)$, $1 \leq n \leq N$, are computed. Thus, the total time needed for all N matrices is $O(K^2LN)$.

In conclusion, the time complexity of the dynamic programming solution is $O(K^2LN)$ for OPUV and $O(K^2(L+N))$ for OPUF and OPE. As for the space complexity, notice that, in order to store temporary results only $O(\max(K,L))$ storage space is enough, while $O(K^2)$ memory locations are required to store each matrix W_n. Thus, the algorithm space complexity is $O(\max(K^2N,L))$ for OPUV and $O(\max(K^2,L))$ for OPUF and OPE.

4.2 Equivalent Matrix Search Problem Formulation

In this section we reveal an equivalent problem formulation, which is the first step toward the faster solution algorithms.

Specifically, the RDOP problem can be regarded as the problem of finding the column maxima in N matrices. To see this, first we need to introduce the following

notation. For each $1 \leq n \leq N$, let M_n denote the $K \times K$-dimensional upper triangular matrix with elements defined as follows

$$M_n(j,k) = \Delta D_{opt}(n-1,j) + w_n(j+1,k), \tag{4.2}$$

for all $0 \leq j < k \leq K$. It is clear now that solving (4.1) for fixed n and all k, is equivalent to finding the maximum value in each column of the matrix M_n. We will also refer to this problem as the matrix search problem. This observation alone is not sufficient to speed up the computations, however, as we will soon see, each matrix M_n has a special property, termed total monotonicity, which serves this purpose. Specifically, an upper triangular matrix M_n is said to be totally monotone (Aggarwal *et al.*, 1987) if the following relation holds

$$M_n(j,k) \leq M_n(j',k) \Rightarrow M_n(j,k') \leq M_n(j',k') \quad j < j' < k < k'. \tag{4.3}$$

The following result is proved in Appendix A.

Proposition 1. For every $1 \leq n \leq N$, the upper triangular matrix M_n is totally monotone.

4.3 Solution based on SMAWK Algorithm

The SMAWK algorithm introduced in (Aggarwal *et al.*, 1987) solves the matrix search problem in a $K \times K$ totally monotone matrix in $O(K)$ time. In light of comparison against the divide-and-conquer algorithm discussed later, we will briefly review the SMAWK algorithm in this section. Further details are given in Appendix B.

Let us denote by $j(k)$, the largest row index such that $M_n(j(k), k)$ is the maximum element in column k. The solution to the maximum column search problem finds $j(k)$ for all $1 \leq k \leq K$. SAMWK algorithm makes use of the total monotonicity and is composed of two key subroutines, *MAXCOMPUTE* and *REDUCE*. In general, the *REDUCE* subroutine takes as input an $n \times m$ totally monotone matrix G_1, with $n \geq m$ and produces an $m \times m$ submatrix G_2, such that for $1 \leq k \leq m$, submatrix G_2 contains row $j(k)$ from G_1, i.e., the row that contains the maximum of column k. The *MAXCOMPUTE* subroutine takes as input an $n \times m$ totally monotone matrix G_1, with $n \geq m$ and invokes the *REDUCE* to get an $m \times m$ submatrix G_2 followed by a recursively call *MAXCOMPUTE* on the even columns submatrix of G_2, based on which the remaining column maxima on odd columns is computed.

According to Appendix B, SMAWK algorithm can be described as alternating invocations on *MAXCOMPUTE* and *REDUCE*. We solve the problem by invoking *MAXCOMPUTE*(M_n), where M_n is of size $K \times K$. The second *MAXCOMPUTE* takes a matrix of size $K \times K/2$ as argument. The algorithm continues invoking *MAXCOMPUTE* and *REDUCE* in turns until the last *REDUCE* outputs a matrix of size 1×1.

If we know the positions of the maxima in the even columns in an $K \times K$ matrix, following the total monotonicity, it only takes $O(K)$ time to find the maxima in the odd columns. We already know that the call to *REDUCE* takes linear time, i.e., $O(K)$. Thus, if $T(K)$ denotes the time taken by $MAXCOMPUTE$, with the input matrix of size $K \times K/2$, we have the following recursive definition. Notice that for simplicity, the analysis for the extra (first) invocation to reduce the matrix to $m \times m$

is omitted.

$$T(K) = T(K/2) + O(K) \tag{4.4}$$

By applying the master theorem (Cormen *et al.*, 2001) or using substitution, we obtain $T(K) = O(K)$. Thus, the SMAWK algorithm gives a linear time complexity solution.

Although SMAWK gives an elegant linear time solution to the matrix search problem, it needs the matrix elements to be known. The construction of the total monotone matrix requires the pre-computation of the "weight" terms $w_n(j,k)$ in (2.3), which takes quadratic time in K and thus offsets the advantages gained by using SMAWK.

In conclusion, the running time of SMAWK algorithm is $O(KN)$ for N matrices. In other words, SMAWK speeds up the computation of (4.1). However, the pre-computation of weights for OPUF and OPE problem is still $O(K^2L)$ and for OPUV, $O(K^2LN)$. Therefore, the total running time for OPUF and OPE is $O(KN + K^2L)$ and for OPUV is $O(KN + K^2LN)$. It is trivial to see that the improvement versus the dynamic programming solution is significant only when $N >> L$.

4.4 Divide-and-Conquer Algorithm

In this section, we discuss the divide-and-conquer algorithm. Due to Proposition 1, the matrix search algorithm, which normally requires all matrix elements to be examined, can be accelerated by leveraging a pleasing property of the row indices where the maxima occur. Specifically, recall that $j(k)$ denotes the row index where

the maximum occurs on column k. If the maximum occurs on several rows we take the largest index. Clearly, relation (4.3) implies that $j(1) \leq j(2) \leq \cdots \leq j(K)$. This property enables a fast divide and conquer algorithm to solve the matrix search problem. Notice that if the value $j(\lfloor K/2 \rfloor)$[1] is known, then, in order to find the maxima for the columns 1 through $\lfloor K/2 \rfloor - 1$, one only needs to look at the elements on the rows 1 through $j(\lfloor K/2 \rfloor)$. Likewise, to find the maxima on the columns with index higher than $\lfloor K/2 \rfloor$ one only needs to check the rows with indices higher or equal than $j(\lfloor K/2 \rfloor)$. In other words, the problem of searching matrix M_n is reduced to two subproblems of smaller sizes, namely searching submatrices $M_n(0 : j(\lfloor K/2 \rfloor), 1 : \lfloor K/2 \rfloor - 1)$ and $M_n(j(\lfloor K/2 \rfloor) : K-1, \lfloor K/2 \rfloor + 1 : K)$, where $M_n(j : j', k : k')$ denotes the submatrix containing the rows in the range j to j' and columns in the range k to k'. Further, the column maxima problem in each submatrix is solved recursively using the same procedure, i.e. first the maximum on the middle column is found, then the problem is divided into two subproblems, and so on. Notice that the divide and conquer algorithm does not examine all elements of M_n, and therefore not all the weights $w_n(j, k)$ need to be computed. Actually, it turns out that, only $O(K \log K)$ matrix elements need to be checked, thus only $O(K \log K)$ weights $w_n(j, k)$ have to be evaluated. We will see that the computations can be organized in such a way so that only $O(L)$ operations per checked matrix element to suffice, leading to a running time of $O(KL \log K)$ for searching matrix M_n.

The pseudocode of the divide and conquer algorithm (D&C, for short) is presented in Listing 4.4. D&C$(M_n, j_1 : j_2, k_1 : k_2)$ denotes the algorithm to solve the matrix search in submatrix $M_n(j_1 : j_2, k_1 + 1 : k_2)$.

