University of California

Los Angeles

Towards a Systematic Approach for Modeling and Optimizing Distributed and Dynamic Multimedia Systems

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical Engineering

by

Brian Foo

2008

The dissertation of Brian Foo is approved.

Richard Wesel

Mani Srivastava

Mario Gerla

Mihaela van der Schaar, Committee Chair

University of California

2008

To my parents …

who reminded me to take care of myself

whenever I became absent-minded

from thinking too much.

Table of Contents

List of Tables

List of Illustrations

ACKNOWLEDGEMENTS

I would like to start by thanking my advisor Mihaela van der Schaar for her enthusiasm and support through the course of my PhD. She has always encouraged me to look outside the details of specific problems and to see the big picture. Under her guidance, I was able to complete many papers on a wide range of research topics. Her breadth and creativity have been constant sources of inspiration for me throughout my stay here at UCLA.

I would also like to thank Professors Rick Wesel, Mani Srivastava, and Mario Gerla for their interest in my work, and their time invested to be part of my committee. During my qualifying exam, each of them have provided useful feedback to guide the direction of my work toward meaningful directions.

I would also like to thank my labmates Fangwen Fu, Hyunggon Park, Nick Mastronarde, Peter Hsiang, Yi Su, and Zhichu Lin for helping me to think through many of my research ideas, and for helping to peer review my papers before submission. I will also treasure the personal times spent with each of them, and how they have enriched my life through both meaningful and fun conversations.

I would also like to thank my supervisor at IBM, Deepak Turaga, as well as Olivier Verscheure, and my manager Lisa Amini, for giving me the opportunity to do very interesting research with them in their stream processing group. I would also like to thank the members of the Marvel image classification group, Rong Yan and John Smith, for their collaboration with us, which has led to numerous papers and even a patent. The employees and interns at IBM have made the internship a wonderful experience.

I would also like to thank Professor Lei He and his student Zhen Cao, with whom I've recently spent time thinking deeply about both theoretical and practical problems in dynamic voltage scaling. Our discussions have lead to a great paper, and I hope to continue cooperating with them in the future.

Finally, I must thank my family, my mom and dad, and my brother Peter, for their continued love and support for me during the course of my PhD career. Through thick and thin, they have been there to give me both guidance and prayer. I would like to dedicate my thesis to my family.

VITA

1980	Born in Tallahassee, Florida, USA
2003	Bachelor of Science, University of California, Berkeley.
2004	Master of Science, Electrical Engineering, Communications, UCLA.
2005	Teaching Assistant, Electrical Engineering Dept., UCLA.
	Joined the Multimedia Communications and Systems Laboratory under Professor Mihaela van der Schaar.
2006	Teaching Assistant, Electrical Engineering Dept., UCLA.
2007	Internship at IBM Watson Research Center, Hawthorne, NY.

PUBLICATIONS

Z. Cao, B. Foo, L. He, M. van der Schaar, “Optimality and Improvement of Dynamic Voltage Scaling Algorithms for Multimedia Applications,” *Design Automation Conference (DAC),* 2008. (Nominated for best paper award)

B. Foo, Y. Andreopoulos, M. van der Schaar. “Analytical Complexity Modeling of Wavelet-based Video Coders.” *ICASSP* 2007.

B. Foo, Y. Andreopoulos, M. van der Schaar. “Analytical Rate-Distortion-Complexity Modeling of Wavelet-based Video Coders.” *IEEE Trans. on Signal Processing,* Vol. 56, No. 2, Feb. 2008.

B. Foo, D. Turaga, O. Verscheure, M. van der Schaar, L. Amini, “Configuring Trees of Classifiers in Distributed Stream Mining Systems,” *IBM Austin Center for Advanced Studies,* 2008.

B. Foo, D. Turaga, O. Verscheure, M. van der Schaar, L. Amini. “Resource Constrained Stream Mining with Classifier Tree Topologies,” *IEEE Signal Processing Letters,* May 2008.

B. Foo, M. van der Schaar. “A Queuing Theoretic Approach to Processor Power Adaptation for Video Decoding Systems.” *IEEE Trans. on Signal Processing,* Vol 56, No. 1, Jan. 2008.

B. Foo, M. van der Schaar, “Distributed optimization for real-time multimedia stream mining systems,” *SPIE Multimedia Content Access,* Jan. 2008. (Invited paper)

B. Foo, M. van der Schaar, “Joint Scheduling and Resource Allocation for Multiple Video Decoding Tasks,” *SPIE Multimedia Communications and Networking,* 2008.

B. Foo, M. van der Schaar, “Queuing-theoretic Processor Power Adaptation for Video Decoding Systems.” *International Conference on Image Processing,* 2007.

ABSTRACT OF THE DISSERTATION

Towards a Systematic Approach for Modeling and Optimizing Distributed and Dynamic Multimedia Systems

by

Brian Foo

Doctor of Philosophy in Engineering

University of California, Los Angeles, 2008

Professor Mihaela van der Schaar, Chair

Recent advances in low-power, multi-core and distributed computing technologies have opened up exciting research opportunities, as well as unique challenges, for modeling, designing, and optimizing multimedia systems and applications. First, multimedia applications are highly dynamic, with source characteristics and workloads that can change significantly within milliseconds. Hence, systems need to be able to optimally adapt their scheduling, resource allocation, and resource adaptation strategies on-the-fly to meet the multimedia applications' time-varying resource demands within the delay constraints specified by each application. Second, systems often need to support multiple concurrent multimedia applications and thus, (Pareto) efficient and fair resource management solutions for dividing processing resources among the competing applications need to be designed. Finally, some applications require distributed computing resources or processing elements, which are located across different autonomous sites. These different sites can collaborate in order to jointly process the multimedia data by exchanging information about their specific system implementations, algorithms and processing capabilities. However, exchanging this information among these autonomous entities may result in unacceptable delays or

transmission overheads. Moreover, they may even refuse to share this information due to proprietary or legal restrictions. Thus, information-decentralization can present a major obstacle for optimizing the performance of delay-sensitive multimedia applications that require coordination and cooperation between distributed, autonomous sites.

This dissertation addresses the above challenges by providing a systematic framework for modeling and optimizing multimedia systems in dynamic, resource-constrained, and informationally-distributed environments. In particular, we propose a stochastic modeling approach to capture the dynamically changing utilities and workload variations inherent in multimedia applications. This approach enables us to determine analytical solutions for optimizing the performance of applications on resource-constrained systems. Furthermore, the problem of information-decentralization can be addressed in our framework by systematically decomposing the joint multi-applications and multi-site optimization problems, and designing corresponding mechanisms for exchanging model parameters, which characterize the utilities, constraints and features of the autonomous entities. This systematic decomposition enables entities to autonomously coordinate and collaborate under informational and delay constraints. Finally, to optimize the performance of the multimedia applications or systems in these distributed environments, we deploy multi-agent learning strategies, which enable individual sites or applications to model the behaviors of its competitors or peers and, based on this, select their optimal parameters, configurations, and algorithms in an autonomous manner. Summarizing, our framework proposes a unified approach combining stochastic modeling, systematic information exchange mechanisms, and interactive learning solutions for optimizing the performance of a wide range of multimedia systems.

A unique and distinguishing feature of our approach is the extent of multimedia algorithms and systems domain specific knowledge used in developing the proposed framework for modeling, and optimizing the interacting system components and applications. This is in contrast to existing distributed optimization or game theoretic approaches, which use simplistic utility - resource functions, and often ignore the

dynamics and constraints experienced in actual multimedia systems. Instead, our developed modeling and optimization framework is directly shaped by the specific characteristics, constraints and requirements of multimedia systems. Specifically, the proposed framework provides pragmatic implementation solutions for (i) the optimization of dynamic voltage scaling algorithms for multimedia applications, (ii) energy-aware resource management for multiple multimedia tasks, and (iii) resource-constrained adaptation for cascaded classifier topologies in distributed stream mining systems.

CHAPTER 1

Introduction

1.1 Motivation

With the advent of networked multimedia applications such as Internet/web TV, YouTube, peer-to-peer multimedia streaming, videoconferencing, on-line gaming, remote surveillance and monitoring, distributed data mining etc., multiple multimedia applications need to be executed simultaneously and are required to share the available resources of various heterogeneous and distributed systems. In order to efficiently utilize such systems, the different properties of (each of) the processor(s), the power consumption, the processor usage and load balancing etc. must be taken into account.

To develop an efficient and fair solution for multiple concurrent delay-critical multimedia tasks, the design of a resource management framework, which coordinates the scheduling and allocation of resources, becomes of paramount importance. This holds for resource-constrained embedded devices as well as powerful general purpose processors and distributed systems. In a New York Times article published in [1], the development of efficient system software is cited as *the* main challenge for the successful deployment of powerful emerging processors, which have as primary drivers futuristic multimedia applications.

Existing system software tries to insulate multimedia applications from the system implementation without providing them incentives to adapt their algorithms to effectively use the available hardware resources or to consider other tasks' requirements. There is currently no systematic support for trading resources among the simultaneously running tasks in order to enforce utility-dependent fairness rules defined by the system, end user(s) or service provider. Thus, current solutions result in an inefficient resource usage, as they do not provide incentives to multimedia algorithms to adapt their realizations to operate at the optimal point in the resource-utility space. Note that for multimedia applications to efficiently utilize system resources, utilities need to be explicitly modeled by considering the multimedia content characteristics and delay constraints of the applications, which is often time-varying.

In summary, existing application-agnostic resource management solutions result in a non-scalable and inefficient system implementation for the various concurrently-running multimedia applications. Thus, a rigorous methodology for joint system resource management and strategic inter-task optimization is necessary in order to allow the successful and fair deployment of multiple delay-sensitive multimedia applications on the same system. This also needs to be complemented by strategic learning and adaptation of multimedia algorithms such that they can take advantage of the available resources (see Figure 1).

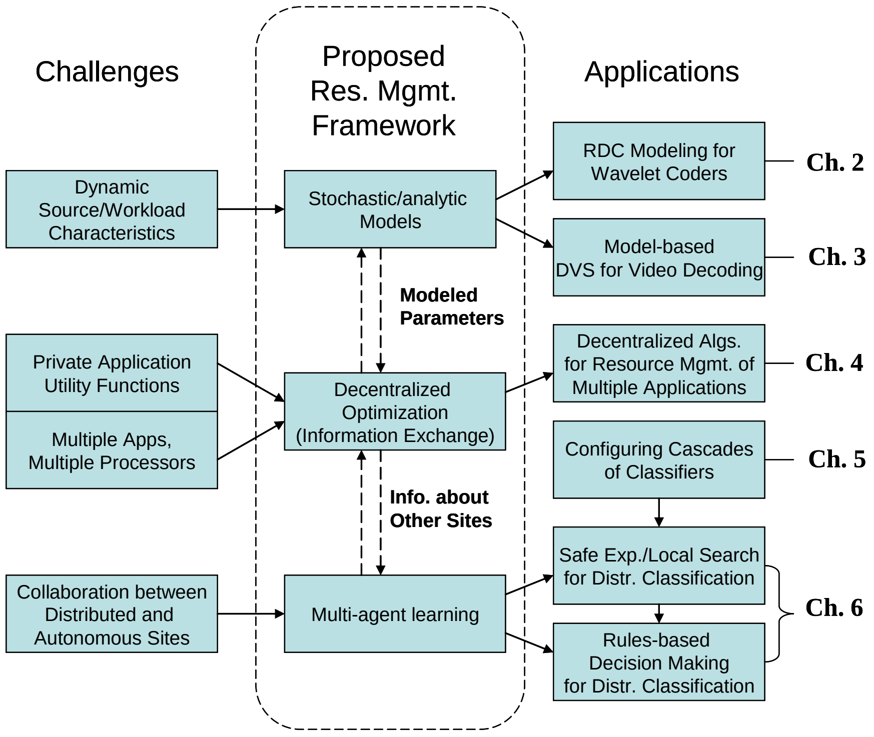

Challenges
Proposed Res. Mgmt. Framework
Applications
Dynamic Source/Workload Characteristics
Stochastic/analytic Models
RDC Modeling for Wavelet Coders
Ch. 2
Model-based DVS for Video Decoding
Ch. 3
Modeled Parameters
Private Application Utility Functions
Multiple Apps, Multiple Processors
Decentralized Optimization (Information Exchange)
Decentralized Algs. for Resource Mgmt. of Multiple Applications
Ch. 4
Configuring Cascades of Classifiers
Ch. 5
Info. about Other Sites
Collaboration between Distributed and Autonomous Sites
Multi-agent learning
Safe Exp./Local Search for Distr. Classification
Rules-based Decision Making for Distr. Classification
Ch. 6

Figure 1 Summary of the Proposed Resource Management Framework

1.2 Thesis goal

To address the above challenges, in this dissertation, we propose a modeling and optimization framework for multimedia applications in dynamic, resource-constrained, and informationally-distributed environments, and discuss practical implementations of the framework for optimizing a variety of multimedia systems (see Figure 1). In particular, we propose a stochastic modeling approach to capture the dynamically changing utilities and workload variations inherent in multimedia applications. This approach enables us to determine analytical solutions for optimizing the performance of applications on resource-constrained systems. Furthermore, the problem of information-decentralization can be addressed in our framework by systematically decomposing the joint multi-applications and multi-site optimization problems, and designing corresponding mechanisms for exchanging model parameters, which characterize the utilities, constraints and features of the autonomous entities. This systematic decomposition enables entities to autonomously coordinate and collaborate under informational and delay constraints. Finally, to optimize the performance of the multimedia applications or systems in these distributed environments, we deploy multi-agent learning strategies, which enable individual sites or applications to model the behaviors of its competitors or peers and, based on this, select their optimal parameters, configurations, and algorithms in an autonomous manner. Summarizing, our framework combines stochastic modeling, systematic information exchange mechanisms, and interactive learning solutions to optimize the performance of a wide range of multimedia systems.

A unique and distinguishing feature of our approach is the extent of multimedia algorithms and systems domain specific knowledge used in developing the proposed framework for modeling, and optimizing the interacting system components and applications. This is in contrast to existing distributed optimization or game theoretic approaches, which use simplistic utility - resource functions, and often ignore the dynamics and constraints experienced in actual multimedia systems. Instead, our developed modeling and optimization framework is directly shaped by the specific characteristics, constraints and requirements of multimedia systems. Specifically, the proposed framework provides pragmatic implementation solutions for (i) the optimization of dynamic voltage scaling algorithms for multimedia applications, (ii) energy-aware resource management for multiple multimedia tasks, and (iii) resource-constrained adaptation for cascaded classifier topologies in distributed stream mining systems.

1.3 Thesis Outline and Contributions

Our framework uses stochastic modeling to accurately predict the dynamic workload and quality fluctuations of multimedia applications. Modeling the workload enables us to make probabilistic guarantees on meeting stringent application delay deadlines, and to reallocate/reconfigure system resources accordingly to guarantee high application performance. Some practical implementations of our modeling framework include bitstream shaping using rate-distortion-complexity tradeoff models (Chapter 2), and proactive dynamic voltage scaling (DVS) algorithms for real-time multimedia tasks (Chapter 3). Secondly, modeling resource demands and utilities for multiple applications enables us to exchange parameters between applications to efficiently communicate information (Chapter 4). This low-complexity scheme of exchanging modeled parameters enables the system to implement decentralized resource management solutions that converge quickly to efficient and fair centralized resource allocation solutions. Next, we introduce a solution for configuring trees of classifiers on distributed stream mining systems to improve classification performance (Chapter 5). We then extend this problem to configuring cascades of classifiers, where each classifier may be located across different administrative domains (Chapter 6). The extension requires using our proposed framework to address the challenge of distributed resources and analytics by multi-agent modeling, where autonomous sites can utilize both locally observable and exchanged information to model the behaviors of other sites, and *learn* the dynamic characteristics of the processed stream. This approach enables distributed sites to reconfigure their algorithms to jointly optimize the performance of an application, even when very little information can be exchanged between the sites.

In **Chapter 2**, we present a novel *rate-distortion-complexity* (R-D-C) analysis for state-of-the-art wavelet video coding methods based on stochastic source models. This analysis is obtained by explicitly modeling aspects found in a broad class of operational wavelet video coders, i.e. embedded quantization, quadtree decompositions of block significance maps and context-adaptive entropy coding of subband blocks. Importantly, the proposed modeling approach derives for the first time analytical estimates of the expected number of operations (complexity) of a broad class of wavelet video coding algorithms based on stochastic source models, the coding algorithm characteristics and the system parameters. Analytical modeling of the performance of video coders is essential in a variety of applications, such as complexity-driven bitstream shaping [11], where an accurate estimation of rate, distortion and complexity is required. The accuracy of the proposed analytical R-D-C expressions is justified against experimental data obtained with a state-of-the-art motion-compensated temporal filtering based wavelet video coder, and several new insights are revealed on the different tradeoffs between rate-distortion performance and the required decoding complexity.