[1]For any real value x, $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x.

```
1  D&C(M_n, j_1 : j_2, k_1 : k_2)
2  if k_1 + 1 > k_2 stop
3  else do k := ⌊(k_1 + k_2 + 1)/2⌋
4          if j_2 ≤ k_1 do
5                  for i := k_1 + 1 to k do
6                    B(j_2 + 1, i) := merge(B(j_2 + 1, i − 1), B_i)
7                    compute w_n(j_2 + 1, i) and M_n(j_2, i)
8                  for j := j_2 − 1 downto j_1 do
9                    B(j + 1, k) := merge(B(j + 2, k), B_{j+1})
10                   compute w_n(j + 1, k) and M_n(j, k)
11         else do
12                 for j := k − 1 downto j_1 do
13                 B(j + 1, k) := merge(B(j + 2, k), B_{j+1})
14                 compute w_n(j + 1, k) and M_n(j, k)
15         j(k) := arg max_{j_1 ≤ j ≤ min(j_2, k−1)} M_n(j, k)
16         D&C(M_n, j_1 : j(k), k_1 : k − 1)
17         D&C(M_n, j(k) : j_2, k : k_2)
```

Listing 4.1: Peudocode for the Divide-and-Conquer algorithm

In order to search matrix M_n, D&C$(M_n, 0 : K-1, 0 : K)$ is first invoked. The composite streams $B(j,k)$ once computed are stored[2] such that to be accessible by any subsequent recursive calls. This can be done by using a two dimensional array of linked lists as a global variable. Each composite stream is stored as a linked list in the corresponding position. It is easy to see that, if $j_2 \leq k_1$, then $B(j_2+1, k_1)$ has already

[2]Only the length L prefix of a composite stream has to be computed and stored.

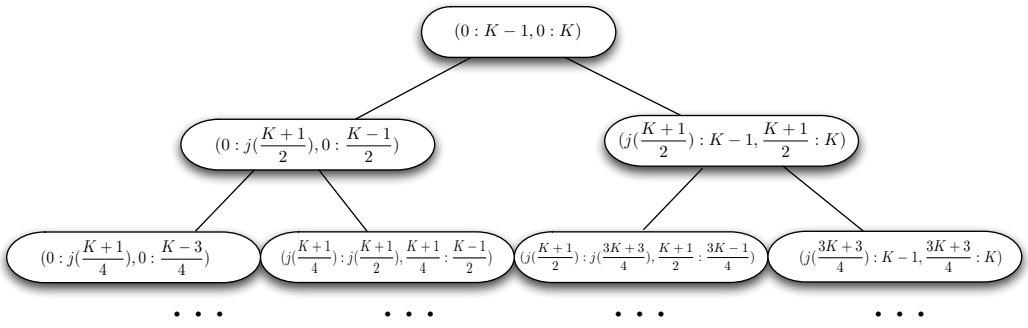


Figure 4.1: Recursion tree associated to the divide and conquer algorithm.

been evaluated during one of the previous invocations, therefore the pseudocode on line 6 is correctly defined. The merge procedure on lines 6, 9 and 13 takes only $O(L)$ operations. Lines 7, 10 and 14 also require $O(L)$ operations. The number of composite streams evaluated when $j_2 \leq k_1$ (lines 5-10), is $\lfloor (k_2 - k_1 + 1)/2 \rfloor + j_2 - j_1$, while for the case $j_2 > k_1$, it is $\lfloor (k_2 - k_1 + 1)/2 \rfloor - j_1 < (k_2 - k_1 + 1)/2 + j_2 - j_1$. We conclude that the running time spent to evaluate the composite streams is $O(L(k_2 - k_1 + j_2 - j_1))$. Since to execute line 15, $O(j_2 - j_1)$ operations are sufficient, it follows that the overhead needed to reduce the problem $M_n(j_1 : j_2, k_1 + 1 : k_2)$ into two subproblems is $O(L(k_2 - k_1 + j_2 - j_1))$.

In order to evaluate the total running time needed to execute D&C$(M_n, 0 : K - 1, 0 : K)$, it is instructive to analyze the associated recursion tree, where each node represents a subproblem solved recursively. In particular, the root represents the initial problem $M_n(0 : K - 1, 1 : K)$. For every tree node, the left, respectively right, child represents the first, respectively second, subproblem invoked recursively at that node. Fig. 4.1 illustrates levels $0, 1$ and 2 of the recursion tree.

To simplify the running time evaluation we assume that K is a power of 2. The total run-time can be obtained by adding up the overhead needed to reduce the

problem at every tree node. An important observation to aid this task is that the summation of the overheads for all nodes on any given level of the tree is $O(LK)$. To see this, notice first that any level l has 2^l nodes and each node corresponds to a submatrix with $K/2^l$ columns, i.e., $k_2 - k_1 = K/2^l$. Thus, the summation of values $k_2 - k_1$ over all nodes at this level is K. On the other hand, as we scan the nodes on level l from left to right, the value j_2 for a node (the last row in the submatrix) is identical to the value of j_1 for the next node (the first row). Thus, the summation of $j_2 - j_1$ over all nodes is $K - 1$. Now, having established that the total overhead on any level is $O(LK)$, and since the height of the recursion tree is $\log_2 K$, it follows that the number of operations to solve the matrix search problem is $O(LK \log K)$ as claimed. The space complexity of the algorithm is $O(K^2 + LK \log K)$ due to the need to store the composite streams.

Finally, to solve the RDOP problem (2.5), the divide and conquer algorithm is applied to search each of the matrices $M_1, \cdots, M_N$, yielding an $O(NLK \log K)$ time complexity algorithm. On the other hand, for the OPUF and OPE problems the weights $w_n(j, k)$ evaluated for some matrix M_n can be reused for subsequent matrices, thus their computation on lines 7, 10 and 14, and the invocation of the "merge" routine on lines 6, 9 and 13, are not performed all the time. Thus, the number of operations to search matrix M_n decreases as n increases. This observation suggest that for OPUF and OPE the running time estimate of $O(NLK \log K)$ is a loose upperbound, claim which is supported by our experimental results.

Chapter 5

Experimental Results

The main scope of this chapter is to assess empirically the speed of the proposed D&C algorithm for the three cases formulated in the previous section versus the dynamic programming (DP) counterpart. More specifically, we analyze the running time to solve the OPE, OPUF and OPUV problems introduced in chapter 2 by using the DP and D&C solutions described in chapter 4. The performance analysis based on the expected distortion at the receiver is also presented for the purpose of illustrating overall impact of using a rate distortion optimized partitioning versus an ad-hoc one.

Recall that the problem of RDOP is to find the partition that minimizes the expected distortion, or equivalently, finding the solution to (2.5). The OPE (OP with Equal coefficients) is an instance of the RDOP problem, where all the coefficients $\gamma(n, r)$ of equation (2.1) are equal. The OPE problem can also be interpreted as minimizing the distortion when all packets are decoded. This problem was considered in (Wu *et al.*, 2001) and solved via dynamic programming. The OPUF (OP with Unequal coefficients and Fixed across the packets) problem considers a scenario where the source packets lengths are equal and the coefficients might differ within individual

packet but remain the same across packets. This is useful in transmission schemes where all packets have the same redundancy allocation. The last case of RDOP problem is OPUV where the coefficients $\gamma(n, r)$ might be different within and across packets, corresponding to the case where the redundancy allocation differs from packet to packet.

This chapter is organized as follows. In section 5.1, the experiment settings are specified. The results for running time comparison between D&C and DP solutions are presented in section 5.2. Finally, section 5.3 presents the performance analysis.

5.1 Experiment Settings

Two images, Lena and Peppers of size 512×512 images were tested for the comparison. We used the QccSPIHT encoder from QccPack library (Fowler, 2008). The encoder is modified to output independent primary streams corresponding to SPIHT trees and the mean of the original image was subtracted to enhance the performance. A 5-level Cohen-Daubechies-Feauveau 9/7 wavelet transform (Cohen *et al.*, 1992) was applied, followed by a SPIHT encoder without arithmetic coding. The number of primary stream is $K = 256$, 192 of which corresponding to spatial orientation trees and 64 to non-root coefficients[1]. The rate distortion curve of each primary stream was approximated by its convex hull. After the optimal grouping was obtained, the primary streams in each group were interleaved according to their relative order of symbols in the original SPIHT stream.

In our experiments, the subband dispersed order introduced in (Dumitrescu *et al.*,

[1]Notice that each primary stream can be obtained independently by applying the SPIHT encoder only on the corresponding set of coefficients.

2010) is used as the fixed ordering of the basic streams. The packet header encoding the grouping side information contains $\log_2 K = 16$ bits. We assume that all the other side information is available at the decoder.

For the purpose of speed assessment we have considered all three cases of the RDOP problem, namely, OPUV, OPUF and OPE. Recall that for OPE all coefficients are equal to 1. The coefficients $\gamma(n, r)$ used in the OPUV problem were obtained assuming an FMUEP transmission scenario over a packet loss channel obeying an exponential model with packet loss rate $\mu = 0.05$. Specifically, we have computed the coefficients as follows. For each N and L we solved the FMUEP redundancy allocation problem assuming a total transmission budget of NL as in (Dumitrescu *et al.*, 2010) (notice that the size of a channel packet is assumed to be equal to L). Specifically, first the optimal values x_m, $1 \leq m \leq N$, were computed only based on the rate-distortion curve of the SPIHT coded stream for the whole image. Recall that x_m denotes the number of rows in layer m of the packetization array. Then the number of source symbols in each layer m was set to be the approximately the same for all packets, as follows. If the remainder of the division of mx_m by N was j, then we set $x_m^{(n)} = \lfloor mx_m/N \rfloor$ for $1 \leq n \leq N - j$, and $x_m^{(n)} = \lceil mx_m/N \rceil$ for $N - j \leq n \leq N$. Let L_n denote the number of source symbols in the n-th channel packet obtained according to this solution, after subtracting the two symbols used in the header. Then the coefficient $\gamma(n, r)$, for $1 \leq r \leq L_n$, was computed based on the redundancy protection assigned to the r-th symbol, while for $L_n < r \leq L$, we set $\gamma(n, r) = \gamma(n, L_n)$. Notice that the above procedure produces coefficient vectors $(\gamma(n, r))_{1 \leq r \leq L}$ very similar or even identical for different values of n. Thus, the corresponding RDOP problem is actually very close to OPUF. However, in order

N		8	16	24	32	40
L		819	409	273	204	163
OPE	D&C (s)	0.177	0.104	0.076	0.062	0.049
	DP (s)	17.226	8.650	5.767	4.408	3.469
OPUF	D&C (s)	0.184	0.112	0.082	0.069	0.055
	DP (s)	17.237	8.647	5.836	4.472	3.518
OPUV	D&C (s)	0.253	0.174	0.165	0.167	0.171
	DP (s)	19.077	10.535	7.810	6.620	5.979

Table 5.1: Lena $R_s = 0.2$

to evaluate the running time for the OPUV problem, we deliberately disregarded similarities of the coefficients from packet to packet and thus, treated the problem as a pure OPUV problem. Finally, for the OPUF problem solved in our tests, we used the coefficients corresponding to the first source packet in the OPUV problem.

Tests measuring the RDOP performance were done for independent packet loss (IPL) channel model and for exponential packet loss (EXP) channel model with mean packet loss rates $\mu = 0.05, 0.1$ and 0.15 respectively.

5.2 Running Time Evaluation

For the speed comparison we have run tests for two source coding rates R_s: $R_s = 0.2$ bpp and $R_s = 0.5$ bpp and number of packets $N = 8, 16, 24, 32, 40$. The $D\&C$ and DP algorithm were implemented in C language and were run on a 2.4 GHz Intel Core 2 Duo MacBook with 2GB 1067 MHz DDR3 main memory. We assume that all source packets have the same number L of symbols, thus, for each N and R_s, L is computed such that $R_s = 8 \cdot N(L+2)/(512 \times 512)$ (notice that $L+2$ is the length of the packet after including the header).

N		8	16	24	32	40
L		2048	1024	682	512	409
OPE	D&C (s)	0.585	0.247	0.174	0.130	0.113
	DP (s)	44.187	21.082	14.421	10.922	8.595
OPUF	D&C (s)	0.586	0.268	0.187	0.146	0.124
	DP (s)	45.899	21.365	14.548	10.962	8.699
OPUV	D&C (s)	0.758	0.487	0.378	0.363	0.371
	DP (s)	51.622	25.889	19.055	15.614	13.981

Table 5.2: Lena $R_s = 0.5$

N		8	16	24	32	40
L		819	409	273	204	163
OPE	D&C (s)	0.187	0.114	0.083	0.068	0.052
	DP (s)	17.211	8.663	5.775	4.435	3.480
OPUF	D&C (s)	0.193	0.118	0.087	0.072	0.059
	DP (s)	17.267	8.660	5.885	4.482	3.530
OPUV	D&C (s)	0.259	0.177	0.172	0.177	0.185
	DP (s)	19.160	10.595	7.877	6.719	6.075

Table 5.3: Peppers $R_s = 0.2$

The running time results are shown in Tables 5.1-5.4. Each table contains the running time in seconds for one image at one source coding rate. It can be seen that D&C is significantly faster than DP making RDOP feasible in practice. Moreover, for fixed value of the product NL (notice that this value is constant for each table), the running time spent by D&C to solve the OPUF problem consistently decreases as N increases, fact which supports the claim that the time complexity estimate of $O(NKL \log K)$ is not tight. Furthermore, the D&C algorithm is faster for the OPUF problem than for OPUV, as expected, the speed up being accentuated as N increases. However, even at $N = 40$, the the D&C solution for the OPUV problem

N		8	16	24	32	40
L		2048	1024	682	512	409
OPE	D&C (s)	0.641	0.272	0.193	0.146	0.125
	DP (s)	44.200	21.063	14.430	10.937	8.614
OPUF	D&C (s)	0.647	0.285	0.203	0.159	0.137
	DP (s)	46.155	21.279	14.602	11.111	8.771
OPUV	D&C (s)	0.807	0.507	0.396	0.383	0.394
	DP (s)	52.235	26.021	19.186	15.828	14.117

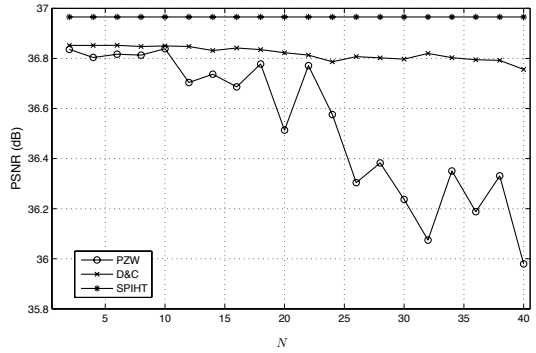
Table 5.4: Peppers $R_s = 0.5$

remains competitive with the OPE counterpart, being only ≈ 3 times slower.

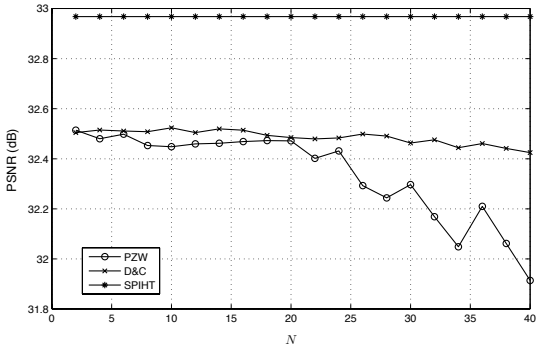
On the other hand, the running time values for OPE and OPUF are very close, for each of the two algorithms. Specifically, using the D&C algorithm, the running time for OPUF is higher than for OPE by at most 0.008, respectively 0.021, seconds when $R_s = 0.2$ bpp, respectively $R_s = 0.5$ bpp.