In **Chapter 3**, we present novel approaches to *DVS algorithms* for multimedia applications that relies on building stochastic models for video decoding complexity. Previous works on video-related voltage scaling algorithms often lacked a good complexity model of multimedia applications, and thus could not achieve energy-efficient performance on battery-limited devices. Our contribution in these sections are threefold. First, based on workload traces, we determine using an offline linear programming (LP) method the minimum (optimal) energy consumption for processing multimedia tasks under stringent delay deadlines. This lower bound enables us to evaluate the efficiency of various existing DVS algorithms. Second, we propose a classification-based complexity model that explicitly considers the video source characteristics, the encoding algorithm, and platform specifics to predict job execution times. In particular, separate models are presented for different types of decoding jobs (e.g. entropy decoding, inverse transform, motion compensation) and different classes of sequences. Based on the models, we construct two online voltage scaling algorithms to process decoding jobs such that they meet their display deadlines with high probability. The first DVS algorithm is based on a robust linear programming (rLP) approach, adapted from the offline LP solution. Simulation results from decoding over a wide range of video sequences shows that on average, both the queuing-based DVS algorithm and the rLP DVS algorithm perform close to optimal, with the queuing-based algorithm consuming roughly 4% more energy than optimal, and the robust LP algorithm 1% more than optimal. This is contrasted with other state-of-the-art DVS algorithms which consume roughly 16% or more energy under the same miss rates. Finally, we propose a joint voltage scaling and quality-aware priority scheduling algorithm that decodes jobs in order of their distortion impact, such that by setting the processor to various power levels and decoding only the jobs that contribute most to the overall quality, efficient quality and energy tradeoffs can be achieved. We show experimentally that under heavy resource constraints, the priority-based scheduling algorithm achieves a much higher quality than conventional earliest-deadline-first (EDF) scheduling algorithms.

In **Chapter 4**, we discuss *fair and efficient resource allocation schemes* for multiple multimedia applications sharing a system. Because multimedia tasks can achieve different utilities (e.g. video qualities) under different amounts of system resources (e.g. Chapter 2), various algorithms have been proposed to determine resource allocation schemes that maximize the social welfare of all applications. However, these approaches often require a resource manager that can obtain a centralized model of the utilities and resource demands for each application, which can lead to intolerable computational complexity and information exchange overhead. Moreover, autonomous multimedia applications owned by different companies may have incentives to hide information about their algorithms and utilities from the system, or from other competing applications. Hence, to optimally allocate resources in an informationally-decentralized environment, we present a novel, low-complexity resource management solution that does not require the resource manager to know the applications' utility functions.

Instead, by allowing applications to negotiate with a resource manager through simple control messages, we implement informationally-decentralized algorithms that optimally achieve a variety of centralized resource allocation solutions, such as maximizing the social welfare of the applications, minimizing system energy consumption, manipulating the workload to achieve a desired distribution of tasks on processing elements, and performing power scheduling for multimedia tasks. Our analysis and simulation results reveal that these algorithms converge quickly to their respective optimal solutions. Furthermore, we show that by modeling changes in the applications' resource requirements, the resource manager can communicate messages to applications and guarantee near optimal resource allocation in dynamic environments.

In **Chapter 5**, we discuss a novel methodology for *configuring cascaded classifier topologies*, specifically binary classifier trees, in resource-constrained, distributed multimedia stream mining systems. Binary classifier trees enable stream mining applications to successively identify and dynamically filter different attributes in data content, and have been shown to outperform single classifier systems. Inherent in such a topology is the need to model the (average) performance of the classifier system using an appropriate cost/utility metric, and to model the relationships of analytics between the classifiers. In contrast with traditional load shedding, our approach configures classifiers with optimized operating points after jointly considering the misclassification cost of each end-to-end class of interest in the tree, the resource constraints for every classifier, and the confidence level of each data object that is classified. The proposed approach allows for both intelligent load shedding as well as data replication across branches of the tree based on the available system resources. Both a centralized solution, and distributed solutions that enable each classifier in the tree to reconfigure itself based on local information exchanges, are provided. We evaluate the centralized and distributed algorithms based on a sports video concept detection application and identify huge cost savings over load shedding alone. We also analyze the associated tradeoffs between convergence time, information overhead, and the performance of results achieved by each of the proposed distributed algorithms.

In **Chapter 6**, we introduce an additional challenge involved in optimizing *a cascade of classifiers for real-time, distributed stream mining applications*; in particular, when the classifiers themselves are placed across sites located in *different administrative or autonomous domains*. Such sites may be unwilling to share their proprietary datasets or analytics, nor capable of providing a repository to store an entire collection of data across multiple sites. As opposed to the problem introduced in Chapter 6, where the performance of an entire classifier system could be analytically modeled based on known relationships between classifiers, the distributed nature of analytics (e.g. utilized models or stored data sets) severely limits the amount of information available for jointly optimizing, much less modeling, the performance of a cascade of distributed classifiers. To address this problem, we introduce three novel approaches: 1) We formulate an average utility metric based on classification and queuing theoretic models to capture both the performance and delay of a chain of classifiers. 2) We introduce a low-complexity framework for estimating system utility in a distributed fashion, where local performance metrics are exchanged between classifiers to obtain an estimate of the overall utility at any time instant. Additionally, this message exchange mechanism has very low communications overhead, and does not require classifiers to reveal sensitive information about their analytics. 3) We introduce a distributed, multi-agent learning algorithm (i.e. safe experimentation and local search) to enable each classifier site to reconfigure itself autonomously, such that in spite of the limited information exchanged, each classifier eventually converges to a configuration that maximizes the overall stream processing application utility under fixed stream characteristics. We perform experiments to measure the value of exchanging limited information between classifiers by comparing the utility of the learning solution to a solution without information exchange, i.e. relying only on locally and independently optimizing each classifier. Furthermore, we analyze performance and convergence rate tradeoffs between different learning rates for the multi-agent solution.

Additionally, we also introduce a rules-based decision-making framework, which uses a more sophisticated learning solution for adaptively choosing algorithms to reconfigure classifiers in dynamic environments (i.e. when stream characteristics are rapidly changing). In particular, while the experimentation solution enables classifiers to converge to the optimal configurations for static streams, streams with time-varying characteristics necessitate frequent reconfiguration of classifier elements to ensure acceptable end-to-end performance and delay under resource constraints. In such dynamic environments, utilizing a fixed algorithm for classifier reconfiguration can often lead to poor performance. Hence, we propose a novel optimization framework aimed at developing *rules* for choosing among multiple algorithms, the best algorithm to reconfigure the classifier system under different system conditions. This framework involves an adaptive, Markov model-based solution for learning the optimal rule when stream dynamics are initially unknown. Furthermore, we also discuss how rules can be decomposed across multiple sites, such that each site can individually learn near optimal rules based on local models. Finally, we propose a method for evolving new rules from a set of existing rules. Simulation results for both Chapters 7 and 8 are presented for a speech classification system for both static and dynamic streams.

The thesis concludes with **Chapter 7**, which presents the design "philosophy" (principles and methods) discovered during the duration of this thesis for developing systematic resource management solutions for multimedia applications. This chapter also discusses the possible impact of the techniques presented in this thesis.

CHAPTER 2
Analytical Rate-Distortion-Complexity Modeling for Wavelet Video Coders

2.1 Introduction

Energy consumption is an important issue in mobile devices. In the case of multimedia, the battery life of such devices has been shown to be directly linked to the complexity of coding algorithms [4] [5] [9]. For this reason, recent advances in scalable coding algorithms that provide schemes enabling a variety of rate-distortion-complexity (R-D-C) tradeoffs with state-of-the-art performance [3] are very appealing frameworks for such resource constrained systems. This flexibility in video encoding and decoding is also very suitable for the increasing diversity of multimedia implementation platforms based on embedded systems or processors that can provide significant tradeoffs between video coding quality and energy consumption [4] [5]. In order to select the optimal operational point for a multimedia application in a particular system, accurate modeling of the source, algorithm *and system (implementation) characteristics* is required. Such modeling approaches are important because they can also serve as the driving mechanism behind the design of future complexity-scalable coders.

Two methods have been used to determine the rate-distortion and the complexity characteristics of operational video coders. The first is an empirical approach, where analytical formulations are fitted to experimental data to derive an operational model suitable for a particular class of video sequences and a particular instantiation of a compression algorithm in a fixed implementation architecture; see [18] [19] for such examples of R-D models and [4]-[8] for such examples of complexity modeling. While this modeling approach is simple, the obtained rate-distortion-complexity (R-D-C) expressions cannot be generalized because their dependency on the sequence, algorithm and system parameters is not explicitly expressed via the model. As a result, while current state-of-the-art multimedia compression algorithms and standards provide profiles for rate control [1] [5] [7], they lack analytical methods to determine the complexity tradeoffs between different coding operations that can be exploited for different systems.

The second approach is a theoretical approach, where stochastic models are used for pixels or transform coefficients. Using this approach, analytical expressions can be derived for the R-D-C behavior of a particular system or class of systems processing a broad category of input sources in function of the sources' statistics; see [10] [13] [14] for such examples of R-D models and [11] [12] [15] for complexity modeling using operational source statistics and off-line or on-line training to estimate (learn) the algorithm and system parameters. We remark that, although there is a significant volume of work in modeling of transform-domain statistics [27] [28] [29] and also in the efficiency analysis of coding mechanisms [18], there is significantly less literature on rate-distortion modeling for state-of-the-art operational video coders, and (to the best of the authors' knowledge) scarcely any work exists on *complexity modeling* for such systems in function of *stochastic source models* and *algorithm characteristics*. We emphasize that, while the derived theoretical expressions of such approaches are typically more complex than the expressions derived from the first category, the dependencies on the source and system modeling parameters are explicitly indicated via the derived analytical framework. This is of great importance to several cross-layer or resource optimization problems [4] [9] [11] that need to judicially balance the network or system resources in order to accommodate the viewer preferences in the most efficient manner. In addition, the explicit dependency of the derived R-D-C estimation on source, algorithm and system parameters facilitates the application of the derived framework for a variety of input video source classes. Moreover, various algorithms and systems of interest are accommodated in this way. Finally, a rigorous, analytical R-D-C formulation methodology can be easily extended to model properties of various other coding algorithms based on input source and algorithm parameters. In this way, analytical comparisons of the R-D-C efficiency for particular algorithms can accompany experimental testing in order to facilitate system design decisions and options.

For these reasons, we follow the second category of approaches and provide a unified R-D-C modeling framework for motion-compensated temporal filtering (MCTF) based wavelet video coders [3] [23] [26] [30]. Two aspects are typically required for unified R-D-C modeling of video coding: *i*) modeling of the temporal prediction process [16] [17]; *ii*) modeling of the quantization and coding process [10] [13]. Since the motion-compensation complexity of the MCTF process has been studied in detail in prior work [8] [12], we focus on the second part and assume that the transform-domain statistics of the intra and error frames produced by the temporal decomposition are available. Unlike the existing theoretical work [10] [13] in this area, the proposed R-D-C model is based on a thorough analysis of *different coding operations* (quadtree coding, coefficient significance and refinement coding) and as a result can encompass many state-of-the-art wavelet video coders found in the literature.

Consequently, this chapter extends prior R-D modeling of block-based wavelet video coders to a broader class of coding mechanisms. Perhaps more importantly, we propose an analytical derivation of complexity estimates for the entropy decoding and the inverse spatial transform, thereby complementing some of our prior work on complexity estimation [8] [11] [12].

Based on the derived theoretical results and their experimental validation, we explore the R-D-C space of achievable operational points for state-of-the-art wavelet video coders and derive several interesting properties for the interrelation of rate-distortion performance and the associated decoding and inverse spatial transform complexity.

This chapter is organized as follows. Section 2.2 introduces the types of quantization and coding schemes analyzed in this chapter. Some important nomenclature is also provided. Section 2.3 presents the utilized wavelet coefficient models and derived probability estimates for a variety of coding/decoding operations. These probabililities will be used to determine the average rate, distortion and complexity (Sections 2.4-2.6, respectively) for decoding a video sequence. Section 2.7 displays theoretical and experimental R-D-C results that validate the proposed models and discusses several interesting R-D-C properties of operational video coders. Section 2.8 draws some conclusions based on what was discussed in this chapter.

2.2 Overview of Wavelet Video Coders

In this section, we introduce a basic overview of state-of-the-art wavelet coding schemes. They involve temporal decomposition via MCTF, spatial discrete wavelet transform (DWT) decomposition, embedded quantization, and the entropy coding process.

2.2.1 Temporal Decomposition

Recent state-of-the-art scalable video coding schemes are based on motion compensated temporal filtering [3]. During MCTF, the original video frames are filtered temporally in the direction of motion [3] [16], prior to performing the spatial transformation and coding. Video frames are filtered into L (low-frequency or average) and H (high-frequency or error) frames [3]. The process is applied initially in a group of pictures (GOP) and also to all the subsequently-produced L frames thereby forming a total of T_{MCTF} temporal levels. After the temporal decomposition, the derived L and H temporal frames are spatially decomposed in a hierarchy of *spatio-temporal subbands*. Quantization and entropy coding are applied to these subbands to form the final compressed bitstream.

2.2.2 Embedded Quantization

An important category of quantizers used in image and video coding is the family of embedded double-deadzone scalar quantizers [20]. For this family, each input wavelet coefficient x is quantized to:

$$Q_b(x) = \left\{\mathrm{sign}(x) \cdot \left\lfloor \tfrac{|x|}{2^b \Delta} \right\rfloor, \text{ if } \frac{|x|}{2^b \Delta} \geq 1;\ 0, \text{otherwise}\right\}, \tag{1}$$

where $\lfloor a \rfloor$ denotes the integer part of a; $\Delta > 0$ is the basic quantization step size (basic partition interval size) of the quantizer family; $b \in \mathbb{Z}_+$ indicates the quantizer level (granularity), with higher values of b indicating coarser quantizers. In general, b is upper bounded by a value $B_{\max}$, selected to cover the dynamic range of the input signal. The signal reconstruction is performed by:

$$Q_b^{-1}(Q_b(x)) = \left\{ \operatorname{sign}(Q_b(x)) \cdot \left(|Q_b(x)| + \tfrac{1}{2}\right) 2^b \Delta, \text{ if } Q_b(x) \neq 0;\ 0 \text{ if } Q_b(x) = 0 \right\} \tag{2}$$

where the reconstructed value $Q_b^{-1}(Q_b(x))$ is placed in the middle of the corresponding uncertainty interval (partition cell), and $Q_b(x)$ is the partition cell index, which is bounded by a predefined value for each quantizer level. For example, $0 \leq Q_b(x) \leq M_b - 1$, for each b, with $M_{B_{\max}} = \ldots = M_0 = 2$ and $\Delta = 1$ for the popular case of successive approximation quantization (SAQ) [20]. If the b least-significant bits of $Q_0(x)$ are not available, one can still dequantize at a lower level of quality using the inverse quantization formula given in (2). SAQ can be implemented via thresholding, by applying a monotonically decreasing set of thresholds of the form $T_{b-1} = T_b / 2$, with $B_{\max} \geq b \geq 1$ and $T_{B_{\max}} = \alpha_{\text{quant}} \cdot x_{\max}$, where $x_{\max}$ is the highest coefficient magnitude in the input wavelet decomposition, and α_{quant} is a constant that is taken as $\alpha_{\text{quant}} > 1/2$. By using SAQ, the significance of the wavelet coefficients with respect to any threshold T_b is indicated in a corresponding binary map, called the significance map. Coding of $B_{\max}, \ldots, B_{\min}$ significance maps corresponds to coding the $B_{\max} - B_{\min} + 1$ most significant bitplanes of each wavelet coefficient x.

2.2.3 Coding of the Significance Maps and Coefficients

In all state-of-the-art wavelet coders [20]-[26], the coding process exploits intra-band dependencies following a block-partitioning process within each transform subband. This coding process is performed for every bitplane b. As indicated in Figure 2, several coding passes that identify coefficient significance (“Significance Pass”) or refine wavelet coefficients (“Refinement Pass”) with respect to the current SAQ threshold are performed either within quadtree coding [23] [24] or within block coding [20]. Several state-of-the-art embedded image coders invoke both approaches, i.e. the quadtree coding partitions the input subbands until a minimum block size, which is then coded with the block coding module [21] [22].

We analyze such intra-band coders that use quadtrees to decompose subbands into non-overlapping blocks of dyadically-decreasing sizes followed by block coding for the blocks of the maximally decomposed quadtree [21] [22]. In particular, the initial subbands are hierarchically split in K quadtree levels using several coding passes, with blocks at quadtree level K having the smallest size. The significance information (i.e. whether the block contains significant coefficients) is encoded using depth-first-search along the quadtree, where the significance of a block at quadtree level k is encoded only if its parent block at quadtree level $k-1$ is found to be significant. For the blocks found significant at the bottom of the quadtree (level K), the block coding is invoked. Block coding performs raster scan to obtain the significance of each coefficient. The coefficients found significant are then placed in a refinement list to be refined at the next finer quantization level.

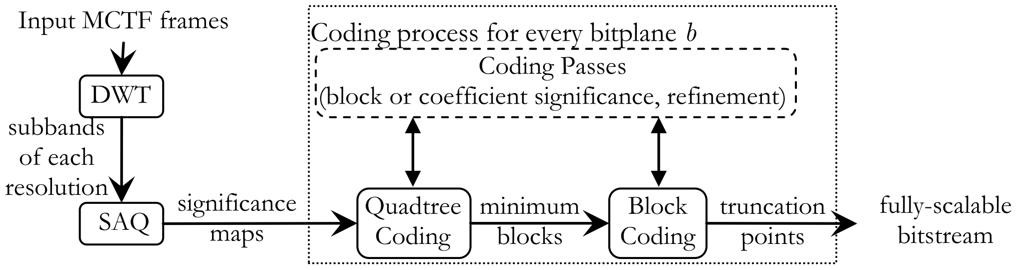


Figure 2 Block diagram of intra-band coding process of state-of-the-art wavelet-based coders encompassing quadtree and block coding of the significance maps.