5.3 Performance Analysis

In this section we assess the impact in practice of using a rate-distortion optimized partitioning versus an ad-hoc one. To this end we consider two situations. The first situation is when the partitioning is optimized for minimum total distortion, i.e. by solving OPE. The OPE performance is assessed against the simple partitioning PZW proposed in (Rogers and Cosman, 1998), by comparing the PSNR values when all N packets are decoded. Fig. 5.1 plots the PSNR as a function of N, for OPE and PZW partitionings for Lena at source coding rates $R_s = 0.2$ and $R_s = 0.5$. For a fair comparison, the packet header in PZW contains 16 bits too. It can be observed that OPE outperforms PZW always for $N > 2$ (N takes even values up to 40) and



(a) rate=0.5

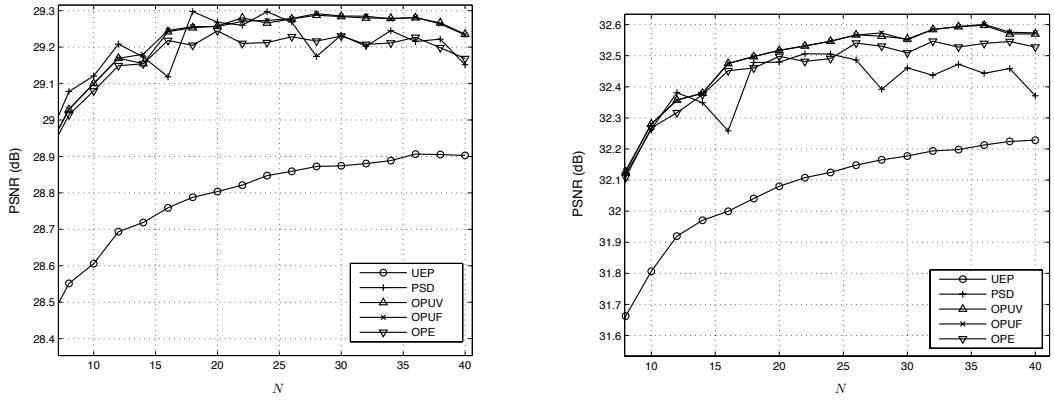


(b) rate=0.2

Figure 5.1: PSNR values at various N, for Lena; Comparison: D&C, PZW, SPIHT.

the gap becomes significant as N increases. The degradation in performance of a simple partitioning strategy like PZW as N increases, corroborated with the high speed of D&C, support the use in practice of OPE partitioning obtained via the D&C algorithm.

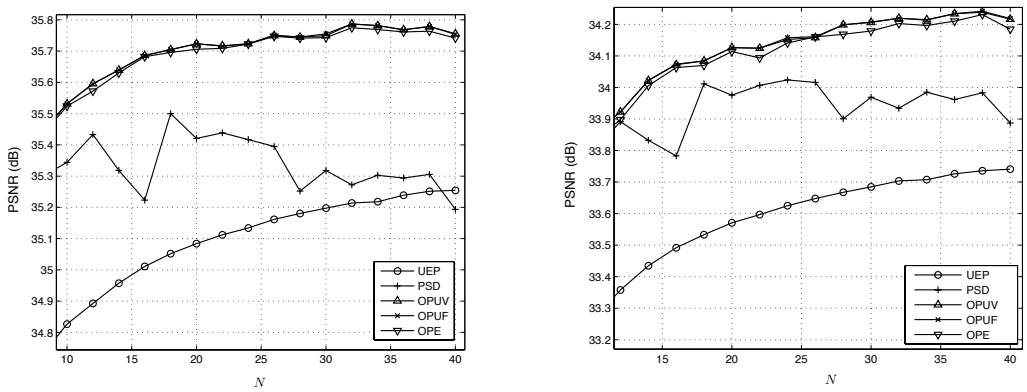
Fig. 5.1 also includes the PSNR achieved by SPIHT at the corresponding rate R_s, for comparison. As expected, the performance of OPE is worse. This loss is partly due to packetization constraints and partly to the overhead needed to encode the grouping. However, as the rate increases, this sacrifice in performance decreases. As such, at $R_s = 0.5$ bpp, the gap is only about 0.2 dB.



(a) rate=0.2 (b) rate=0.5

Figure 5.2: PSNR values for various N, for Lena. EXP channel with $\mu = 0.15$

The second scenario is FMUEP. We consider two transmission rates $R_t = 0.2$ bpp and $R_t = 0.5$ bpp. and two channel models: EXP (exponential packet loss) and IPL (independent packet loss) with three packet loss rates $\mu = 0.05, 0.1, 0.15$. The same two images are tested. For each transmission rate, all even values of N in the range 8 to 40 are considered. The FMUEP redundancy allocation is obtained by using the algorithm described in (Dumitrescu *et al.*, 2010). We compare four partitioning strategies, termed OPUV, OPE, OPUF and PSD. OPUV is the partitioning optimized for minimum expected distortion. OPE is optimized for minimum distortion when all N packets are received. OPUF is obtained by solving the OPUF problem, where all coefficients are as for the first source packet in the OPUV problem, thus the cost function is an approximation to the expected distortion. Finally, PSD is obtained by allocating an equal number of primary streams to each source packet, thus there is no need for side information to specify the partitioning. FMUEP with these partitioning strategies are also compared against UEP, which is the classic unequal erasure protection scheme (Dumitrescu *et al.*, 2004) applied to a progressive



(a) rate=0.2 (b) rate=0.5

Figure 5.3: PSNR values for various N, for Lena, $R_t = 0.5$ bpp, EXP channel with lower packet loss rate.

codestream, without having independently decodable source packets. All the aforementioned strategies are compared by using the expected distortion at the receiver as a performance measure.

Our tests reveal that OPUV outperforms PSD most of the time and the gap can reach up to 0.5 dB. The only cases when PSD is better are only for the EXP channel at high loss rate (0.15) and small or moderate values of N, and the margin is slim. Moreover, given fixed image, channel model, packet loss rate and transmission rate, the gap between OPUV and PSD tends to increase with N. In our tests the performance of OPUV and OPUF are almost identical (the difference is smaller than 0.01 dB) showing that for the FMUEP framework the RDOP problem can be solved faster by using the OPUF approximation with virtually no performance loss. As expected, the OPUV performance is greater or equal to that of OPE with a difference of at most 0.073 dB. This result demonstrates that optimizing the partitioning for minimum expected distortion can improve the performance versus a partitioning optimized for

minimum total distortion, even though the improvement is small.

We mention that the impact of OPE versus PSD was tested in (Dumitrescu *et al.*, 2010) as well, where the two partitionings were termed OptSD and PSD respectively. The experimental setting of (Dumitrescu *et al.*, 2010) is similar to ours with only one exception. In (Dumitrescu *et al.*, 2010), the image mean is not subtracted from the value of each pixel, before applying the wavelet transform, as we have done in our experiments. It turns out that subtracting the mean first from the image, enhances the performance of OPE since in our experiments OPE is always better than or competitive with PSD, while in (Dumitrescu *et al.*, 2010), for exponential channel with loss rate 0.15 the OPE results were far worse than PSD and even below UEP level (Dumitrescu *et al.*, 2010).

To illustrate our observations we plot in Figures 5.2 and 5.3 the performance of the above mentioned groupings for various values of N, for image Lena under EXP channel for the following settings: Fig. 5.2a): $R_t = 0.2$ bpp, $\mu = 0.15$; Fig. 5.2b): $R_t = 0.5$ bpp, $\mu = 0.15$; Fig. 5.3a): $R_t = 0.5$ bpp, $\mu = 0.05$; Fig. 5.3b): $R_t = 0.5$, $\mu = 0.1$. For each scheme the performance is measured by converting the expected distortion to PSNR.

In conclusion our experiments show that the proposed divide and conquer algorithm for the RDOP problem is very fast in practice. Moreover, our tests demonstrate that optimizing the partitioning by solving the RDOP problem can greatly improve the performance for the tested transmission schemes.

Chapter 6

Conclusion and Future Work

This work addresses the rate distortion optimal packetization (RDOP) of embedded streams into independent source packets. First we extend the previous problem formulation which aims at minimizing the distortion when all packets are received, to the minimization of the expected distortion for a wide range of transmission scenarios. Then we show that the existing dynamic programming algorithm can be extended to solve the new RDOP problem. Assuming convexity of the rate-distortion curves, the dynamic programming algorithm runs in $O(K^2LN)$ time, where N is the number of packets with L symbols each, and K is the number of primary streams (which are to be partitioned into N groups). Finally, under the same convexity assumption, we propose a fast divide and conquer algorithm which reduces the total time consumption to $O(NKL \log K)$. Furthermore, simulation results with SPIHT coded images demonstrate that the speed up is significant in practice.

The results presented in this thesis also provides a basis for future work. Due to limited memory resources, we were not able to run our algorithm under rate higher than 0.5 bpp. One area of future work, under the same availability of the resources, is

analyzing the running time and performance for higher bit rates. This can be done by combining more bits to form source symbols thus reducing the space requirements in the algorithm. Another research avenue is looking for different ways to form the basic streams. Other areas of future work include using the problem formulation in broader applications, incorporating the faster algorithm with other optimizing solutions, etc.

Appendix A

Proof of Proposition 1

This appendix presents the proof of Proposition 1. First let us restate this proposition.

Proposition 1. For every $1 \leq n \leq N$, the upper triangular matrix M_n is totally monotone.

Proof.

It can be easily seen that in order to prove relation (4.3), it is sufficient to show that

$$M_n(j,k) + M_n(j',k') \geq M_n(j',k) + M_n(j,k'), \qquad j < j' < k < k'.$$

By applying (4.2) and reducing the like terms, the above relation becomes

$$w_n(j+1,k) + w_n(j'+1,k') \geq w_n(j'+1,k) + w_n(j+1,k'),\ j < j' < k < k'. \quad \text{(A.1)}$$

Let us fix now n and indices j, j', k, k' satisfying $j < j' < k < k'$. In the remainder of this proof we will use the notation L instead of L_n for simplicity. Further, denote

by $a_1, a_2, \cdots, a_L$. the symbols in the composite stream $B(j'+1,k)$, labeled in the order they appear in the bitstream. Likewise, let $\mu_1, \mu_2, \cdots, \mu_L$, be the symbols in the composite stream $B(j+1,k)$, $\nu_1, \nu_2, \cdots, \nu_L$ the symbols in $B(j'+1,k')$, and $\rho_1, \rho_2, \cdots, \rho_L$, the symbols in $B(j+1,k')$. For any symbol η let us also denote by $\Delta(\eta)$ the decrease in distortion achieved by decoding that symbol. For convenience we will refer to the quantity $\Delta(\eta)$, as the "utility" associated to symbol η. Thus, since in any composite stream the symbols are ordered in non-increasing order of their utilities, the following relations hold Thus, by the definition of the composite streams, the following relations hold

$$\Delta(a_i) \geq \Delta(a_{i+1}), \Delta(\mu_i) \geq \Delta(\mu_{i+1}), \Delta(\nu_i) \geq \Delta(\nu_{i+1}), \Delta(\rho_i) \geq \Delta(\rho_{i+1}), \\ 1 \leq i \leq L-1. \qquad \text{(A.2)}$$

Recall that $B(j+1,k)$ is obtained by rate-distortion optimal interleaving of $B(j+1,j')$ and $B(j'+1,k)$. Therefore there must be some $s, 0 \leq s \leq L$, such that $B(j+1,k)$ contains the first s symbols of $B(j'+1,k)$, i.e., $a_1, \cdots, a_s$, and the first $L-s$ symbols of $B(j+1,j')$. For each $1 \leq \ell \leq s$, let us now denote by α_ℓ the position of symbol a_ℓ in $B(j+1,k)$. This means that the symbol μ_{α_ℓ} is identical to a_ℓ. Clearly, we have $\alpha_\ell < \alpha_{\ell+1}$, for all $1 \leq \ell \leq s-1$. Likewise, let $t, 0 \leq t \leq L$, be the number of symbols from $B(j'+1,k)$, included in $B(j'+1,k')$ and let $u, 0 \leq u \leq L$, denote the number of symbols from $B(j'+1,k)$, which appear in and $B(j+1,k')$. Furthermore, for each $1 \leq \ell \leq t$, let β_ℓ denote the position of symbol a_ℓ in $B(j'+1,k')$, and for $1 \leq \ell \leq u$, let δ_ℓ be the position of symbol a_ℓ in $B(j+1,k')$. It is easy to verify that $\beta_\ell < \beta_{\ell+1}$, for all $1 \leq \ell \leq t-1$, and $\delta_\ell < \delta_{\ell+1}$, for all $1 \leq \ell \leq u-1$.

To prove the claim we need first to rewrite conveniently each quantity $\Delta(\eta)$. Let

us start with the symbols in $B(j'+1,k)$. For each $\ell, 1 \leq \ell \leq L$, one can write

$$\Delta(a_\ell) = \Delta(a_L) + \sum_{m=\ell}^{L-1} (\Delta(a_m) - \Delta(a_{m+1})). \tag{A.3}$$

With the convention that $\alpha_0 = 0$ and $\alpha_{s+1} = L+1$, we further write for each ℓ and i such that $0 \leq \ell \leq s$ and $\alpha_\ell + 1 \leq i \leq \min(L, \alpha_{\ell+1})$,

$$\Delta(\mu_i) = (\Delta(\mu_i) - \Delta(a_{\ell+1})) + \Delta(a_L) + \sum_{m=\ell+1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1})). \tag{A.4}$$

The previous relation holds because $\ell+1 \leq L$. To see that the latter equality is valid, notice that, if there exists i such that $\alpha_\ell + 1 \leq i \leq \min(L, \alpha_{\ell+1})$, then necessarily $\alpha_\ell \leq L-1$, which implies $\ell \leq L-1$.

Likewise, let $\beta_0 = \delta_0 = 0$ and $\beta_{t+1} = \delta_{u+1} = L+1$ by convention. Then for each ℓ and i such that $0 \leq \ell \leq t$ and $\beta_\ell + 1 \leq i \leq \min(L, \beta_{\ell+1})$, rewrite

$$\Delta(\nu_i) = (\Delta(\nu_i) - \Delta(a_{\ell+1})) + \Delta(a_L) + \sum_{m=\ell+1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1})). \tag{A.5}$$

Finally, for each ℓ and i such that $0 \leq \ell \leq u$ and $\delta_\ell + 1 \leq i \leq \min(L, \delta_{\ell+1})$, write

$$\Delta(\rho_i) = (\Delta(\rho_i) - \Delta(a_{\ell+1})) + \Delta(a_L) + \sum_{j=\ell+1}^{L-1} (\Delta(a_j) - \Delta(a_{j+1})). \tag{A.6}$$

An important observation, to be used in the proof, is that all the expressions enclosed in parentheses in (A.3)-(A.6) are non-negative in virtue of relations (A.2).

Next we rewrite the terms in (A.1) using the definition (2.3) and (A.3)-(A.6). For simplicity we will use the notation γ_i instead of $\gamma(n,i)$, for all $1 \leq i \leq L$, since n is

fixed. For convenience we will also denote $\gamma_{L+1} = 0$. Then $w_n(j+1,k)$ can be written as follows

$$
\begin{aligned}
w_n(j+1,k) &= \sum_{i=1}^{L} \gamma_i \Delta(\mu_i) \\
&=^{(a)} \sum_{i=1}^{L+1} \gamma_i \Delta(\mu_i) \\
&= \sum_{\ell=0}^{s} \sum_{i=\alpha_\ell+1}^{\alpha_{\ell+1}} \gamma_i \cdot \Delta(\mu_i) \\
&=^{(b)} \sum_{\ell=0}^{s} \sum_{i=\alpha_\ell+1}^{\alpha_{\ell+1}} \gamma_i \cdot \big((\Delta(\mu_i) - \Delta(a_{\ell+1})) + \Delta(a_L) + \sum_{m=\ell+1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1}))\big) \\
&= \sum_{\ell=0}^{s} \sum_{i=\alpha_\ell+1}^{\alpha_{\ell+1}} \gamma_i \cdot (\Delta(\mu_i) - \Delta(a_{\ell+1})) \\
&\quad + \sum_{m=1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1})) \sum_{\ell=0}^{\min\{m-1,s\}} \sum_{i=\alpha_\ell+1}^{\alpha_{\ell+1}} \gamma_i \\
&\quad + \Delta(a_L) \sum_{i=1}^{L} \gamma_i \\
&= \sum_{\ell=0}^{s} \sum_{i=\alpha_\ell+1}^{\alpha_{\ell+1}} \gamma_i \cdot (\Delta(\mu_i) - \Delta(a_{\ell+1})) \\
&\quad + \underbrace{\sum_{m=1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1})) \sum_{i=1}^{\alpha_{\min\{m-1,s\}+1}} \gamma_i}_{T_1} \\
&\quad + \Delta(a_L) \sum_{i=1}^{L} \gamma_i, \qquad \text{(A.7)}
\end{aligned}
$$

where equality (a) follows from $\gamma_{L+1} = 0$, and (b) is based on (A.4). Similarly, using (A.5) and (A.6), the quantities $w_n(j'+1,k')$ and $w_n(j+1,k')$ can be expressed as

follows

$$w_n(j'+1,k') = \sum_{\ell=0}^{t} \sum_{i=\beta_\ell+1}^{\beta_{\ell+1}} \gamma_i \cdot (\Delta(\nu_i) - \Delta(a_{\ell+1})) + \underbrace{\sum_{m=1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1})) \sum_{i=1}^{\beta_{\min\{m-1,t\}+1}} \gamma_i}_{T_2} + \Delta(a_L) \sum_{i=1}^{L} \gamma_i, \tag{A.8}$$

$$w_n(j+1,k') = \sum_{\ell=0}^{u} \sum_{i=\delta_\ell+1}^{\delta_{\ell+1}} \gamma_i \cdot (\Delta(\rho_i) - \Delta(a_{\ell+1})) + \underbrace{\sum_{m=1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1})) \sum_{i=1}^{\delta_{\min\{m-1,u\}+1}} \gamma_i}_{T_3} + \Delta(a_L) \sum_{i=1}^{L} \gamma_i. \tag{A.9}$$