The produced symbols from each coding pass, from block significance information to coefficient significance, refinement, and sign information, are then encoded using context-based adaptive arithmetic coding [34] [35]. This technique exploits the dependencies between the symbols to be encoded and the neighboring symbols (the context) [34]. Context conditioning reduces the entropy and improves the coding performance. An example of context-based entropy coding is to use several arithmetic coder models with different initial probabilities to encode coefficients based on the significance of their neighbors, since a coefficient with significant neighboring coefficients has a larger probability to be significant than coefficients with insignificant neighboring coefficients. Using these separate arithmetic coder models for different “contexts,” context-based coding schemes achieve better performance than simply compressing all symbols using a single arithmetic coder [34] [35].

2.3 Estimating Block Significance Probabilities in Quadtree Decompositions

In this section, we introduce the utilized stochastic source model for wavelet coefficients. We then derive probabilities of significance for quadtree decompositions over quantized spatio-temporal subbands. These probabilities form the core of the rate and complexity estimation derived in the remaining sections of this chapter as they provide the means of establishing the percentage of blocks that are expected to be coded or decoded at a given distortion bound, expressed by the terminating SAQ threshold $T_{B_{\min}}$. In addition, the percentage of significant areas within the spatio-temporal subbands along with the percentage of non-zero coefficients are the two features (or “decomposition functions” [12]) that express the complexity of the inverse DWT.

2.3.1 Source Models for Wavelet Coefficients

The R-D characteristics of low-frequency wavelet coefficients are typically modeled using the high-rate uniform quantization assumption [10] [20] for independent zero-mean Gaussian random variables. This model will be accurate if the low-frequency coefficients exhibit sufficiently low correlation. We investigate this in Table 1, which displays the ratio of the average correlation between neighboring coefficients to the average coefficient variance. In Figure 3 we validated that the Gaussian distribution for low-frequency spatio-temporal subband coefficients was accurate.

While low-frequency spatio-temporal subbands account for a large percentage of the video coding rate, the high-frequency spatio-temporal subbands also contribute a significant amount to the overall coding rate and complexity [14]. Thus, accurate modeling of the high-frequency spatio-temporal subband statistics is also very important for precise R-D-C modeling of wavelet video coders. When applied to image or residual frame data, typical wavelet filters tend to produce decorrelated coefficients in the high-frequency subbands. However, dependencies remain among coefficients within the same scale and across different scales [27]. Certain highly-popular wavelet filter-banks, such as the Daubechies 9/7 filter-pair, have further properties that can reduce most of the interscale dependencies, leaving only dependencies among neighboring coefficients within the same subband [27].

Table 1 Ratio of correlation between neighboring coefficients to the average coefficient variance for the LL subband of L-frames of *Foreman*, *Coastguard*, *Silent*, and *Mobile* after a 2 temporal level-4 spatial level decomposition.

	Foreman	*Coastguard*	*Silent*	*Mobile*
Autocorrelation Coefficient	0.5340	0.6733	0.5100	0.3763

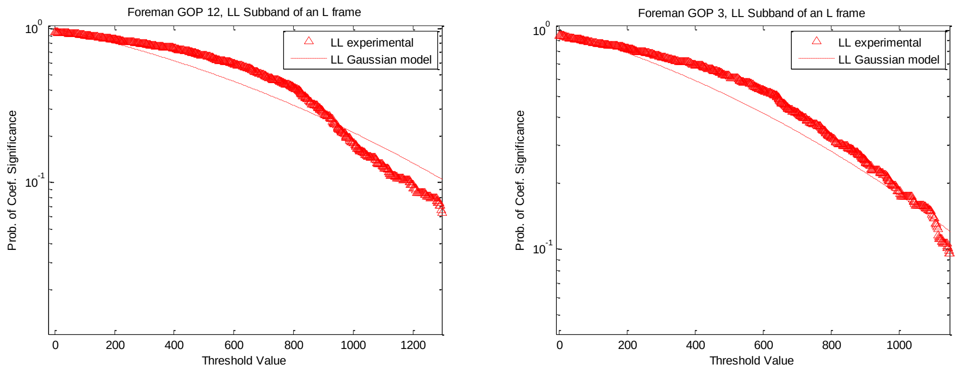


Figure 3 Experimental examples that the Gaussian assumption for the low-frequency wavelet coefficients of the MCTF-based decomposition (with four spatio-temporal levels) is accurate.

In order to capture the experimentally-observed heavy-tailed non-Gaussian distribution of wavelet coefficients within each subband, we model high-frequency wavelet coefficients as a doubly-stochastic process, i.e. a Gaussian distribution parameterized by Θ, which is exponentially distributed with parameter σ^2:

$$\Theta \sim p(\theta) = \frac{1}{\sigma^2} e^{-\frac{1}{\sigma^2}\theta} \tag{3}$$

In this case, each high-frequency wavelet coefficient x can be modeled by a random variable X with marginally Laplacian distribution and variance σ^2 [37]:

$$X \sim p(x) = \int_0^\infty p(x \mid \theta) p(\theta) d\theta = \int_0^\infty \frac{1}{\sqrt{2\pi\theta}} e^{-\frac{1}{2\theta}x^2} \frac{1}{\sigma^2} e^{-\frac{1}{\sigma^2}\theta} d\theta = \frac{1}{\sqrt{2}\sigma} e^{-\frac{\sqrt{2}}{\sigma}|x|} \tag{4}$$

where $p(x)$ indicates the probability density function (PDF). Figure 4 demonstrates the accuracy of the doubly-stochastic model of (4) for different spatio-temporal high-frequency subbands. In addition, Table 2 presents the change in the subband statistics for different spatio-temporal levels across the MCTF decomposition and the corresponding rate for terminating the coding of the wavelet coefficients of each spatio-temporal level at several bitplanes. The coder of [30] was used for the examples of this section. While we focused on 4x4 blocks in **Table** 2, similar results can be shown for larger block sizes (e.g. 8x8), since, for natural images or error frames, wavelet coefficients within such small areas of high-frequency subbands are modeled accurately with the same local parameter θ.

Based on the results of Table 2 we conclude that there is significant variation in the rate associated with each spatio-temporal level, ranging from 0 bpp to almost 1 bpp for low-rate coding ($B_{\min} = 7$ in Table 2) and from 0.15 bpp to almost 5 bpp for medium and high rate coding ($B_{\min} = 3$ in Table 2). Furthermore, the higher (coarser) spatio-temporal high-frequency subbands exhibit significant variance and the correlation of the subband statistics (parameter θ) varies significantly as well. Consequently, there is a significant portion of the coding rate attributed to them for a variety of quantization thresholds; thus, accurate modeling of the rate-distortion-complexity characteristics of high-frequency spatio-temporal subbands is important for predicting the overall R-D-C behavior. Finally, although the results of Table 2 reveal certain trends between the spatio-temporal subband rate and the model parameters (σ^2 and θ), the overall rate contribution of each subband depends not only on the statistics of the input but also on the details of the invoked coding algorithm. Hence, a detailed theoretical analysis of the coding operations as a function of the source model is of paramount importance for precise R-D-C estimations, and intuitive models based on the source statistics and experimental observations do not suffice.

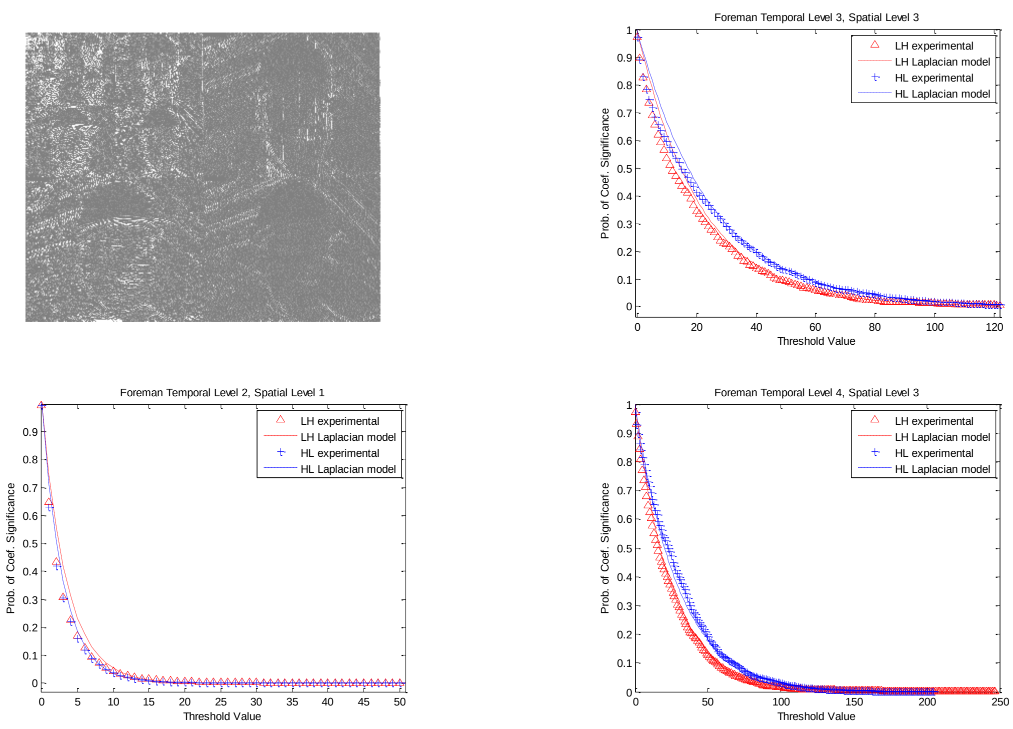


Figure 4 The discrete wavelet transform of an H frame of temporal level two (top left) and plots of the doubly-stochastic (Laplacian) model and simulation data for several H frames of different spatio-temporal resolutions.

Table 2 Examples of subband variances as well as the variance of the correlation θ (for block sizes of 4×4) formed across the spatio-temporal MCTF subbands of sequence Foreman, along with the corresponding bitrates for several values of $B_{\min}$.

Temporal(T)-Spatial(S) level	Subband variance σ^2, [variance of θ]			Rate for various decoded bitplanes $B_{\min}$				
	LH	HL	HH, LL (if exists)	7	6	5	4	3
1T-1S	3.18, [12.1]	1.87, [9.1]	1.61. [7.8]	0	0	0.002	0.026	0.149
2T-2S	6.59, [37.5]	5.55, [22.8]	4.46, [25.7]	0.002	0.022	0.122	0.391	1.026
3T-3S	18.2, [60.3]	14.8, [70.1]	11.7, [63.1]	0.118	0.421	0.963	1.707	2.712
4T-4S	39.7, [241]	33.2, [496]	{26.0,[144]}, {53.1, 690]}	0.970	1.732	2.672	3.793	4.934

We denote the minimum decoded bitplane threshold level as $T_{B_{\min}} = 2^{B_{\min}}$. In addition, we define the following parameters for all bitplanes b:

$$\upsilon_b = \frac{T_b}{\sigma} \tag{5}$$

$$\rho_b = e^{-\sqrt{2}\upsilon_b} \tag{6}$$

where υ_b describes the ratio of the threshold of bitplane b to the variance of each wavelet coefficient, and ρ_b is the probability of significance of a wavelet coefficient under a certain υ_b under the model of (4).

2.3.2 Probability of Block Significance at Bitplane b

We begin this section by introducing some notation. We define the significance test of a block of n coefficients with respect to a threshold T_b as $\mathrm{sig}(T_b, n) \in \{0,1\}$, where $\mathrm{sig}(T_b, n) = 1$ if at least one coefficient within the block is found significant with respect to the threshold T_b, and $\mathrm{sig}(T_b, n) = 0$ otherwise. We also define the newly significance test as $\mathrm{newsig}(T_b, n) = \{0,1\}$, which returns one if the block was found to be significant at bitplane b and insignificant at bitplane $b+1$, i.e. $\mathrm{sig}(T_b, n) = 1$ and $\mathrm{sig}(T_{b+1}, n) = 0$. For notational abbreviation, the *probability* of a block being significant or newly-significant at bitplane b is indicated by $\chi_{v_b,n}^{\mathrm{band}}$ and $\delta_{v_b,n}^{\mathrm{band}}$, respectively, with $\mathrm{band} = \{\mathrm{low,high}\}$ indicating the frequency subband that the block belongs to. Note that these metrics depend on v_b, which is a function of the bitplane b as well as the variance of subband coefficients, σ^2. Let us first consider a high-frequency spatio-temporal subband, which may be any subband of an error (H) frame, or any high-frequency subband of an L frame. The probability that a block of n wavelet coefficients is found significant during (or before) the significance pass at bitplane b, $\chi_{v_b,n}^{\mathrm{high}}$, is no more than the probability that at least one coefficient in the block is found significant when compared to $T_b = 2^b$. Thus, we have:

$$\chi_{v_b,n}^{\mathrm{high}} = \Pr\{\mathrm{sig}(T_b, n) = 1\} = 1 - \Pr\{|\,\mathbf{X}\,|_\infty \le T_b\} \tag{7}$$

where $\mathbf{X} = (X_1, ..., X_n)$ is a length-n random vector of variables X_i $(1 \le i \le n)$ for coefficients in the same block, and $|\bullet|_\infty$ is the $L^{\inf}$ norm. Considering that block sizes are generally small enough to capture local variances, we follow the doubly stochastic model in equation (3). Given Θ, the conditional joint distribution of $\mathbf{X}$ is then a uncorrelated Gaussian random vector. Given that the variance of the subband coefficients is σ^2, the probability density function of $\mathbf{X}$ is:

$$p(\mathbf{x}) = \int_{0+}^{\infty} p(\theta)p(\mathbf{x} \mid \theta)d\theta = \int_{0+}^{\infty} \frac{1}{\sigma^2} e^{-\frac{1}{\sigma^2}\theta} \cdot \frac{1}{(2\pi\theta)^{n/2}} e^{-\frac{1}{2\theta}(x_1^2 + x_2^2 + ... + x_n^2)} d\theta \tag{8}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)$ is a vector of coefficient values, and $p(\mathbf{x})$ is the n-dimensional PDF of $\mathbf{X}$.

Proposition 1: The probability that a block of size n is significant compared to threshold T_b can be approximated by:

$$\chi_{\upsilon_b,n}^{\text{high}} \cong \exp\left\{\frac{\upsilon_b^2}{\ln(n)^{1.296}+0.166}\right\} \tag{9}$$

*Proof***:** See Appendix A of [62] for details. ■

For low-frequency subbands, assuming sufficiently-decorrelated Gaussian distributed coefficients, the probability of block significance is simply the n-dimensional Gaussian tail probability along one of the orthogonal axes:

$$\chi_{\upsilon_b,n}^{\text{low}} = \left[\operatorname{erfc}\left(\frac{\upsilon_b}{\sqrt{2}}\right)\right]^n \tag{10}$$

where $\operatorname{erfc}\left(\frac{\upsilon_b}{\sqrt{2}}\right) = 1 - \operatorname{erf}\left(\frac{\upsilon_b}{\sqrt{2}}\right)$, and $\operatorname{erf}\left(\frac{\upsilon_b}{\sqrt{2}}\right)$ can be piecewise approximated by (11):

$$\operatorname{erf}\left(\frac{\upsilon_b}{\sqrt{2}}\right) \approx \begin{cases} 0.2\upsilon_b(4.4-\upsilon_b) & ,0 \le \upsilon_b \le 2.2 \\ .98 & ,2.2 < \upsilon_b < 2.6 \\ 1 & ,\upsilon_b \ge 2.6 \end{cases} \tag{11}$$

in order to avoid numerical integrations during the model calculation.

2.3.3 Probability of Finding a Newly Significant Block at Bitplane b

In order to model the number of operations performed during the significance pass at each bitplane, it is necessary to derive the probability that a block is found significant at bitplane b, but not at any higher bitplanes. This is due to the fact that in all coding algorithms using quadtrees of wavelet coefficients, once a block is found significant at bitplane b, it is moved into the refinement list and its significance is not encoded at the subsequent bitplanes $b-1,\dots,B_{\min}$.

Proposition 2: The probability that a block of n coefficients in a high-frequency subband is found significant at bitplane b, but it is insignificant at bitplane $b+1,\dots,B_{\max}$, is:

$$\delta_{\upsilon_b,n}^{\text{high}} = \Pr\{\text{newsig}(T_b,n)=1\} \cong \chi_{\upsilon_b,n}^{\text{high}}(1-\chi_{\upsilon_b,n}^{\text{high}}) \tag{12}$$

*Proof***:** See Appendix A of [62] for details.

■

We note that our proof only specifies the existence of a large enough n for the approximation above, but it does not give the exact lower bound for n. To verify the accuracy of this estimate for typical blocks during the quadtree significance passes, we did a qualitative comparison of plotted curves for $\mathrm{erf}(x)/\mathrm{erf}(2x)$ raised to various powers. For the minimum block size $n = 16$ used in practical coders [21] [25] [26] the match is approximately equal (Figure 5). The fit only improves for a larger n.

For low-frequency subbands, we will assume decorrelated coefficients. Our result is given below.

Proposition 3: The probability that a block of n coefficients in a low-frequency subband is found significant at bitplane b, but it is insignificant at bitplane $b+1,\ldots,B_{\max}$ is:

$$\begin{aligned}\delta^{\text{low}}_{\upsilon_b,n} &= \left[\mathrm{erfc}\left(\frac{\upsilon_b}{\sqrt{2}}\right)\right]^n - \left[\mathrm{erfc}\left(\sqrt{2}\upsilon_b\right)\right]^n = \left[\mathrm{erf}\left(\sqrt{2}\upsilon_b\right)\right]^n - \left[\mathrm{erf}\left(\frac{\upsilon_b}{\sqrt{2}}\right)\right]^n \\ &\cong \left[\mathrm{erf}\left(\sqrt{2}\upsilon_b\right)\right]^n \left[1 - \left[\mathrm{erfc}\left(\frac{\upsilon_b}{\sqrt{2}}\right)\right]^n\right] = (1-\chi^{\text{low}}_{\upsilon_{b+1},n})\chi^{\text{low}}_{\upsilon_b,n}\end{aligned} \tag{13}$$

Proof: The approximation of (13) is a straightforward result of Lemma 2 in [62]. ■

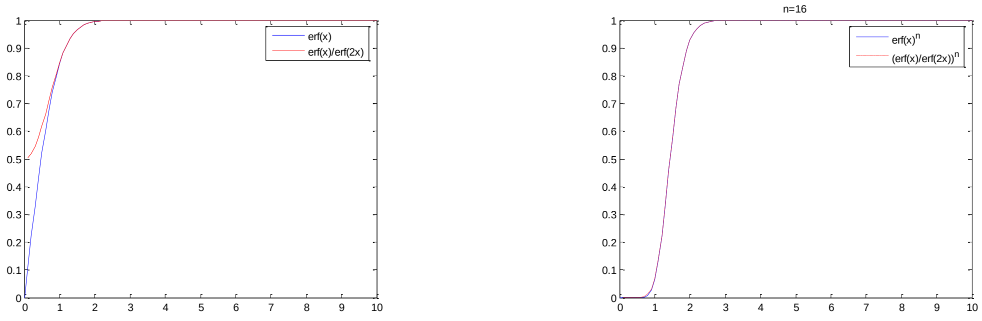


Figure 5 Plot of: $\mathrm{erf}(x)$ **vs.** $\mathrm{erf}(x)/\mathrm{erf}(2x)$ **(left), and** $[\mathrm{erf}(x)]^{16}$ **vs.** $[\mathrm{erf}(x)/\mathrm{erf}(2x)]^{16}$ **(right).**

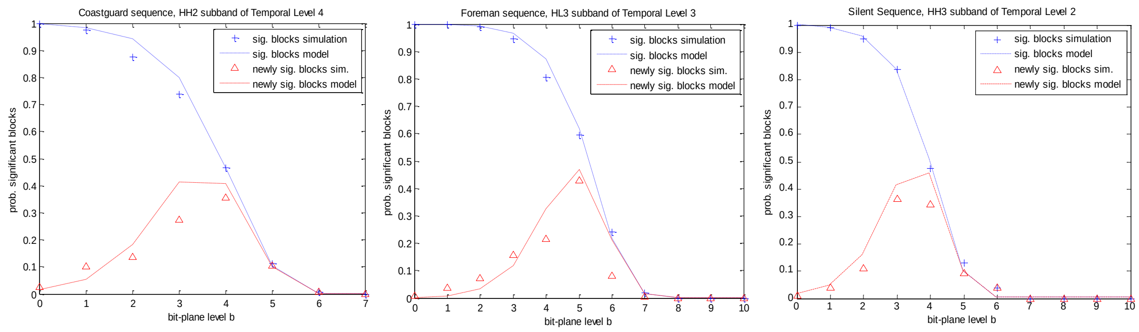


Figure 6 Simulation and model prediction of significance and newly-significance of 4×4 blocks in various high-frequency spatio-temporal subbands.

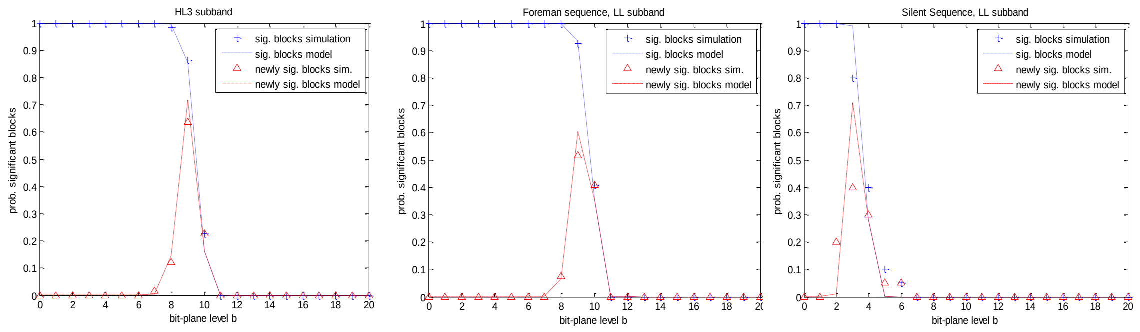


Figure 7 Simulation and model prediction of significance and newly-significance of 4×4 blocks in LL subbands of L frames.

To verify the accuracy of the proposed model of (9) and (12), Figure 6 demonstrates the model prediction of the probability of block significance and newly-significance (with $n = 16$) for several high-frequency subbands belonging to various temporal levels in MCTF-decomposed frames of video sequences. Similarly, we plot several examples that validate the proposed model of (10) and (13) (low-frequency spatio-temporal subbands) in Figure 7. The experimentally-derived significance and newly-significance for low and high-frequency spatio-temporal subbands are in agreement with the theoretical derivations for a large variety of cases, as shown by these experiments. In addition, the model validation reveals several interesting properties. Firstly, the approximation (13) for the significance probability of blocks in the low-frequency spatio-temporal subbands suggests that most of the blocks within the subband will be found newly-significant at the same bitplane, or at most at two consecutive bitplanes. This observation can be intuitively explained due to the removal of high-frequency (detail) information based on the repetitive application of the low-pass analysis filter. Experimental validation is given in Figure 7, where most of the blocks become significant within bitplanes $b = \{10, 9\}$. Secondly, the high-frequency spatio-temporal subbands exhibit a heavy-tail distribution based on the doubly-stochastic model of (4) and therefore the probability of significance (and newly-significance) is more skewed, as seen in Figure 6.

2.4 Rate Approximation of Intra-band Embedded Coding

In this section, we derive rate estimations for an embedded coding scheme using a quadtree decomposition structure followed by block-coding for the maximum depth of the quadtree. This follows the system model outlined in Figure 2. Using the high-frequency and low-frequency wavelet coefficient models and the previously-derived probability estimates from Proposition 1-Proposition 3 in Section 2.3, we derive the rate of quadtree coding, block coding, and coefficient refinement coding. The derived rate estimates can be modified to fit a variety of wavelet coding schemes consisting of subsets of the general structure of Figure 2, i.e. quadtree-based coders [23] [24], and block-based coders [20] [25] [26].

2.4.1 Rate of Significance-map Coding for High-frequency Subbands

In most video frames, the probability of block significances at various levels in the quadtree varies considerably depending on the spatio-temporal subband statistics and the block size, since small blocks encapsulate small areas while large blocks represent large areas within each subband. For most practical wavelet video coders, we can simplify our analysis by assuming that the significance map encoding of quadtree structures remains virtually uncompressed. An example that strongly suggests this property is given in Table 3, where operational measurements from the coder of [30] were used for a variety of input video sequences. Note that while the rate is not identically 1 bit per quadtree symbol encoded, the difference in size between quadtrees leads to a distribution that cannot be well compressed, especially for lower bitplanes. In the remainder of this chapter, we assume that the rate of significance-map coding for blocks in the quadtree structure is approximately equal to the number of times significance-map symbols are encoded. We additionally assume that coefficients in blocks of all sizes are sufficiently correlated so that the i.i.d. joint Gaussian distribution with a fixed local variance θ can be used to model them.

The significance of a block in the quadtree decomposition may be encoded in two cases: *i*) If the block is found newly significant at bitplane b, its significance will be encoded at that moment and it will never be encoded again; *ii*) if the block's parent is found to be significant at bitplane b even though the block itself is non-significant, it will be coded continuously until the block is found newly significant. Condition (*ii*) is added in most state-of-the-art coders to exploit intra-band spatial correlation of wavelet coefficients.

Under the above-stated two conditions, we now derive the probability of block significance, which corresponds to the rate of block significance coding. In general, if a block at quadtree level k, $2 \le k \le K$, has n coefficients, its parent block at level $k-1$ has $4n$ coefficients[1]. The number of symbols (or rate) used to encode the significance of a block of size n found significant at bitplane b depends on the probability that its parent is found significant at higher bitplanes $b+r$, $r \ge 0$ (which means that the block significance will be coded a total of $r+1$ times). Given the subband variance σ^2, this can be formulated as:

$$R_{\text{block\_newsig}}(\upsilon_b, n) = \sum_{r=0}^{B_{\max}-b} \Pr\{\text{sig}(T_{b+r}, 4n) = 1 \mid \text{newsig}(T_b, n) = 1\} \tag{14}$$

Averaging the rates over all bitplanes $B_{\min}, \ldots, B_{\max}$, we get the following rate estimate:

$$\begin{aligned} R_{\text{block\_sig}}(\upsilon_{B_{\min}}, n) &= \sum_{b=B_{\min}}^{B_{\max}} \Pr\{\text{newsig}(T_b, n) = 1\} R_{\text{block\_newsig}}(\upsilon_b, n) \\ &= \sum_{b=B_{\min}}^{B_{\max}} \delta_{\upsilon_b, n}^{\text{band}} \sum_{r=0}^{B_{\max}-b} \Pr\{\text{sig}(T_{b+r}, 4n) = 1 \mid \text{newsig}(T_b, n) = 1\} \end{aligned} \tag{15}$$

where $\text{band} \in \{\text{low,high}\}$ depending on the type of frequency subband we are interested in. The probability within the summation of (14) can be estimated by obtaining the local distribution of parameter θ given the newly-significant child block at bitplane b, and then determining the probability of a significant coefficient (in the other 3 child blocks) at bitplane $b+r$, thereby deriving the conditional probability of significance for the parent block of $4n$ coefficients:

$$\Pr\{\text{newsig}(T_b, n) = 1 \mid \text{sig}(T_{b+r}, 4n) = 1\} = \Pr\{\text{newsig}(T_b, n) = 1 \mid \Theta\} \tag{16}$$

where:

$$\Theta \sim p(\theta \mid \text{sig}(T_{b+r}, 4n)) = \frac{\Pr\{\text{sig}(T_{b+r}, 4n) \mid \theta\} \, p(\theta)}{\Pr\{\text{sig}(T_{b+r}, 4n)\}} = \frac{\left(1 - \left[\text{erf}\left(\frac{T_{b+r+1}}{\sqrt{2\theta}}\right)\right]^{4n}\right) \frac{1}{\sigma^2} e^{-\frac{\theta}{\sigma^2}}}{\chi_{\upsilon_{b+r}, 4n}^{\text{band}}} \tag{17}$$

Thus, (16) becomes:

[1] In the subsequent derivations of this chapter, whenever blocks of size n and $4n$ appear in the same expression, it is implied that the first is the child block at quadtree level k while the second is the parent block at quadtree level $k-1$.

$$\begin{aligned}\Pr\{\text{newsig}(T_b,n)=1 \mid \Theta\} &= \int_{\theta=0}^{\infty} \Pr\{\theta \mid \text{newsig}(T_{b+r},4n)\}\cdot \Pr\{\text{newsig}(T_b,n)=1 \mid \theta, \text{newsig}(T_{b+r},4n)\}\,d\theta \\ &= \frac{1}{\delta_{\upsilon_b,n}^{\text{band}}}\int_{\theta=0}^{\infty}\frac{1}{\sigma^2}e^{-\frac{\theta}{\sigma^2}}\left(1-\left[\text{erf}\left(\frac{T_{b+r}}{\sqrt{2\theta}}\right)\right]^{4n}\right)\left(\left[\text{erf}\left(\frac{T_{b+1}}{\sqrt{2\theta}}\right)\right]^{n}-\left[\text{erf}\left(\frac{T_b}{\sqrt{2\theta}}\right)\right]^{n}\right)d\theta\end{aligned} \quad (18)$$

Note that $[\text{erf}(T/x)]^n$ can be approximated by an indicator function. Hence, we approximate the integral of (18) by treating the two multiplicative $\text{erf}(\cdot)$ terms as an indicator and a step function to obtain:

$$1-\left[\text{erf}\left(\frac{T_{b+r}}{\sqrt{2\theta}}\right)\right]^{4n} \cong \text{I}\left(\theta \geq \frac{T_{b+r}^2}{\ln(4n)^{1.296}+0.166}\right) \quad (19)$$

$$\left[\text{erf}\left(\frac{T_{b+1}}{\sqrt{2\theta}}\right)\right]^{n}-\left[\text{erf}\left(\frac{T_b}{\sqrt{2\theta}}\right)\right]^{n} \cong \text{I}\left(\frac{T_b^2}{\ln(n)^{1.296}+0.166} \leq \theta \leq \frac{T_{b+1}^2}{\ln(n)^{1.296}+0.166}\right) \quad (20)$$

where I is the indicator function. Based on (19) and (20), we can approximate (18) by:

$$\gamma_{\upsilon_b,n,r}^{\text{band}} \cong \begin{cases}\frac{1}{\delta_{\upsilon_b,n}^{\text{band}}}\left(\chi_{\upsilon_{b+r},4n}^{\text{band}}-\chi_{\upsilon_{b+1},n}^{\text{band}}\right)^{+} & ,\frac{T_b^2}{\ln(n)^{1.296}+0.166} \leq \frac{T_{b+r}^2}{\ln(4n)^{1.296}+0.166} \\ 1 & ,\text{otherwise}\end{cases} \quad (21)$$

Combining (15), (19)–(21) together, we obtain the final expression:

$$R_{\text{block\_sig}}(\upsilon_{B_{\min}},n) = \sum_{b=B_{\min}}^{B_{\max}} \delta_{\upsilon_b,n}^{\text{band}} \sum_{r=0}^{B_{\max}-b} \gamma_{\upsilon_b,n,r}^{\text{band}} \quad (22)$$

The average rate per coefficient is $R_{\text{block\_sig}}(T_b,n)/n$. If we let n be the smallest block size, then we must sum up the rates for K levels of the quadtree decomposition in the subbands of each spatial resolution to obtain the total rate for quadtree encoding:

$$R_{\text{quadtree}}(\upsilon_{B_{\min}}) = \sum_{k=0}^{K-1}\frac{R_{\text{block\_sig}}(\upsilon_{B_{\min}},4^k n)}{4^k n} \quad (23)$$

2.4.2 Block and Refinement-coding Rate in High-frequency Subbands

In this subsection, we estimate the rate of encoding both the significance and the refinement of coefficients based on the quantization level (or minimum bitplane level) $T_{B_{\min}}$.

Table 3. Examples of the number of 8×8 blocks encoded at each bitplane (symbols) and the average rate of encoding each block significance (rate is measured in bits-per-symbol), using the coder of [30]. Note that the rate per symbol increases as the bitplane decreases and approaches one bit-per-symbol.

H frames, temporal level 3, spatial level 2	$T_b = 16$		$T_b = 32$		$T_b = 64$	
	Symbols	Rate	Symbols	Rate	Symbols	Rate
Coastguard, HL	2254	0.97	1556	0.85	447	0.73
Foreman, HL	1056	0.87	571	0.79	284	0.79
Silent, HL	1787	0.91	1046	0.81	348	0.79

First, we consider the significance rate. In block-based coders like JPEG2000 [20] or ESCOT [26] (i.e. intra-band coders that do not employ quadtree decompositions), a subband is simply divided into blocks, which are then coded independently. In these algorithms, the significance of each coefficient within the block is encoded. If a coefficient is significant, then its refinement bits are also encoded. The significance of a coefficient relative to the threshold $T_{B_{\min}}$ is a binary value. Hence, the rate from independently encoding the significance of each coefficient can be expressed by the binary entropy function $\mathrm{H}(\rho_{B_{\min}})$, where $\rho_{B_{\min}}$ is the probability that the coefficient is significant compared to $T_{B_{\min}}$. However, when context-based entropy decoding is employed, dependencies between neighboring coefficients can be exploited, such that the rate (based on the doubly-stochastic model) is [10]:

$$R_{\mathrm{ZC}}(\upsilon_{B_{\min}}, n) = \int_{\theta=0^+}^{\infty} \frac{1}{\sigma^2} e^{-\frac{1}{\sigma^2}\theta^2} \mathrm{H}\left(\mathrm{erf}\left(\frac{T}{\sqrt{2\theta}}\right)\right) d\theta \cong \mathrm{H}(\rho_{B_{\min}}) - 0.6707\upsilon_{B_{\min}} e^{-1.407\upsilon_{B_{\min}}} \qquad (24)$$

A common weakness of using only block-coding methods without quadtree coding [20] [25] [26] is that the spatial distribution of local variances in the spatio-temporal subbands is not exploited, since the same context-conditioning scheme is utilized for all blocks. Coders combining quadtree coding and block-coding techniques [21] [22] exploit the spatial correlation of local variances by using the quadtree decomposition which inherently assigns fewer symbols to the insignificant areas: if all coefficients within a certain block are insignificant, quadtree-based coders return a single “0” for that block and do not encode the coefficients within that block. However, for (minimum-sized) blocks that are significant, context coding is used for the coefficients in the block. Hence, the effective significance coding rate for the blocks resulting at the maximum quadtree depth depends on the probability of the smallest blocks being significant, and the context coding rate conditioned on the block being significant:

$$\begin{aligned} R_{\mathrm{ZC,QB}}(\upsilon_{B_{\min}}, n) &= \mathrm{H}(\mathrm{sig}(T_{B_{\min}}, 1) \mid \mathrm{sig}(T_{B_{\min}}, n), \Theta) \\ &= \int_{\theta=0}^{\infty} p(\theta) \cdot \Pr\{\mathrm{sig}(T_{B_{\min}}, n) | \theta\} \cdot \mathrm{H}(\Pr\{\mathrm{sig}(T_{B_{\min}}, 1) \mid \mathrm{sig}(T_{B_{\min}}, n), \theta\}) d\theta \end{aligned} \qquad (25)$$

where $\mathrm{sig}(T_{B_{\min}}, 1)$ indicates the significance test at bitplane $B_{\min}$ for an individual coefficient within a wavelet subband. The probability of coefficient significance given block significance and local variance θ can be solved using Bayes rule:

$$\Pr\{\mathrm{sig}(T_{B_{\min}},1) | \mathrm{sig}(T_{B_{\min}},n),\theta\} = \frac{\Pr\{\mathrm{sig}(T_{B_{\min}},n) | \mathrm{sig}(T_{B_{\min}},1),\theta\} \Pr\{\mathrm{sig}(T_{B_{\min}},1) | \theta\}}{\Pr\{\mathrm{sig}(T_{B_{\min}},n) | \theta\}} = \frac{1-\mathrm{erf}\left(\frac{T_{B_{\min}}}{\sqrt{2\theta}}\right)}{1-\left[\mathrm{erf}\left(\frac{T_{B_{\min}}}{\sqrt{2\theta}}\right)\right]^n}$$

(26)

The resulting expression is:

$$R_{\mathrm{ZC,QB}}(\upsilon_{B_{\min}}, n) = \int_{\theta=0}^{\infty} \frac{1}{\sigma^2} e^{-\frac{1}{\sigma^2}\theta^2} \left(1-\left[\mathrm{erf}\left(\frac{T_{B_{\min}}}{\sqrt{2\theta}}\right)\right]^n\right) \mathrm{H}\left(\frac{1-\mathrm{erf}\left(\frac{T_{B_{\min}}}{\sqrt{2\theta}}\right)}{1-\left[\mathrm{erf}\left(\frac{T_{B_{\min}}}{\sqrt{2\theta}}\right)\right]^n}\right) d\theta \qquad (27)$$

Notice that the expression is parametrical to the number of coefficients per smallest block, n. A minimum block size of approximately $n = 16$ has been shown to achieve the most savings over pure context-based coding.