Finally, based on (A.3), we obtain

$$w_n(j'+1,k) = \sum_{i=1}^{L} \gamma_i \Delta(a_i) = \underbrace{\sum_{m=1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1})) \sum_{i=1}^{m} \gamma_i}_{T_4} + \Delta(a_L) \sum_{i=1}^{L} \gamma_i. \tag{A.10}$$

In order to prove relation (A.1), we first show that

$$T_1 + T_2 \geq T_3 + T_4. \tag{A.11}$$

For this notice that

$$T_1 + T_2 = \sum_{m=1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1}))[\sum_{i=1}^{\alpha_{\min\{m,s+1\}}} \gamma_i + \sum_{i=1}^{\beta_{\min\{m,t+1\}}} \gamma_i], \tag{A.12}$$

$$T_3 + T_4 = \sum_{m=1}^{L-1} (\Delta(a_m) - \Delta(a_{m+1}))[\sum_{i=1}^{\delta_{\min\{m,u+1\}}} \gamma_i + \sum_{i=1}^{m} \gamma_i]. \tag{A.13}$$

Since $\Delta(a_m) - \Delta(a_{m+1}) \geq 0$ for all $1 \leq m \leq L-1$, in order to prove (A.11), it is sufficient to show that

$$\sum_{i=1}^{\alpha_{\min\{m,s+1\}}} \gamma_i + \sum_{i=1}^{\beta_{\min\{m,t+1\}}} \gamma_i \geq \sum_{i=1}^{\delta_{\min\{m,u+1\}}} \gamma_i + \sum_{i=1}^{m} \gamma_i \tag{A.14}$$

A key observation used in the proof is the following. For any $1 \leq \ell \leq u-1$, the symbols in $B(j+1, k')$ appearing between a_ℓ and $a_{\ell+1}$, in other words the symbols $\rho_{\delta_\ell+1}, \cdots, \rho_{\delta_{\ell+1}-1}$, are actually the symbols from the set $\{\mu_{\alpha_\ell+1}, \cdots, \mu_{\alpha_{\ell+1}-1}, \nu_{\beta_\ell+1}, \cdots, \nu_{\beta_{\ell+1}-1}\}$, ordered in non-increasing order of their utilities. This implies that $\delta_{\ell+1} - \delta_\ell - 1 = \alpha_{\ell+1} - \alpha_\ell - 1 + \beta_{\ell+1} - \beta_\ell - 1$. Summation over all ℓ such that $1 \leq \ell \leq m-1$, for $1 \leq m \leq u$, further yields

$$\delta_m = \alpha_m + \beta_m - m, \quad 1 \leq m \leq u. \tag{A.15}$$

The definitions of α_m and β_m imply that

$$m \leq \alpha_m, \qquad m \leq \beta_m, \quad 1 \leq m \leq u. \tag{A.16}$$

The above inequalities combined with (A.15) further lead to

$$\delta_m \geq \alpha_m, \qquad \delta_m \geq \beta_m. \tag{A.17}$$

To prove (A.14) we will distinguish between the following two cases: 1) $m \leq u$; 2) $m > u$.

Case 1: $m \leq u$. Since $u \leq \min\{s, t\}$, it follows that $m \leq s$ and $m \leq t$. Thus, (A.14) becomes

$$\sum_{i=1}^{\alpha_m} \gamma_i + \sum_{i=1}^{\beta_m} \gamma_i \geq \sum_{i=1}^{\delta_m} \gamma_i + \sum_{i=1}^{m} \gamma_i. \tag{A.18}$$

We may assume without restricting the generality that $\alpha_m \leq \beta_m$ (the other case is symmetrical). Then, by using (A.17) and (A.16), (A.18) can be rewritten as relation

$$\sum_{i=m+1}^{\alpha_m} \gamma_i \geq \sum_{i=\beta_m+1}^{\delta_m} \gamma_i. \tag{A.19}$$

By (A.15), the number of terms on both sides of the above inequality is the same. Moreover, since $\gamma_i \geq \gamma_{i+1}$, $1 \leq i \leq L-1$, (according to (2.2)) and $\alpha_m \leq \beta_m$, it follows that any term on the left hand side is larger than or equal to any term on the right hand side. These facts establish the validity of relation (A.19).

Case 2: $m \geq u + 1$. In this case, one has $\delta_{\min\{m,u+1\}} = L + 1$. If $m \geq t + 1$ then $\beta_{\min\{m,t+1\}} = L+1$, and since $\alpha_{\min\{m,s+1\}} \geq m$ clearly holds, inequality (A.14) follows. The situation when $m \geq s + 1$ can be treated similarly. Therefore, let us consider

now the case $m < s+1$ and $m < t+1$. Now (A.14) becomes

$$\sum_{i=1}^{\alpha_m} \gamma_i + \sum_{i=1}^{\beta_m} \gamma_i \geq \sum_{i=1}^{L} \gamma_i + \sum_{i=1}^{m} \gamma_i, \tag{A.20}$$

where we have used the fact that $\gamma_{L+1} = 0$. Relations $u+1 \leq m < s+1$ and $u+1 \leq m < t+1$ imply that $u < s$ and $u < t$, hence symbol a_{u+1} appears in both $B(j+1,k)$ and $B(j'+1,k')$. On the other hand, this symbol does not appear in $B(j+1,k')$, fact which leads to the conclusion that $L-\delta_u \leq \alpha_{u+1}-\alpha_u-1+\beta_{u+1}-\beta_u-1$. Combining with $\delta_u = \alpha_u + \beta_u - u$, which holds according to (A.15), one further obtains that $L \leq \alpha_{u+1} - 1 + \beta_{u+1} - 1 - u$. The previous inequality together with $\alpha_{u+1} \leq \alpha_m + (u+1) - m$[1] and $\beta_m \geq \beta_{u+1} + m - (u+1)$, lead to

$$L \leq \alpha_m + \beta_m - m. \tag{A.21}$$

By symmetry we may again assume that $\alpha_m \leq \beta_m$. Then (A.20) can be rewritten as

$$\sum_{i=m+1}^{\alpha_m} \gamma_i \geq \sum_{i=\beta_m+1}^{L} \gamma_i. \tag{A.22}$$

By (A.21), the number of terms on the left hand side is larger than or equal to the number of terms on the right hand side. Combining with the monotonicity relations $\gamma_i \geq \gamma_{i+1}$ $1 \leq i \leq L-1$, and with $\alpha_m \leq \beta_m$, inequality (A.22) follows. Now the proof of (A.11) is complete.

[1]This relation follows by summing inequalities $\alpha_i \leq \alpha_{i+1} - 1$ for $u+1 \leq i \leq m-1$. A similar argument is valid for the next relation.