Consider now the rate of refinement-coding for significant coefficients. For a given error subband, the rate of quantizing a significant coefficient X based on its neighborhood information $\mathcal{N}X$ can be estimated as [10]:

$$R_{\mathrm{refinement}}(Q_{B_{\min}}(X) \mid \mathcal{N}X) \cong 1 - \log_2\left(\frac{1}{\rho_{B_{\min}}} - 1\right) - \frac{\log_2 \rho_{B_{\min}}}{1-\rho_{B_{\min}}} - \frac{0.2988}{(\upsilon_{B_{\min}} + 0.9773)^{0.8}} \qquad (28)$$

Therefore, the total coding rate can be estimated as:

$$R_{\mathrm{high}}(\upsilon_{B_{\min}}) = R_{\mathrm{quadtree}}(\upsilon_{B_{\min}}) + R_{\mathrm{ZC,QB}}(\upsilon_{B_{\min}}, n) + \rho_{B_{\min}} R_{\mathrm{refinement}}(Q_{B_{\min}}(X) \mid \mathcal{N}X) \qquad (29)$$

2.4.3 Coding Rate of Low-frequency Subbands

Following the independent Gaussian model assumption for the low-frequency subbands of L frames, we derived the following rate estimate for the coding of low-frequency wavelet coefficients.

Proposition 4: The rate of encoding a low-frequency coefficient can be approximated as follows:

$$R_{\text{low}}(\upsilon_{B_{\min}}) \cong \mathrm{H}\Big(\operatorname{erf}\big(\tfrac{\upsilon_{B_{\min}}}{\sqrt{2}}\big)\Big) + \operatorname{erfc}\big(\tfrac{\upsilon_{B_{\min}}}{\sqrt{2}}\big)\log_2\left(\operatorname{erfc}\big(\tfrac{\upsilon_{B_{\min}}}{\sqrt{2}}\big)\frac{\sqrt{2\pi e}}{\upsilon_{B_{\min}}}\right) - \frac{\upsilon_{B_{\min}}}{\sqrt{2\pi}}\exp\big(-\tfrac{\upsilon_{B_{\min}}^2}{2}\big)\log_2 e \qquad (30)$$

*Proof***:** We use the estimation method of Mallat and Falzon [29] for the rate in the low-rate (high distortion) region:

$$\begin{aligned}\mathrm{H}(Q_{B_{\min}}(X)) &\cong \mathrm{H}(|X| \geq T_{B_{\min}}) + p(|X| \geq T_{B_{\min}})\mathrm{H}(Q_{B_{\min}}(X) \mid (|X| \geq T_{B_{\min}}))\\ &= \mathrm{H}\Big(\operatorname{erf}\Big(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\Big) + p(|X| \geq T_{B_{\min}})\mathrm{H}(Q_{B_{\min}}(X) \mid (|X| \geq T_{B_{\min}}))\end{aligned} \qquad (31)$$

For significant coefficients, we use the high-resolution hypothesis from [20], which is:

$$\mathrm{H}(Q_{B_{\min}}(X)) \cong \mathrm{H}(X) - \log_2 \upsilon_{B_{\min}} \qquad (32)$$

This gives us:

$$\begin{aligned}\mathrm{H}(Q_{B_{\min}}(X) \mid (|X| \geq T_{B_{\min}})) &\cong \mathrm{H}(X \mid (|X| \geq T_{B_{\min}})) - \log_2(\upsilon_{B_{\min}})\\ &= -2\int_{\upsilon_{B_{\min}}}^{\infty} \frac{\exp(-\frac{x^2}{2})}{\operatorname{erfc}(\frac{\upsilon_{B_{\min}}}{2})\sqrt{2\pi}}\log_2\left(\frac{\exp(-\frac{x^2}{2})}{\operatorname{erfc}(\frac{\upsilon_{B_{\min}}}{2})\sqrt{2\pi}}\right)dx - \log_2(\upsilon_{B_{\min}})\\ &= \log_2\Big(\operatorname{erfc}\big(\tfrac{\upsilon_{B_{\min}}}{2}\big)\frac{\sqrt{2\pi}}{\upsilon_{B_{\min}}}\Big) + \int_{\upsilon_{B_{\min}}}^{\infty} \frac{\exp(-\frac{x^2}{2})}{\operatorname{erfc}(\frac{\upsilon_{B_{\min}}}{2})\sqrt{2\pi}}x^2\log_2(e)dx\\ &= \log_2\Big(\operatorname{erfc}\big(\tfrac{\upsilon_{B_{\min}}}{2}\big)\frac{\sqrt{2\pi}}{\upsilon_{B_{\min}}}\Big) + \log_2(e)\left(\frac{\upsilon_{B_{\min}}\exp(-\frac{\upsilon_{B_{\min}}^2}{2})}{\operatorname{erfc}(\frac{\upsilon_{B_{\min}}}{2})\sqrt{2\pi}} + \int_{\upsilon_{B_{\min}}}^{\infty}\frac{\exp(-\frac{x^2}{2})}{\operatorname{erfc}(\frac{\upsilon_{B_{\min}}}{2})\sqrt{2\pi}}dx\right)\\ &= \log_2\Big(\operatorname{erfc}\big(\tfrac{\upsilon_{B_{\min}}}{2}\big)\frac{\sqrt{2\pi e}}{\upsilon_{B_{\min}}}\Big) + \left(\frac{\upsilon_{B_{\min}}\exp(-\frac{\upsilon_{B_{\min}}^2}{2})}{\operatorname{erfc}(\frac{\upsilon_{B_{\min}}}{2})\sqrt{2\pi}}\right)\log_2 e\end{aligned} \qquad (33)$$

The proof follows from substituting (33) into (31). ■

(In order to avoid integrations in (30), we use the approximation for $\operatorname{erf}\big(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\big)$ given in (11).)

Notice that, for high-rate (low distortion) regions where the variance of each coefficient is significantly larger than the quantization stepsize (i.e. $\upsilon_{B_{\min}}$ is small), $\operatorname{erfc}\Big(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\Big) \approx 1$, and the rate estimate of (30) becomes the well-known high-rate approximation [20]:

$$R_{\text{low}}(\upsilon_{B_{\min}}) \cong \log_2\sqrt{2\pi e} - \log_2 \upsilon_{B_{\min}} \qquad (34)$$

Table 4 gives an example of the accuracy of (30) as a function of quantization step. We also present the results with the more conventional model of (34) used in prior work [10] in order to indicate the superior approximation achieved with the proposed estimation of (30). Notice from Table 4 that, although the proposed model still remains relatively inaccurate when only the highest 2-3 bitplanes are decoded, it becomes increasingly accurate as the number of decoded bitplanes increases.

Table 4. Example comparison between the actual encoded rate ("Encoded size" using the coder of [30]), the conventional approximation from prior work, and the proposed approximation of (30) for an 44x38 LL subband of an L-frame in the *Foreman* sequence.

Quantization step size (T_b)	Encoded size (bits)	Conventional approximation [10]	Error	Proposed approximation	Error
1024	1336	3289	146.2%	941	-29.6%
512	3208	4513	40.7%	3019	-5.9%
256	5032	5920	17.7%	5128	1.9%
128	6816	7507	10.1%	7053	3.5%
64	8472	9080	7.2%	8856	4.5%
32	10096	10661	5.6%	10595	4.9%

Based on the derived estimations of (29) and (34), the average coding rate per pixel over all subbands of MCTF-based wavelet video coding is:

$$R_{\text{total}}(\upsilon_{B_{\min}}) = 4^{-J} R_{\text{low},J,0} + \sum_{j=1}^{J}\sum_{m=1}^{3} 4^{-j} R_{\text{high},j,m} \tag{35}$$

where (band, j, m) indicates which model $\text{band}=\{\text{low,high}\}$ is used to determine the rate of the subband of the $m\text{th}$ orientation in the $j\text{th}$ scale of the wavelet frame decomposition, with $j = 1$ indicating the finest resolution, and $j = J$ indicating the coarsest resolution (lowest frequency). $R_{\text{low},J,0}$ indicates the coarsest LL subband.

2.5 Distortion Estimation for MCTF Wavelet Coding

In this section we determine the distortion of the various coding passes mentioned previously for the different subbands, and a general formulation of the average distortion for the combined decoding followed by MCTF reconstruction is derived.

2.5.1 Distortion of Decoding followed by Inverse Spatial DWT

As mentioned in Section 2.2, SAQ followed by the accumulation of all coding passes up to any bitplane $B_{\min}$ corresponds to a double-deadzone uniform quantization of wavelet coefficients. In other words, once the produced bitstream is truncated at bitplane $B_{\min}$, we have a quantizer of the form given in (1) with $T_{B_{\min}} = 2^{B_{\min}}\Delta$. The average distortion of a high-frequency subband when a uniform quantizer with this deadzone is applied is [10]:

$$\begin{aligned} D_{\text{high}} &= \mathrm{E}[(X-\hat{X})^2] \\ &= \left\{-\rho_{B_{\min}}\left(\upsilon_{B_{\min}} + 1/\sqrt{2}\right)^2 + 1 - \rho_{B_{\min}}{}^2\upsilon_{B_{\min}}{}^2/\left(1-\rho_{B_{\min}}\right)^2\right\}\sigma^2 \end{aligned} \tag{36}$$

where σ^2 is the variance of the Laplacian-distributed coefficients. We now derive the distortion of the low-frequency subband of an L frame.

Proposition 5: The estimated distortion for the low-frequency spatio-temporal subband is:

$$D_{\text{low}} = \left(\operatorname{erf}\left(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\right) - \sqrt{\frac{2}{\pi}}\upsilon_{B_{\min}} e^{-\frac{\upsilon_{B_{\min}}^2}{2}} + \frac{\upsilon_{B_{\min}}^2}{12}\operatorname{erfc}\left(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\right)\right)\sigma^2 \tag{37}$$

Proof: Based on the procedure in [29], we separate the distortion calculation D_{low} into the distortion of non-significant coefficients (deadzone), and the distortion of significant coefficients. The deadzone distortion is the variance of a truncated Gaussian at $T_{B_{\min}}$, which can be shown to be $D_{\text{low,zero}} = 1 - \sqrt{\frac{2}{\pi}}\upsilon_{B_{\min}} e^{-\frac{\upsilon_{B_{\min}}^2}{2}} / \operatorname{erf}\left(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\right)$ using integration by parts. For significant coefficients, we use the high-rate assumption [20] [29], $D_{\text{low,nonzero}} \cong \frac{\upsilon_{B_{\min}}^2}{12}\sigma^2$. Hence, the total distortion is the weighted sum of the two metrics, or:

$$D_{\text{low}} = p_{\text{low,zero}} D_{\text{low,zero}} + p_{\text{low,nonzero}} D_{\text{low,nonzero}} = \operatorname{erf}\left(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\right) D_{\text{low,zero}} + \operatorname{erfc}\left(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\right) D_{\text{low,nonzero}} \tag{38}$$

which gives us (37).

■

Notice that, similar to the corresponding rate estimation of (30), for low-distortion (high-rate) regions where the variance of each coefficient is significantly larger than the quantization stepsize (i.e. $\upsilon_{B_{\min}}$ is small), $\operatorname{erfc}\left(\frac{\upsilon_{B_{\min}}}{\sqrt{2}}\right) \approx 1$, and the distortion estimate of (37) converges to the well-known high-rate approximation of $D_{low} \cong \frac{\upsilon_{B_{\min}}^2}{12}\sigma^2$.

For all the different subbands at all scales of the DWT, we get an average distortion:

$$\mathbf{d} = 4^{-J} G_J \cdot D_{\text{low},J,0} + \sum_{j=1}^{J}\sum_{m=1}^{3} 4^{-j} G_j \cdot D_{\text{high},j,m} \tag{39}$$

where G_j is the synthesis gain of the wavelet filter at the jth scale level, and $D_{\text{band},j,m}$ is the expected distortion of the mth type subband at the jth scale level, with $\text{band}=\{\text{low},\text{high}\}$.

2.5.2 Distortion for MCTF Reconstruction

For generalized MCTF filtering, distortion takes on a linear combination of each L and H frame produced by the decomposition [10] [31]:

$$\bar{\mathbf{d}}_L^{(0)} = \sum_{k=1}^{T_{\text{MCTF}}} B^{(k)} \left(\prod_{j=1}^{k-1} A^{(j)} \right) \mathbf{d}_H^{(k)} + \left(\prod_{j=1}^{T_{\text{MCTF}}} A^{(j)} \right) \bar{\mathbf{d}}_L^{(T_{\text{MCTF}})} = [\mathcal{A}, \mathcal{B}_1, \ldots, \mathcal{B}_{T_{\text{MCTF}}}][\bar{\mathbf{d}}_L^{(T_{\text{MCTF}})} \mathbf{d}_H^{(1)}, \ldots, \mathbf{d}_H^{(T_{\text{MCTF}})}]^T \quad (40)$$

The last derivation of (40) is valid because the weight of each H-frame at each temporal level is a function of only the average number of connected pixels in the GOP. However, we note that this approximation can also be applied across several GOPs if the motion between GOPs is similar. In our experiments, linear minimum mean square error (MMSE) fitting is used to determine the weights of L-frames and H-frames and predict the distortion associated with the sequence.

2.6 Complexity of Entropy Decoding and IDWT

2.6.1 Generic Complexity Modeling for Video Decoding

Since many multimedia decoders today typically reside in a variety of handheld (Video iPod, 3G cellphones, etc) and portable devices (notebooks, PDAs) that have stringent power and processing constraints, they are in general more resource-constrained than encoders. Hence, while a similar complexity estimation framework can be likewise derived for the encoder, we opt to focus on the decoding complexity. A second (and more algorithm-related) reason is that the encoding complexity is strongly dominated by the motion estimation complexity rather than the coding operations. Hence, accurate modeling of embedded encoding per-se is of a lesser importance for the R-D-C analysis of the encoder, as it is for the decoder.

In order to represent different decoder (receiver) architectures in a generic manner at the encoder (server) side, in our recent work [11] [12] we have deployed a concept that has been successful in the area of computer systems, namely, a virtual machine. The key idea of the proposed paradigm is that the same bitstream will require/involve different resources/complexities on various decoders. We adopt a generic complexity model that captures the abstract/generic complexity metrics (GCMs) of the employed decoding or streaming algorithm depending on the content characteristics and transmission bitrate. GCMs are derived by computing or estimating the average number of times the different operations are executed, such as the number of read symbols during entropy decoding, the number of multiply-accumulate operations performed during inverse transform, the number of motion compensation operations per pixel or coefficient, and the frequency of invocation of fractional pixel interpolation. The value of each GCM may be determined at encoding time for each adaptation unit q (e.g. the q-th video frame, or the q-th macroblock) following experimental or modeling approaches [11]. The previous chapter demonstrated that the mapping of the derived GCMs to execution time provides a very accurate and straightforward manner of predicting the real (system-specific) complexity [36]. The added advantage of GCMs however is that they are not system-specific and they are also not restricted to a particular coding structure (predictive or MCTF-based). This makes them applicable for a broad class of motion-compensated video decoders. In this section, we focus on the derivation of entropy decoding and inverse transform GCMs based on stochastic models that analytically express the dependencies on the source characteristics and the algorithm operations.

2.6.2 Entropy Decoding Complexity

The complexity of decoding the quadtree significance at bitplane b depends on the size of the quadtree before the significance pass. Since the quadtree is virtually uncompressed for the vast majority of cases, the complexity is of the order of the quadtree significance map encoding rate:

$$C_{\text{quadtree}}(\upsilon_{B_{\min}}, n) \cong R_{\text{quadtree}}(\upsilon_{B_{\min}}, n) \tag{41}$$

with $R_{\text{quadtree}}(\upsilon_{B_{\min}}, n)$ given by (23) based on Proposition 1-Proposition 3. This includes both the number of read symbols (RS) associated with quadtree coding, and writing the significances into the quadtree structure.

Concerning block coding, we group together the number of symbols read from significance coding and refinement. Notice that, as long as the coefficient is in a significant block at bitplane b or higher, its significance will be coded, or it will be refined at bitplane b. Summing up all symbols read in the passes until bitplane $B_{\min}$ we have:

$$C_{\text{block}}(\upsilon_{B_{\min}}) = 4^{K-1} n \sum_{b=B_{\min}}^{B_{\max}} \chi_{\upsilon_b,n}^{\text{low}} + 4^{K-1} n \sum_{b=B_{\min}}^{B_{\max}} \chi_{\upsilon_b,n}^{\text{high}} \tag{42}$$

where $4^{K-1}n$ is the number of coefficients in the subband. Notice that the combination of (41) and (42) predicts the number of RS operations during entropy coding/decoding of a low or high-frequency spatio-temporal subband to a certain bitplane $B_{\min}$. Since each subband is encoded independently, the complexity metrics must first be estimated for each subband and then summed in the same weighted fashion as the rate calculation. In other words, for a given frame i, $1 \le i \le N$, we have:

$$C_{\text{op}}^{i}(\upsilon_{B_{\min}}) = 4^{-J} C_{\text{op},J,0}(\upsilon_{B_{\min}}) + \sum_{j=1}^{J} \sum_{m=1}^{3} 4^{-j} C_{\text{op},j,m}(\upsilon_{B_{\min}}) \tag{43}$$

where $\text{op} \in \{\text{quadtree,block}\}$ and $C_{\text{op},j,k}(\upsilon_{B_{\min}})$ is the quadtree and block coding complexity for each subband at spatial resolution j. Having obtained the RS estimates for quadtree and block coding, the expression $\alpha_{\text{quadtree}} C_{\text{quadtree}}^{i}(\upsilon_{B_{\min}}) + \alpha_{\text{block}} C_{\text{block}}^{i}(\upsilon_{B_{\min}})$ derives an estimate of the real complexity for frame i, where α_{op} is an approximate algorithmic (and platform dependent) complexity associated with each symbol used to perform operation op. See [36] for extended examples of adaptive generation of weighting factors for mapping GCM estimates to platform-specific complexity.

2.6.3 Complexity of the Inverse Spatial DWT

The complexity of the inverse DWT depends on the number of taps of the filter used as well as on the implementation method (convolution or lifting). In [12], the transform-related complexity of a coding system that processes N video frames is modeled by expressing it as a decomposition into two functions relating to: *i*) the percentage of non-zero coefficients for a given SAQ threshold T_b (function $\mathcal{T}_{\text{nonzero}}$); *ii*) the sum of run-lengths of zero wavelet coefficients (function $\mathcal{T}_{\text{runlen}}$). The motivation behind (*i*) is that in an input-adaptive implementation, the number of non-zero multiply-accumulate operations in the synthesis filter-bank is directly proportional to the percentage of non-zero coefficients. Moreover, the distribution of the zeros within the transform subbands (as expressed by the sum of run-lengths) affects the number of consecutive filtering operations that can be avoided altogether. Once an estimate of $\mathcal{T}_{\text{nonzero}}$ and $\mathcal{T}_{\text{runlen}}$ is derived, the complexity of the inverse spatial DWT (non-zero MAC operations) is formulated as [12]:

$$\mathrm{FC}^N = \mathbf{C}^N_{\text{nonzero}} \cdot \boldsymbol{\mathcal{T}}^N_{\text{nonzero}} + \mathbf{C}^N_{\text{runlen}} \cdot \boldsymbol{\mathcal{T}}^N_{\text{runlen}} + \mathbf{C}^N_{\text{dec\_const}} \cdot \mathbf{1} \tag{44}$$

with $\boldsymbol{\mathcal{T}}^N_{\text{nonzero}}$ and $\boldsymbol{\mathcal{T}}^N_{\text{runlen}}$ the N-element vectors of the corresponding functions and the parameter vectors $\mathbf{C}^N_{\text{nonzero}}$ and $\mathbf{C}^N_{\text{runlen}}$ can be estimated based on linear regression and off-line training [12]. In this chapter, two main differences exist in the derivation of the non-zero MAC operations of the IDWT in comparison to [12]. Firstly, linear MMSE fitting is used to determine $\mathbf{C}^N_{\text{nonzero}}$, $\mathbf{C}^N_{\text{runlen}}$ and predict the number of non-zero MAC operations associated with the sequence. This is equivalent to the process performed for the derivation of the final MCTF distortion in (40) (Section 2.5.2.5.2). More importantly, we present an analytical calculation of the decomposition functions mentioned above based on stochastic source models. In this way, the proposed analytical derivations create a clear link between the source parameters (variances of the distributions) and the derived complexity estimates. The decomposition function $\mathcal{T}_{\text{nonzero}}$ for the high-frequency spatio-temporal subbands is derived by (6), while for the low-frequency spatio-temporal subbands it is derived by:

$$\mathcal{T}_{\text{nonzero}} = \operatorname{erfc}\left(\tfrac{\upsilon_{B_{\min}}}{\sqrt{2}}\right) \tag{45}$$

with $\operatorname{erf}\left(\tfrac{\upsilon_{B_{\min}}}{\sqrt{2}}\right)$ approximated as in (11). In addition, $\mathcal{T}_{\text{runlen}}$ is derived by the percentage of non-significant blocks for a certain SAQ threshold $T_{B_{\min}}$, expressed by:

$$\mathcal{T}_{\text{runlen}} = \Pr\{\operatorname{sig}(T_{B_{\min}}, n) = 0\} = 1 - \chi^{\text{band}}_{\upsilon_{B_{\min}},n} \tag{46}$$

with $\chi^{\text{band}}_{v_{B_{\min}},n}$ estimated by (9) for the high-frequency temporal subbands and by (10) for the LL subband of the L frames. Following the lifting dependencies of popular wavelet filter-pairs, we set an average of $n = 64$ since a window of 7×7 coefficients and 9×9 coefficients is used in the lifting steps of the inverse DWT for the low and high-frequency subbands [20].

Additionally, note that the number of taps also affects memory usage in the system. While memory usage is another concern in battery-limited devices, in this chapter we are primarily concerned with time-based complexity, as this more greatly affects the performance of delay-sensitive applications.

2.7 Simulation Results

In this section we validate the derived analytical R-D-C expressions by presenting experiments with three common interchange format (CIF) resolution sequences (*Coastguard*, *Foreman*, *Silent*) that encapsulate a variety of motion and texture characteristics. Apart from validating the theoretical modeling of rate-distortion and complexity-distortion, the interplay of rate and complexity for achieving the same video quality under different coding structures is discussed.

For validation purposes, we utilize the spatial-domain MCTF version of the coder of [30] that performs multihypothesis MCTF decomposition with a variety of temporal filters and intra-band quadtree-based coding of the significance maps, and block-based intra-band coding after a block size of 4×4 coefficients is reached in the quadtree decomposition. Figure 8–Figure 10 present our results for a variety of spatial (S) and temporal (T) decomposition levels. Distortion, as estimated by (36)–(40) in Section 2.5, is converted into peak signal to noise ratio (PSNR). The entropy-decoding complexity is quantified by the number of read symbols per second. For the inverse transform, we plot the number of non-zero MAC operations per second (FC/s). The results demonstrate that the proposed R-D-C modeling predicts the experimental behavior of the advanced MCTF-based wavelet video coder accurately for all the different cases under investigation. Different choices for MCTF (temporal) levels and spatial decomposition levels lead to different tradeoffs in rate and complexity for the same distortion in the decoded video.

Since the proposed models of Sections 2.3–2.6 enable accurate estimation of rate and complexity for a variety of decoding distortion, the derived framework can be used in a variety of applications where rate or complexity tradeoffs are of paramount importance (see [4] [8] [9] [11]). For example, the R-C curve may be used to optimize post-encoding bitstream shaping, where an encoded bitstream may be truncated and transmitted at a lower rate based on decoder-specified complexity bounds.

There are several interesting aspects to note from our results. For example, for good quality video decoding (PSNR range of 32 dB–40 dB) there is typically an overhead of about 300 to 500 kbps when one uses two temporal levels instead of four and an overhead of about 600 to 900 kbps when one uses two temporal and two spatial levels. Notice that the exact overhead is both sequence and bitrate dependent and the proposed theoretical modeling captures this behavior accurately. Apart from the rate overhead, there is also an increase in the number of entropy decoding operations by about $2.5 \cdot 10^5$ to $5 \cdot 10^5$ RS/s and 10^6 to $2 \cdot 10^6$ RS/s for the "2T-4S" and "2T-2S" cases (respectively) in comparison to the "4T-4S" case. However, concerning the IDWT complexity, Figure 10 demonstrates that a large variation exists in the performance of the different approaches depending on the sequence and bitrate region. The case of "2T-2S" is the best in terms of operations per second, followed by the "4T-4S" case and by the "2T-4S" case, since the two latter require more spatial reconstruction levels. The fact that the "2T-4S" case appears to be worse than the "4T-4S" case can be explained by the increase in the non-zero coefficients due to the fact that the "2T" case includes four times more L frames as compared to the "4T" case, and L frames contain a higher percentage of non-zero coefficients in comparison to H frames (for the same quantization parameters). It is also interesting to notice that, for the low to medium rate coding of the *Coastguard* sequence, the "4T-4S" case is the most efficient both in R-D and C-D performance. The proposed modeling approach agrees with all these observations, a fact that validates the importance of analytical R-D-C modeling methods that adapt based on both source and algorithm statistics.

It is also interesting to note that, if one ignores the coding bitrate and focuses on the complexity-distortion tradeoffs, Figure 9 and Figure 10 reveal that, for the same number of entropy decoding operations, the “4T-4S” case can provide gains of 2 to 8 dB in comparison to the other alternatives. On the other hand, the “2T-2S” case may outperform the other decompositions by 2.5 to 10 dB for the same number of non-zero MAC operations during the IDWT. On different platforms where each entropy decoding operation and IDWT MAC operation may have different respective computational workloads and/or energy consumption levels, the significant tradeoffs between the different types of decoding complexities can be exploited to optimally configure coder parameters to run on a specific system.

Finally, it is interesting to investigate how rate and complexity change for different coding parameters for a higher resolution video, e.g. in sequences of Standard Definition (SD) format. The entropy decoding results for the 720x480 *Mobile* sequence (30 frames/sec) are presented in Figure 11. As seen in the figure, our theoretical approximations are fairly accurate in predicting the large performance gain of 4 temporal levels over 2 temporal levels of decomposition. Interestingly, this gain is not so prominent for CIF sequences, as indicated by Figure 8 and Figure 9. One reason is that the MCTF process can better exploit the correlation between neighboring pixels and coefficients in SD sequences due to the decrease of spatio-temporal aliasing in comparison to CIF sequences. Hence, the percentage of non-zero coefficients in the high-frequency subbands is decreased. Consequently the number of read symbols (entropy decoding complexity) and the required bitrate to encode H-frames are reduced when using more temporal levels.

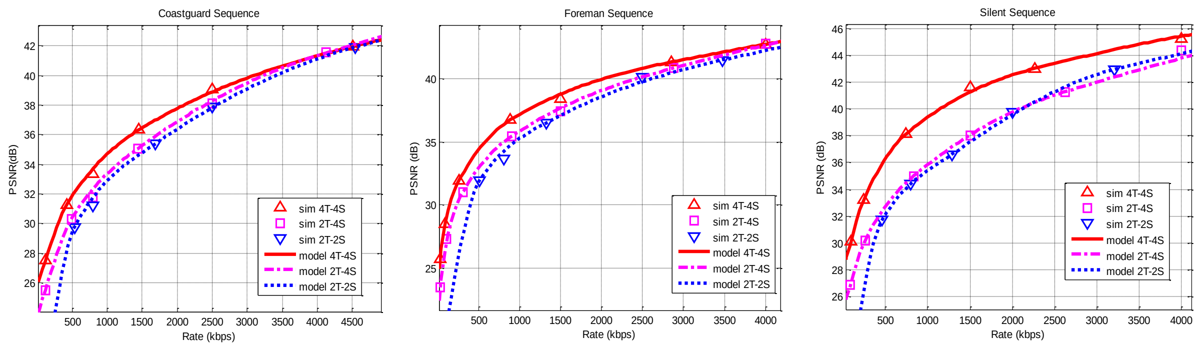


Figure 8. Rate-distortion plots for different configurations of the spatio-temporal decomposition parameters.

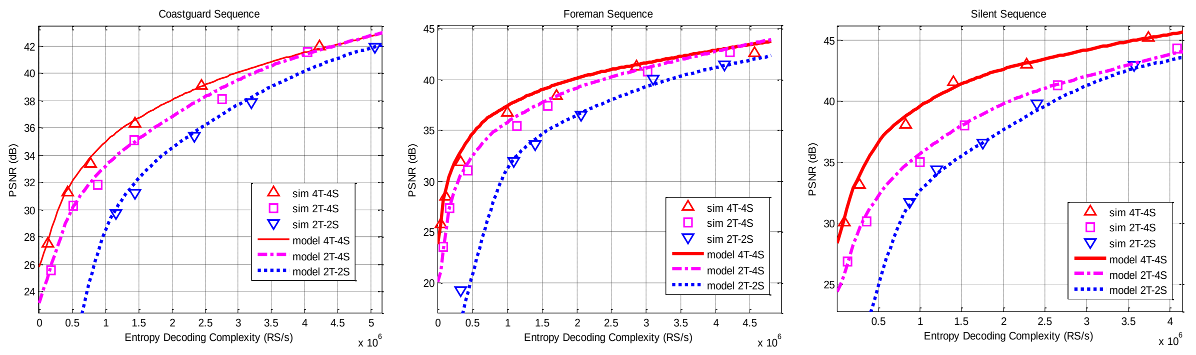


Figure 9. Entropy decoding complexity vs. distortion plots for different spatio-temporal decomposition parameters, where "S" and "T" indicate the number of spatial and temporal levels (respectively).

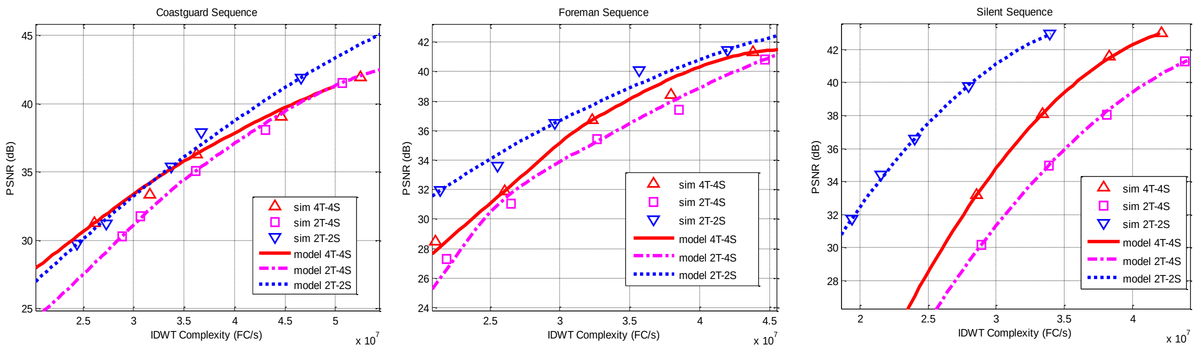


Figure 10. IDWT complexity vs. distortion plots for different spatio-temporal decomposition parameters, where "S" and "T" indicate the number of spatial and temporal levels (respectively).

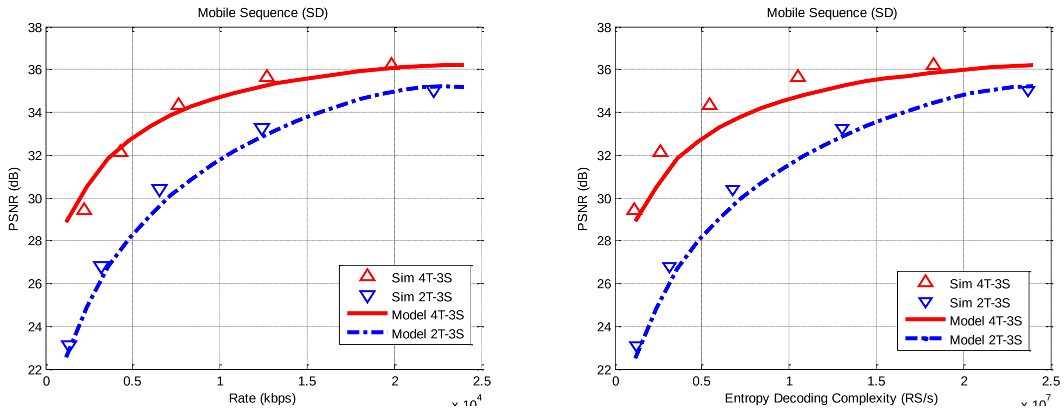


Figure 11 Rate and entropy decoding complexity curves for 720x480 *Mobile* sequence. The cases of 4 and 2 temporal decomposition levels are presented. Both encodings use 3 spatial decomposition levels.