In order to finalize the proof of relation (A.1) it remains to show that

$$\sum_{\ell=0}^{s}\sum_{i=\alpha_\ell+1}^{\alpha_{\ell+1}}\gamma_i\cdot(\Delta(\mu_i)-\Delta(a_{\ell+1}))+\sum_{\ell=0}^{t}\sum_{i=\beta_\ell+1}^{\beta_{\ell+1}}\gamma_i\cdot(\Delta(\nu_i)-\Delta(a_{\ell+1}))\geq \sum_{\ell=0}^{u}\sum_{i=\delta_\ell+1}^{\delta_{\ell+1}}\gamma_i\cdot(\Delta(\rho_i)-\Delta(a_{\ell+1})) \qquad (A.23)$$

Let us denote by Z_i the general term in the summation on the right hand side, i.e., $Z_i=\gamma_i\cdot(\Delta(\rho_i)-\Delta(a_{\ell+1}))$, for any $1\leq\ell\leq u$ and $\delta_\ell+1\leq i\leq\delta_{\ell+1}$. Likewise, let $U_i=\gamma_i\cdot(\Delta(\mu_i)-\Delta(a_{\ell+1}))$ for any $1\leq\ell\leq s$, $\alpha_\ell+1\leq i\leq\alpha_{\ell+1}$, and let $V_i=\gamma_i\cdot(\Delta(\nu_i)-\Delta(a_{\ell+1}))$ for any $1\leq\ell\leq t$, $\beta_\ell+1\leq i\leq\beta_{\ell+1}$. The idea of the proof is the following. We will show each Z_i is either 0 or it has a corresponding term on the left hand side (LHS), which is larger or equal than Z_i. Moreover, this correspondence assigns to each LHS term at most one term on the right hand side. Since all unassigned LHS terms are non-negative, relation (A.23) follows.

Let us fix some ℓ and i such that, $1\leq\ell\leq u$ and $\delta_\ell+1\leq i\leq\delta_{\ell+1}$. Clearly, if $i=\delta_{\ell+1}$ then one has $\gamma_i=0$ (when $\ell=u$) or $\Delta(\rho_i)-\Delta(a_{\ell+1})=0$ (when $\ell<u$), hence $Z_i=0$. On the other hand, if $i\neq\delta_{\ell+1}$, then symbol ρ_i must be either a symbol from the set $\{\mu_{\alpha_\ell+1},\cdots,\mu_{\alpha_{\ell+1}-1}\}$, or from the set $\{\nu_{\beta_\ell+1},\cdots,\nu_{\beta_{\ell+1}-1}\}$. In the former case, $\rho_i=\mu_{p(i)}$ for some index $p(i)$ such that $\alpha_\ell+1\leq p(i)<\alpha_{\ell+1}$, therefore we let $U_{p(i)}$ be the LHS term corresponding to Z_i. It is easy to see that $p(i)\leq i$, thus $\gamma_{p(i)}\geq\gamma_i$ by (2.2). Combining further with the fact that $\Delta(\rho_i)-\Delta(a_{\ell+1})\geq 0$, one obtains that $U_{p(i)}\geq Z_i$. Finally, in the latter case, let $q(i)$ be the index such that $\rho_i=\nu_{q(i)}$ and $\beta_\ell+1\leq q(i)<\beta_{\ell+1}$. It can be easily verified that $q(i)\leq i$, therefore $\gamma_{q(i)}\geq\gamma_i$. We let $V_{q(i)}$ be the LHS term corresponding to Z_i, and $V_{q(i)}\geq Z_i$ holds. With these observations the proof of (A.23) is complete.

Appendix B

Review of SMAWK Algorithm

In this chapter, we present in detail the SMAWK algorithm solution, which is used in section 4.3. The SMAWK algorithm finds the largest row index $j(k)$, where $j(k)$ is the maxima in column k, in an $n \times m$ totally monotone matrix M for all $1 \leq k \leq m$. The following material is adapted from (Aggarwal *et al.*, 1987).

Recall that for an $n \times m$ totally monotone matrix M, the following relations hold, (see Fig B.1)

$$M(j,k) \leq M(j',k) \Rightarrow M(j,k') \leq M(j',k') \qquad \text{for all } j < j' \text{ , } k < k', \qquad \text{(B.24)}$$

We say that an entry $M(j,k)$ is dead if $j \neq j(k)$. A row is dead if all of its entries are dead. The following lemma is the basis of *REDUCE* subroutine.

Lemma B.1 *Let M be a totally monotone $n \times m$ matrix and let $1 \leq j \leq j' \leq n$. If $M(j,k) \geq M(j',k)$ then the entries in $\{M(j',r) : 1 \leq r \leq k\}$ are dead. On the other hand, if $M(j,k) < M(j',k)$ then all the entries in $\{M(j,r) : k \leq r \leq m\}$ are dead.*

The proof can be directly obtained from the total monotonicity (see Fig B.2). The

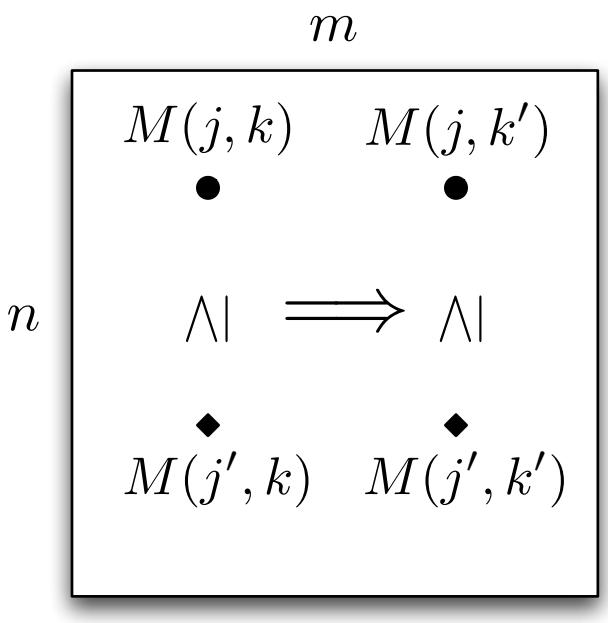
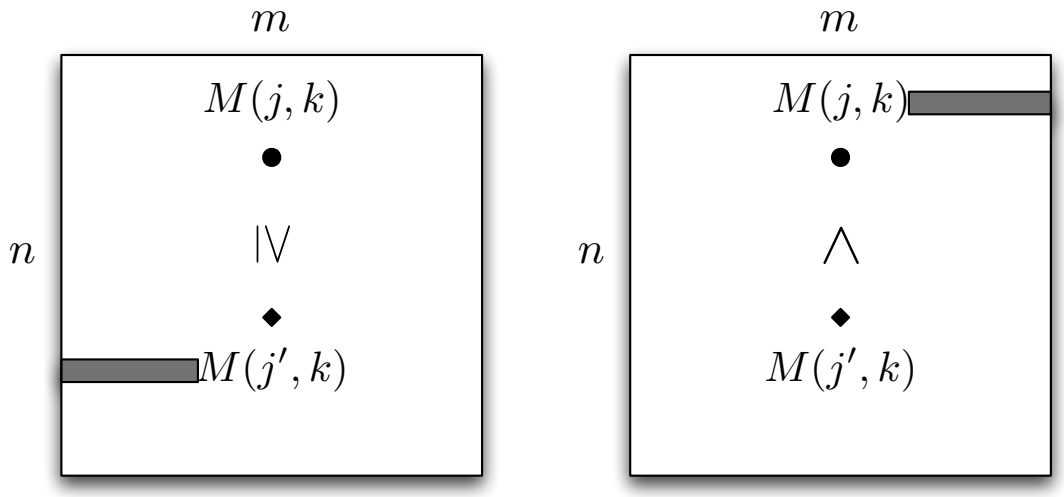


Figure B.1: Illustration of total monotonicity

first claim follows from the fact that $M[j, j'; r', r]$ is totally monotone for all $1 \leq r' < r$. Similarly, the second claim follows directly from the fact that $M[j, j'; r, r'']$ is monotone for all $r < r'' \leq m$.

Now let us describe *REDUCE* subroutine. Let matrix C denote the matrix returned by *REDUCE*. A denotes the input total monotone matrix. Notice that both A and C are 1-based matrices, (i.e., indices start with 1). The pseudocode of *REDUCE* is presented in Listing B.1.