2.8 Conclusions

This chapter presents an analytical modeling framework that derives rate, distortion and (decoding) complexity predictions for wavelet-based video coders. Our analysis encapsulates a broad variety of coding techniques found in state-of-the-art coding schemes. By analytically deriving probabilities for block and coefficient significance according to the quantization threshold (for both low and high-frequency temporal subbands), we are able to establish analytical models that approximate well the R-D-C behavior of a state-of-the-art wavelet-based video coder. In this way, this chapter complements prior work on operational rate-distortion modeling for video coders by extending its applicability to a broader coding paradigm. At the same time, it complements complexity modeling frameworks proposed in earlier work by deriving analytically the input statistics used in these approaches. As such, the modeling framework in this chapter bridges the gap between the operational measurements used in prior complexity modeling work and stochastic estimates common in rate-distortion modeling work.

The theoretical R-D-C analysis presented in this chapter may guide the construction of more efficient intra-band coding mechanisms targeting error frames in particular. An open question concerns the efficiency of quadtree coding versus block coding mechanisms (both compression-wise and implementation-wise) and the optimal setting (e.g. minimum block size or combination of coding passes) for a coder that encodes error frames using both schemes in succession. Perhaps more importantly, the proposed R-D-C analysis allows for the efficient exploration of complexity and rate tradeoffs for different video qualities and resolutions. As indicated by our results with CIF and SD-resolution videos, a careful selection of coder parameters is important for optimizing the performance in a complexity-distortion or rate-distortion sense. A thorough investigation of the theoretical R-D-C tradeoffs between different coder parameters for different video resolutions is an interesting topic for future research.

CHAPTER 3
A Model-based Approach to Processor Power Adaptation for Video Decoding Systems

3.1 Introduction

Dynamic voltage scaling (DVS) has been proposed as a solution for processing real-time tasks while reducing energy consumption on processors that support multiple operating frequencies [38][39][40]. In CMOS circuits, power consumption is given by $Power \propto V^2 \cdot C_{eff} \cdot f$, where V, C_{eff}, f denote the voltage, effective capacitance and operating frequency, respectively. The energy spent on one task is proportional to the time spent for completing that task and time is inversely proportional to frequency. Hence, the energy is proportional to the square of the voltage, i.e., $Energy \propto V^2 C_{eff}$. The energy spent on one process can be reduced by decreasing the voltage, which will correspondingly increase the delay. Based on statistical estimates of the cycle requirement (i.e. complexity or execution time) for each job, a DVS algorithm assigns an operating level (i.e. power and frequency) for processing that job while meeting delay requirements for that job.

In the past few years, a wide variety of DVS algorithms have been proposed for delay-sensitive applications [41]-[48],[50][51]. Some DVS algorithms perform optimization over only one or two tasks, such that the processor power level is determined on the fly to meet imminent (soft) deadlines while considering either the worst case execution time (WCET) [42][43], or the average case execution time (ACET)

[47]. While these approaches have very low computational complexity, the performances are limited in that future tasks with imminent deadlines may require extremely high processing power to finish in time after the completion of the current task. On the other hand, more robust DVS algorithms, such as the cycle-conserving and look-ahead earliest deadline first DVS [41], and Feedback Control-based DVS [44], schedule the power based on multiple future task deadlines. The complexity of such approaches can become huge for large job buffer sizes, since many job deadlines must be jointly considered in such scheduling schemes. This may often be the case for multimedia where video packet arrivals over a network are nondeterministic, and many packets are required to decode each video frame. Consequently, various lower-complexity DVS approaches were proposed, where the number of tasks released for execution (and hence, the number of deadlines to consider in the DVS algorithm) could be controlled by adjusting various parameters, such as the "aggressiveness" factor in [48].

In spite of the wide variety of algorithms proposed, current DVS approaches are limited in several ways:

- Current DVS algorithms lack simple yet accurate complexity models for multimedia tasks. Many DVS algorithms are often optimized in an application-agnostic or ad-hoc manner, or otherwise they add significant overhead to online complexity adaptation [44][48][11][36]. Other more formal, stochastic DVS approaches [52][53][77] for generic applications use models that are not well-suited toward the highly time-varying video decoding complexity.
- Current DVS algorithms often use worst-case or average case complexity measurements (e.g. [42][43][47]), which neglect the fact that multimedia compression algorithms require time-varying resources that differ significantly between jobs. Moreover, "worst-case" and "average-case" metrics do not exploit the information stored by the second moment of job execution times, or by the execution time distributions themselves. Without such information, it is hard to make analytical miss rate guarantees under different voltage scheduling policies.

- Current DVS algorithms do not cooperate with multimedia applications to obtain complexity statistics, which may vary across different coders, different sequences, and different bit rates.
- While the generic framework of imprecise computation has been considered as an approach for loss-tolerant applications such as multimedia [54][55], these algorithms do not take full advantage of the properties of the multimedia algorithm to optimize the quality or energy savings by jointly adapting the power level and the workload. Moreover, there is either no explicit consideration of the distortion impact in loss tolerant multimedia processing, or else an unrealistic model is used for distortion.

To address the limitations above, we propose the following solutions in this chapter:

- We construct a complexity model that not only *explicitly* considers coder operations and frame dependencies (i.e. task deadlines), but can also be characterized by only a few parameters. Importantly, we show that complexity statistics can be decomposed into the sum of complexity metrics that follow simple, well-known distributions. By using offline training sequences, we derive complexity distributions for different classes of sequences and encoded frame types, such that the encoder/server can transmit these parameters online with very low overhead whenever the sequence characteristics or coder parameters change [11][36]. This enables the decoder to adapt its model based on these parameters, such that the decoding system can optimally plan its use of resources based on *a priori* transmitted complexity traffic characteristics.
- We propose an offline linear programming (LP) solution to analyze the optimality of DVS algorithms by deriving an operational lower bound for energy consumption, subject to processing all jobs before their delay deadlines (i.e. zero miss rate). Moreover, this linear programming solution provides the optimal offline scheduling solution regardless of leakage power.
- Based on the online complexity distribution adaptation scheme, we propose two DVS algorithms. First, we introduce an online robust linear programming (rLP) solution for DVS that takes advantage of modeled parameters such as the means and

variances of different classes of jobs. Second, we propose a queuing theoretic model driven DVS algorithm, which allows the processing frequency to be smoothed across time-varying workloads. Both DVS approaches are shown to greatly reduce energy consumption compared to existing algorithms, and can both analytically adapt their algorithms for different target job miss rate and energy consumption levels.

- We propose a quality-aware DVS algorithm based on priority scheduling, where jobs are decomposed based on their dependencies and contributions to overall video quality, such that more important jobs are processed first. In this way, the video stream can be decoded at various quality levels given different power levels, even if the average power is insufficient for decoding all jobs before their deadlines. We demonstrate experimentally that the quality-aware DVS algorithm can retain high quality even if the available energy is reduced significantly.

The chapter is organized as follows. Section 3.2 introduces multimedia application-specific stochastic models to determine workload characteristics. Section 3.3 introduces the offline LP approach for calculating the lower bound of energy consumption for DVS algorithms. Section 3.4 queuing model approach for deadline-driven DVS algorithms. Section 3.6 introduces a quality-adaptive DVS via a priority scheduling approach, where more important jobs are processed first. Section 3.7 provides performance comparisons between the look-ahead DVS algorithm and our deadline-driven queuing-based DVS algorithms, and shows different average power and quality tradeoffs achieved by priority-based DVS. Finally, Section 3.8 concludes our work.

3.2 Stochastic Modeling of Application Workload

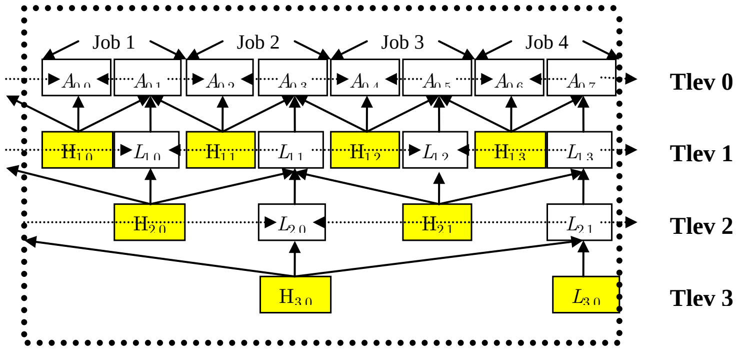
3.2.1 Challenges and Previous Works for Complexity Modeling

Modeling the complexity of state-of-the-art video coders is a challenging task due to the complex group-of-pictures (GOP) structures that exist, where many neighboring video frames are coded together. In addition, some advanced coders (e.g. MPEG4) allow the GOP structure to change over time to adapt to changing video source characteristics, in which case the complexity model must adapt by recapturing statistics whenever the GOP structure changes. As a result of these complex and potentially changing encoding structures, research on complexity prediction and modeling have traditionally fallen into two categories. The first category involves methods that ignore coder-specific operations, such as coarse levels of empirical modeling for complexity [75], or the use of a statistical sliding window [43]. The second category involves modeling complexity at a fine granular level based on functions associated with the process of decoding (e.g. entropy decoding, inverse transform, etc.). However, these works do not provide coder-aware theoretical models [76], or else they are based on platform-independent "virtual" complexities that can not be mapped into real complexity (time) in a straightforward manner [11] [36]. In the following section, we propose an accurate, low-complexity mixed online-offline modeling technique based on using well-known, analytical distributions.

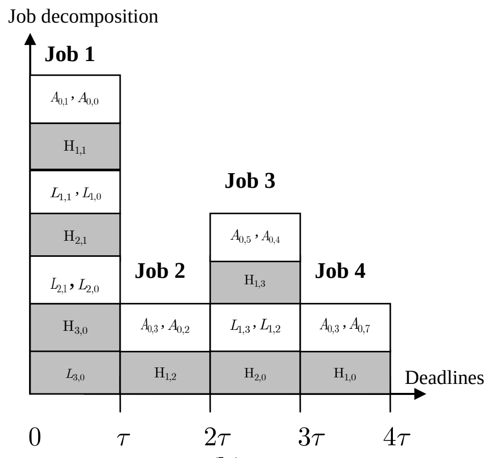
3.2.2 Deriving and Modeling the Service Distribution for Jobs

Based on our discussion above on mixing training data with analytical models, we show an example using an MCTF coder with 4 temporal levels. In order to differentiate between various jobs associated with each GOP, we define job classes $i = 1,...,I$, where a job belongs to class i if it is the $i\text{th}$ job to be decoded in its associated GOP. For example, in the MCTF structure shown in Figure 12, there are a total of 4 classes of jobs which correspond to the decoding of the $\{A_{0,0}, A_{0,1}\}$, $\{A_{0,2}, A_{0,3}\}$, $\{A_{0,4}, A_{0,5}\}$, and $\{A_{0,6}, A_{0,7}\}$ frames in a GOP. For 4 temporal levels, we have 8 job classes. It is important to classify these jobs in such a way because jobs in the same class are expected to have similar complexities, and similar waiting times before being processed.

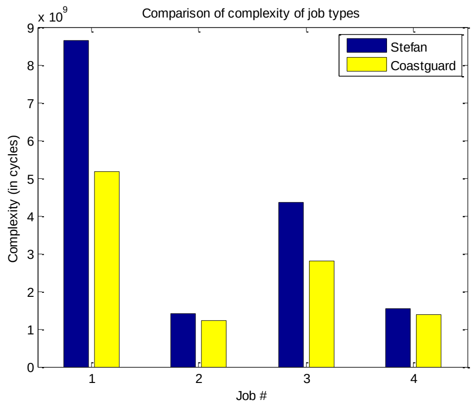
We collected job execution times (offline) from a set of 11 training sequences with 16 GOPs each, decoded at 7 different bit rates. In Figure 13a-b, we show the complexity distributions for various job classes, averaged over all sequences, decoded at bit rates 1152kbps and 320kbps. Here, tics indicate the value measured by the internal processor clock counter. We noticed that the complexity distributions shared similar features, such as the existence of peaks. We also explored the dependency of complexity distributions on various sequences by collecting data from different classes of jobs for particular sequences over 7 different bit rates (from 200kbps to 1.5mbps) and normalizing the measurements by their scales in order to obtain an average distribution shape for each sequence. It was discovered that the shapes vary greatly between different sequences, as shown in the comparison of the sequences *Coastguard* and *Stefan* in Figure 13c-d.


Job 1
Job 2
Job 3
Job 4
$A_{0,0}$ $A_{0,1}$ $A_{0,2}$ $A_{0,3}$ $A_{0,4}$ $A_{0,5}$ $A_{0,6}$ $A_{0,7}$
$H_{1,0}$ $L_{1,0}$ $H_{1,1}$ $L_{1,1}$ $H_{1,2}$ $L_{1,2}$ $H_{1,3}$ $L_{1,3}$
$H_{2,0}$ $L_{2,0}$ $H_{2,1}$ $L_{2,1}$
$H_{3,0}$ $L_{3,0}$
Tlev 0
Tlev 1
Tlev 2
Tlev 3

(a)



(b)



(c)

Figure 12: (a) Periodic 3 temporal level MCTF structure (with dotted lines indicating motion compensation operations), (b) Job decomposition for each soft decoding deadline, (c) Corresponding workloads for the jobs in sequences *Stefan* and *Coastguard*.

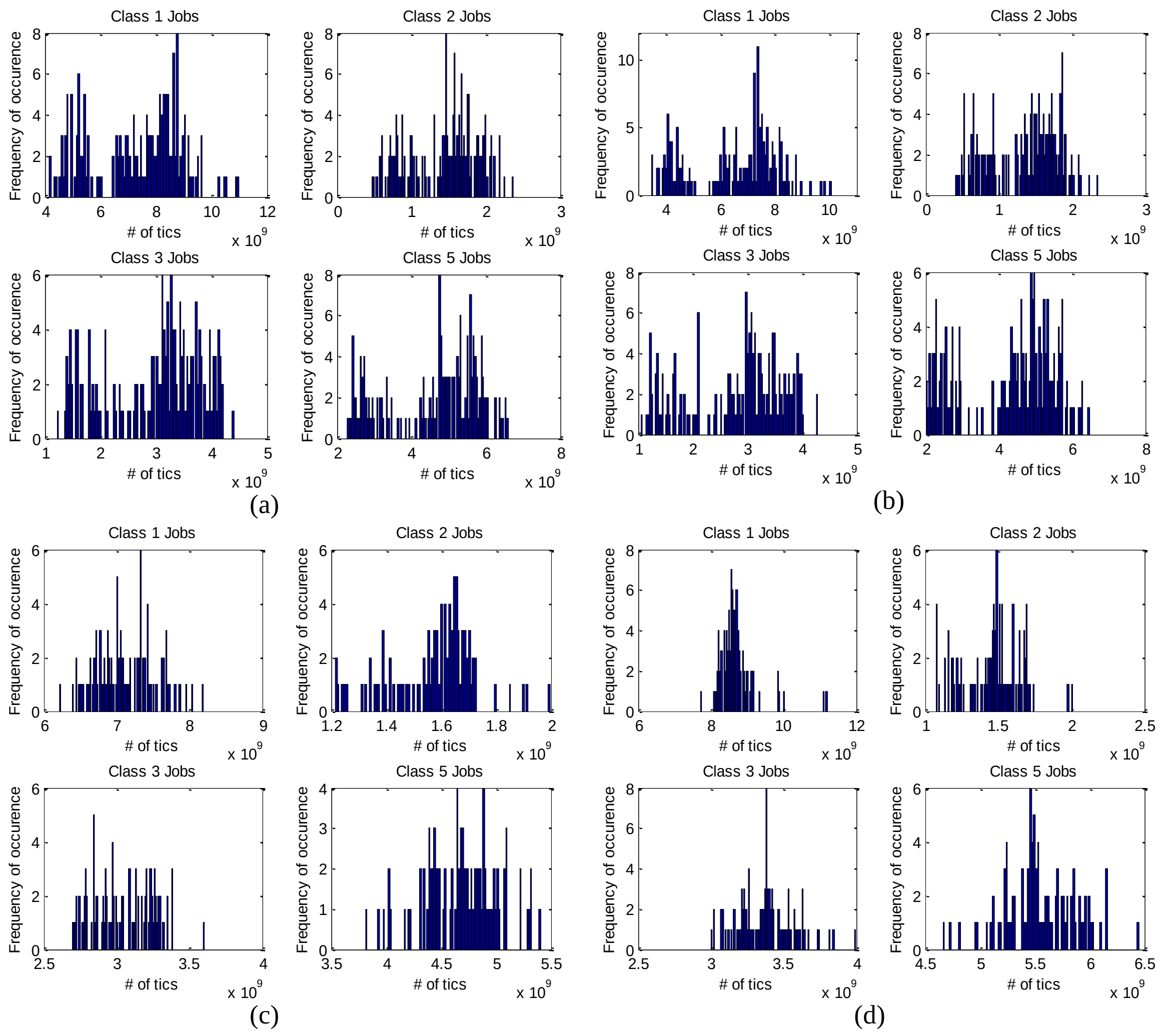


Figure 13 The total service time distribution for various classes of jobs (out of a total of 8 classes) in 4 temporal level MCTF for (a) all training sequences at bit rate 1152kbps, (b) at bit rate 320kbps. (c) The *Coastguard* sequence complexity shape, (d) *Stefan* sequence complexity shape.