(a) $M(j,k) \geq M(j',k)$ (b) $M(j,k) < M(j',k)$

Figure B.2: Illustration of how the dead elements are determined after one comparison based on Lemma B.1. Gray boxes represent dead elements

```
1  REDUCE(A):
2      C ← A; r ← 1
3      while C has more than m rows
4              case
5              C(r,r) ≥ C(r+1,r) and r < m: r ← r+1
6              C(r,r) ≥ C(r+1,r) and r = m: Delete row r+1
7              C(r,r) < C(r+1,r):Delete row r; if r > 1 then r ← r−1
8              endcase
9      return C
```

Listing B.1: pseudocode for REDUCE subroutine

The invariant maintained in the while loop is that for all $1 \leq j \leq r$ and $1 \leq k < j$, entry $C(j,k)$ is dead. Only dead rows are deleted in each case branch. Initially, the

invariant holds since C starts with index 1. If $C(r,r) \geq C(r+1,r)$ then according to Lemma B.1, all elements of row $r+1$ in columns 1 through r are dead. Therefore, if $r < m$, r is incremented by 1 and if $r = m$, row $r+1$ is dead and r remains the same. On the other hand, if $C(r,r) < C(r+1,r)$ then by Lemma B.1 all entries of row r in columns r through m are dead and because the entries of row r in columns 1 through $r-1$ are already dead, the entire row r is dead.

To prove that the subroutine *REDUCE* makes $O(n)$ comparisons, we need to introduce three variables a, b, and c to denote the number of first, second and third executed branches of the case statement respectively. Clearly, the total time is $t = a+b+c$. Rows are only deleted in the last two branches, and since the total number of deleted rows is $n-m$, one has $b+c = n-m$. The index k gets incremented in the first case and decremented or stays the same in the last case. Let Δ denote the net increase in the index k. Clearly $a - c \leq \Delta \leq m-1$. Combining the above facts, we have

$$\begin{aligned} t &= a+b+c \\ &\leq a+2b+c \\ &= a-c+2b+2c \\ &\leq m-1+2n-2m = 2n-m-1 \end{aligned} \tag{B.25}$$

Thus, the time complexity of *REDUCE* is $O(n)$.

MAXCOMPUTE recursively solves the maximum problem on an $n \times m$ totally monotone matrix, where $n \geq m$. The pseudocode is presented in Listing B.2.

```
1 MAXCOMPUTE(A):
2     B ← REDUCE(A)
3     if n = 1 then output the maximum and return
4     C ← B[1, 2, ..., m; 2, 4, ..., 2⌊m/2⌋]
5     MAXCOMPUTE(C)
6     From the known positions of the maxima in the even columns of B,
          find the maximum in its odd rows
7     end
```

Listing B.2: pseudocode for MAXCOMPUTE subroutine

To sum up, SMAWK algorithm invokes *MAXCOMPUTE* and *REDUCE* alternatively until the base case in line 3 of Listing B.2 is reached. Specifically, the RDOP problem can be solved by invoking *MAXCOMPUTE(M_n)* for each n, $1 \leq n \leq N$.

Bibliography

Aggarwal, A., Klawe, M. M., Moran, S., Shor, P. W., and Wilber, R. E. (1987). Geometric applications of a matrix-searching algorithm. *Algorithmica*, **2**, 195–208.

Akyol, E., Tekalp, A., and Civanlar, M. (2007). A flexible multiple description coding framework for adaptive peer-to-peer video streaming. *Selected Topics in Signal Processing, IEEE Journal of*, **1**(2), 231 –245.

Baccaglini, E., Tillo, T., and Olmo, G. (2007). A flexible r-d-based multiple description scheme for jpeg 2000. *Signal Processing Letters, IEEE*, **14**(3), 197 –200.

Chande, V. and Farvardin, N. (2000). Progressive transmission of images over memoryless noisy channels. *Selected Areas in Communications, IEEE Journal on*, **18**(6), 850 –860.

Cohen, A., Daubechies, I., and Feauveau, J. C. (1992). Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, **45**(5), 485–560.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to algorithms, second edition.

Creusere, C. D. (1996). Robust image coding using the embedded zerotree wavelet algorithm. In *Proceedings of the Conference on Data Compression*, pages 432–, Washington, DC, USA. IEEE Computer Society.

Crump, V. and Fischer, T. (1997). Intraframe low bit rate video coding robust to packet erasure. In *Data Compression Conference, 1997. DCC '97. Proceedings*, page 432.

Dumitrescu, S., Wu, X., and Wang, Z. (2004). Globally optimal uneven error-protected packetization of scalable code streams. *IEEE Transactions on Multimedia*, **6**(2), 230–239.

Dumitrescu, S., Wu, X., and Wang, Z. (2007). Efficient algorithms for optimal uneven protection of single and multiple scalable code streams against packet erasures. *Multimedia, IEEE Transactions on*, **9**(7), 1466 –1474.

Dumitrescu, S., Rivers, G., and Shirani, S. (2010). Unequal erasure protection technique for scalable multistreams. *IEEE Transactions on Image Processing*, **19**(2), 422–434.

Fowler, J. E. (2008). Qccpack–quantization,compression, and coding library, http://qccpack.sourceforge.net/.

H. Man, F. K. and Smith, M. (1997). Robust ezw image coding for noisy channels. *Signal Processing Letters, IEEE*, **4**, 227–229.

Hamzaoui, R., Stankovic, V., and Xiong, Z. (2005). Optimized error protection of scalable image bit streams [advances in joint source-channel coding for images]. *Signal Processing Magazine, IEEE*, **22**(6), 91 – 107.

Jiang, W. and Ortega, A. (1999). Multiple description coding via polyphase transform and selective quantization. *Proc. SPIE: Visual Communications and Image Processing*.

Kim, B.-J., Xiong, Z., and Pearlman, W. A. (2000). Low bit-rate scalable video coding with 3-d set partitioning in hierarchical trees (3-d spiht). *IEEE Trans. Circuits Syst. Video Techn.*, **10**(8), 1374–1387.

Mohr, A., Riskin, E., and Ladner, R. (2000). Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction. *Selected Areas in Communications, IEEE Journal on*, **18**(6), 819 –828.

Mohr, A. E., Riskin, E. A., and Ladner, R. E. (1999). Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction. *IEEE Journal on Selected Areas in Communications*, **18**, 819–828.

Puri, R. and Ramchandran, K. (1999). Multiple description source coding using forward error correction codes. In *Signals, Systems, and Computers, 1999. Conference Record of the Thirty-Third Asilomar Conference on*, volume 1, pages 342 –346 vol.1.

Rogers, J. K. and Cosman, P. C. (1998). Robust wavelet zerotree image compression with fixed-length packetization.

Said, A. and Pearlman, W. A. (1996). A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, **6**, 243–250.

Shapiro, J. M. (1993). Embedded image-coding using zerotrees of wavelet coeffficients. *IEEE Transactions on Signal Processing*, **41**, 3445–3462.

Stankovic, V., Hamzaoui, R., and Saupe, D. (2003). Fast algorithm for rate-based optimal error protection of embedded codes. *IEEE Trans. Commun*, **51**, 1788–1795.

Stankovic, V., Hamzaoui, R., and Xiong, Z. (2004). Efficient channel code rate selection algorithms for forward error correction of packetized multimedia bitstreams in varying channels. *Multimedia, IEEE Transactions on*, **6**(2), 240 – 248.

Stockhammer, T. and Buchner, C. (2001). Progressive texture video streaming for lossy packet networks.

Subbalakshmi, K. and Somasundaram, S. (2002). Multiple description image coding framework for ebcot. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages 541 – 544 vol.3.

Taubman, D. (2000). High performance scalable image compression with ebcot. *IEEE Transactions on Image Processing*, **9**, 1158–1170.

Taubman, D. S. and Marcellin, M. W. (2001). *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Norwell, MA, USA.

Thie, J. and Taubman, D. (2005). Optimal erasure protection strategy for scalably compressed data with tree-structured dependencies. *Image Processing, IEEE Transactions on*, **14**(12), 2002 –2011.

Thomos, N., Boulgouris, N. V., and Strintzis, M. G. (2006). Optimized transmission of

jpeg2000 streams over wireless channels. *IEEE Transactions on Image Processing*, **15**(1), 54–67.

Tillo, T., Grangetto, M., and Olmo, G. (2007). Multiple description image coding based on lagrangian rate allocation. *Image Processing, IEEE Transactions on*, **16**(3), 673 –683.

Tillo, T., Baccaglini, E., and Olmo, G. (2010). Multiple descriptions based on multirate coding for jpeg 2000 and h.264/avc. *Image Processing, IEEE Transactions on*, **19**(7), 1756 –1767.

Wu, X., Cheng, S., and Xiong, Z. (2001). On packetization of embedded multimedia bitstreams. *IEEE Transactions on Multimedia*, **3**(1), 132–140.