In order to better model the complexity analytically, we investigated the complexities contributed by different steps of a decoding process. Decoding jobs often involves multiple different functions such as entropy decoding (ED), inverse transform (IT), motion compensation (MC), and fractional pixel interpolation (FI). Hence, the total complexity for class i for a sequence seq, C_i^{seq}, is the sum of complexities associated with each of the various decoding functions:

$$C_i^{seq} = C_{i,\mathrm{ED}}^{seq} + C_{i,\mathrm{IT}}^{seq} + C_{i,\mathrm{MC}}^{seq} + C_{i,\mathrm{FI}}^{seq} \tag{47}$$

where each $C_{i,\mathrm{op}}^{seq}$ indicates the total complexity associated with one type of decoding step for a job of class i. Since a job of class i is composed of decoding and reconstructing various frames at various temporal levels, we can further decompose $C_{i,\mathrm{op}}^{seq}$, $op \in \{ED, IT, MC, FI\}$, into a sum of decoding steps performed at each temporal decomposition level. For example, ED complexity of a class 1 job in 3-level MCTF, as shown in Figure 12, consists of entropy decoding frames $\{L_{3,0}, \mathrm{H}_{3,0}, \mathrm{H}_{2,1}, \mathrm{H}_{1,1}\}$. Likewise, for Ω temporal level MCTF, the ED complexity for a class 1 job can be expressed as the sum of complexities for all of its entropy decoding tasks:

$$C_{1,\mathrm{ED}}^{seq} = C_{L(\Omega),\mathrm{ED}}^{seq} + \sum_{\omega=1}^{\Omega} C_{\mathrm{H}(\omega),\mathrm{ED}}^{seq} \tag{48}$$

where $C_{fr(\omega),ED}^{seq}$ is the entropy decoding complexity of decoding a frame of type fr at temporal level ω. We can finally model $C_{fr(\omega),\mathrm{op}}^{seq}$ with simple distributions that require only a few parameters, and sum up the distributions to form C_i^{seq}. A particular interesting example comes from entropy decoding, where the normalized complexity distribution $\hat{C}_{fr(\omega),ED}^{seq}$ is Poisson, i.e.

$$p_{fr(\omega),ED}^{seq}(n) = \frac{\left(\nu_{fr(\omega)}^{seq}\right)^n e^{-\nu_{fr(\omega)}^{seq}}}{n!} \tag{49}$$

where n is a Poisson bin number, $p_{fr(\omega),ED}^{seq}(n)$ is the probability that the normalized complexity falls into bin n, and $\nu_{fr(\omega)}^{seq}$ is a shape parameter for the normalized complexity distribution. The real entropy decoding complexity distribution can be modeled by a shifted and scaled Poisson distribution, i.e.:

$$C_{fr(\omega),ED}^{seq} \cong a_{fr(\omega)}^{seq} \hat{C}_{fr(\omega),ED}^{seq} + b_{fr(\omega)}^{seq} \tag{50}$$

where $a_{fr(\omega)}^{seq}$ and $b_{fr(\omega)}^{seq}$ are sequence and frame dependent constants. Figure 14 shows the normalized ED complexities $\hat{C}_{fr(\omega),ED}^{seq}$ for various L-frames and H-frames (a-b) averaged over all sequences for bit rates 1152kbps and 320kbps, and (c-d) for particular sequences *Coastguard* and *Stefan*. Notice that the distributions in Figure 14 denote a subset of the complexities forming the distributions for different classes of jobs in Figure 13 using (50), (48), and (47). Table 5 shows the coefficient values for the normalized entropy decoding complexity distribution averaged over all training sequences, along with individual sequences *Coastguard* and *Stefan*. Note that in all 3 cases, the coefficients a_i^{seq}, b_i^{seq}, and ν_i^{seq} vary significantly for different sequences. Hence, in practice, coefficients need to be estimated separately for different sequences. The same modeling technique may be applied to inverse transforms and motion compensation.

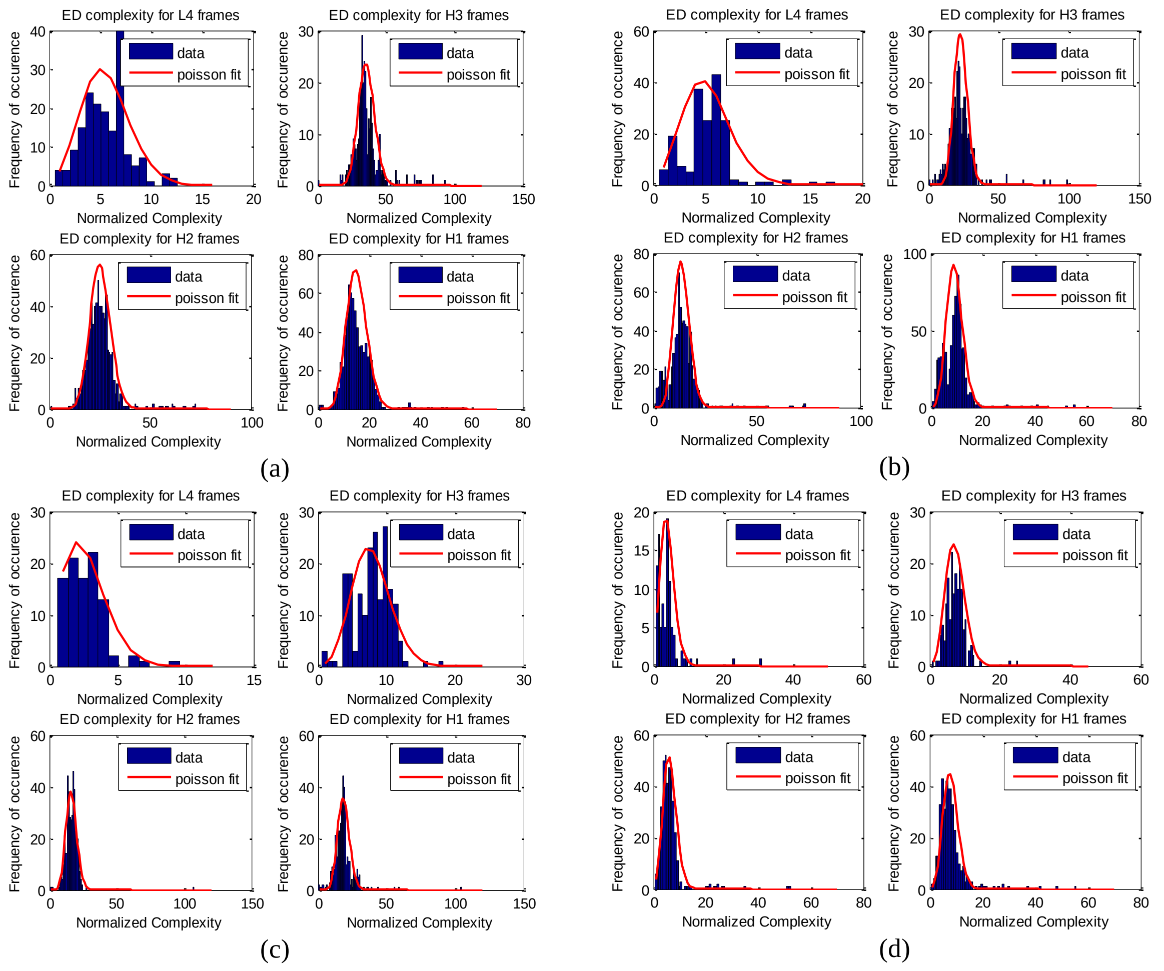


Figure 14 Normalized entropy decoding complexity for various L and H frames in a 4 temporal level MCTF GOP for (a) all training sequences at bit rate 1152kbps and (b) at bit rate 320kbps. (c) The *Coastguard* sequence complexity shape, (d) *Stefan* sequence complexity shape.

Table 5 Affine transform coefficients for the entropy decoding complexity in 4 level MCTF.

Frame	All Sequences (1152kbps)			Coastguard			Stefan		
	a_i^{seq}	b_i^{seq}	ν_i^{seq}	a_i^{seq}	b_i^{seq}	ν_i^{seq}	a_i^{seq}	b_i^{seq}	ν_i^{seq}
L4	4.4e7	8.7e8	21	53e6	2.5e8	2.6	18e6	5.9e8	4.0
H3	1.1e7	0.9e8	9.4	16e6	1.5e8	7.8	7.0e6	2.7e8	7.4
H2	1.7e7	2.0e8	13	7.4e6	1.1e8	16	8.7e6	2.2e8	6.3
H1	3.1e7	4.8e8	8.7	5.0e6	0.9e8	19	7.6e6	1.5e8	8.1

While modeling complexity in this fashion is highly source dependent, our novelty lies in the low complexity and high accuracy of updating the complexity model whenever changes occur in the video source statistics. During long video sequences, advanced coders may change GOP sizes, coding rates, and frame sizes many times due to time-varying source statistics. Without a good complexity model, only loose bounds for complexity can be derived based on coarse parameters, such as the mean and variance of frame sizes and coding bit rates across entire sequences [68]. However, using our complexity model, the encoder can update the decoder's information of the video source by sending only a few complexity distribution parameters prior to a new sequence or scene change, and the decoder can use these parameters to form accurate complexity distributions (The reader is referred to [11] for more details on possible implementations of how these parameters can be efficiently transmitted to the decoder.).

Before moving into queuing theoretic DVS, we make an important remark about the complexity model. The complexity distributions, which are measured and fitted to well-known distributions, are also based on the algorithmic decoder operations. As shown in the above example, entropy decoding complexity follows a simple shifted and scaled Poisson distribution, which is the limiting distribution when there are only a few high complexity tasks and many low complexity tasks. Indeed, algorithmically, entropy decoding for video frames follows such a distribution, since only a few decoded bits are used to reconstruct complex coder structures such as zero-trees, while most decoded bits are used to decode and refine significant coefficients. On the other hand, inverse transform followed a nearly constant complexity due to the particular implementation of the coder. Bidirectional motion compensation complexity led to the existence of two peaks, which occurred due to various macroblocks that required either a single prediction, or two predictions. Due to the space constraint in our manuscript, we omit a thorough analysis of each of these distributions. However, it should be noted that there is good reason to analytically model complexity based on these specific distributions, since they capture the underlying decoder implementation.

3.3 Evaluating the Optimality of DVS Algorithms

In this section, we formalize the objective of DVS algorithms, and provide an offline LP-based algorithm to determine the lower bound for energy consumption under 0% job miss rate (more details can be found in [81]).

3.3.1 Formulation of the Goal of Real-time DVS

For our real-time deadline-driven multimedia DVS problem, we are given a sequence of decoding jobs. For each job (which can be one frame or a pair of frames depending on the GOP structure), we are given its complexity, arrival time and display deadline. The goal of a DVS algorithm is to find a scheduling solution, which consists of the time and the operating voltage for each switch, to minimize total energy consumption. The constraint is that the decoder can only start decoding a job after it arrives, and each job should be finished before its display deadline.

Given M decoding jobs, let $C = \{C_1,\ldots,C_M\}$, $T = \{T_1,\ldots,T_M\}$, $D = \{D_1,\ldots,D_M\}$ be the complexity, arrival time and display deadline of jobs, respectively; let $F = \{F_0,\ldots,F_k\}$, $P = \{P_0,\ldots,P_k\}$ (F_0 and P_0 for sleeping mode) be associated clock frequencies and powers for K voltage levels, respectively; let $S = \{T_s, V_s, N\}$ be the scheduling solution, where N is the number of voltage switches, $T_s=\{t_0,\ldots,t_N, t_0=0\}$ and $V_s=\{v_0,\ldots,v_n\}$ is the time and voltage level for each switch. When the precise complexity of each job is known, the constraints for the problem are given by deterministic C_i and T_i. However, when uncertainties exist in the workload, C_i and T_i can be viewed as stochastic variables and DVS scheduling algorithms cannot guarantee that all jobs will be decoded before their deadlines. Hence, in the stochastic case, the hard deadline constraint can be replaced with the constraint of keeping the miss rate for jobs within a tolerable range.

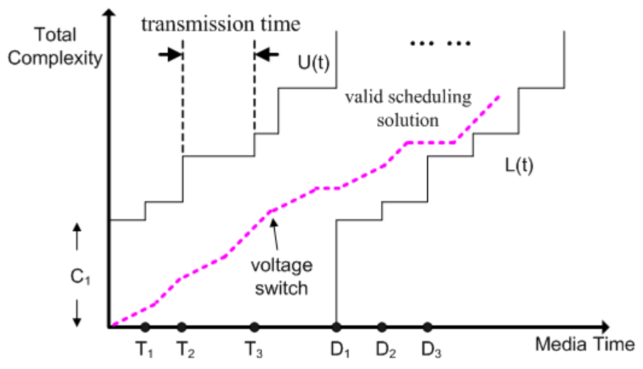


Figure 15 DVS problem formulation

We further illustrate the DVS problem in Figure 15 Here, the step function *U(t)* is the accumulated complexity (i.e., total complexity in terms of clock circles) of decoding jobs transmitted since time zero, and the widths of the steps are transmission times of jobs over network. Another step function *L(t)* indicates the total complexity that needs to be processed by time *t* in order to meet display deadlines. I.e.,

$$U(t) = \sum_{j=1}^{k}(C_j), for\ \ T_{k-1} < t \le T_k,\ 1 \le k \le M,\ \ T_0 = 0 \qquad (51)$$

$$L(t) = \sum_{j=0}^{k-1}(C_j), for\ \ D_{k-1} \le t < D_k,\ 1 \le k \le M, C_0 = 0, D_0 = 0 \qquad (52)$$

Since the decoder cannot start decoding a job before it is received from network, and it must finish the job before its deadline, a valid DVS solution is a piecewise linear curve between *U(t)* and *L(t)*. The slope of each piece indicates the associated clock frequency of the selected voltage level and a corresponding power is associated with each frequency.

The arrival time intervals can vary between each job. As shown in Figure 3.1, the transmission time for job 3 is larger than others. This can often occur when the network bandwidth is time-varying, which is common in wireless networks.

3.3.2 The Operational Lower Bound for DVS

In this section, we propose a method for evaluating offline, based on collected statistics about the workloads of decoded jobs, the operational lower bound for energy consumption. This allows us to evaluate retroactively the performance of our previously used online DVS scheme.

In particular, in the offline scenario, we know the precise complexity and arrival time of each decoding job, and hence we can obtain the optimal scheduling solution.

We define a *transition point* as the time when a new job arrives (i.e. any T_i), or when a job deadline is reached (i.e. any D_i). We also define an *adaptation interval* as the time period between two adjacent transition points. The adaptation intervals for sample *U(t)* and *L(t)* curves are marked in dotted lines in Figure 16. We now prove an important result for voltage scheduling.

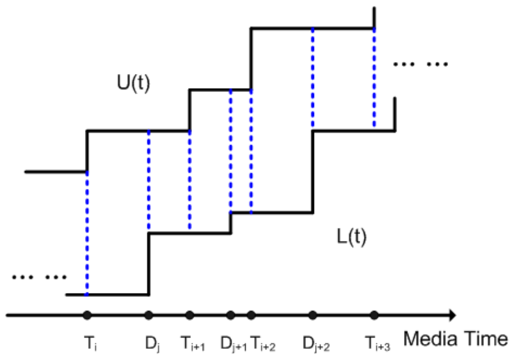


Figure 16 Adaptation intervals

Theorem 1: Within a single adaptation interval (i.e. when *U(t)* and *L(t)* are constant), an arbitrary ordering of any feasible voltage schedule is feasible and consumes the same amount of energy.

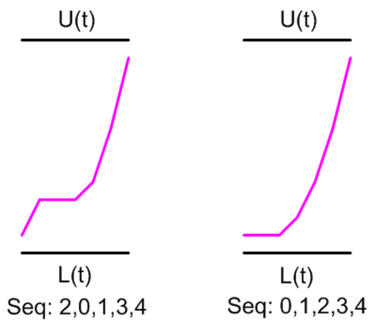


Figure 17 Different voltage scheduling orderings

Proof: Suppose we have a voltage time allocation of each voltage level in this interval. Then, the total energy consumption is the sum of each allocated time multiplies the corresponding power. Similarly, the total complexity consumption is the sum of each allocated time multiplies the corresponding frequency. Then, if the time allocation is fixed, the energy consumption is fixed. Since the voltage scheduling solution is valid and *U(t)* and *L(t)* are stable within this interval, the piecewise linear solution curve will not break the bound in spite of the ordering. As an example in Figure 4.2, the complexity consumption of sequence 2,0,1,3,4 and 0,1,2,3,4 (where the numbers refer to the slopes) is the same. ■

Theorem 4.1 is the *key idea* to map the DVS problem into a tractable LP problem. Rather than finding the precise times for voltage switch, which would create an intractable integer linear programming (ILP) problem as in [78], we instead solve the *percentage* of time that the processor is operating at each voltage level within each adaptation interval. The LP problem can be formulated as follows:

Problem Formulation 1: label the transition points as an ordered set $I = \{I_0,\ldots,I_L\}$, where $I_0=0$ and $I_L = T_{end}$, i.e. we have a total of L adaptation intervals. For these L intervals, we have voltage level allocation vectors given by $A = \{A_1,\ldots,A_L\}$, where $A_i=\{A_{i0},\ldots,A_{iK}\}$ and A_{ij} is the allocation of voltage level j in adaptation interval i. Then, the DVS problem is:

$$\min\ E = \sum_{i=1}^{N}\sum_{j=0}^{K}(A_{ij}\bullet P_j\bullet(I_i - I_{i-1})) \tag{53}$$

such that

$$0 \le A_{ij} \le 1,\ for\ 0 \le j \le K\ and\ \sum_{j=0}^{K}A_{ij} = 1 \tag{54}$$

$$L(I_n) \le \sum_{i=1}^{n}\sum_{j=0}^{K}(F_j\bullet A_{ij}\bullet(I_i - I_{i-1})) \le U(I_n), \forall\ 1 \le n \le L \tag{55}$$

Here, the unknown is the voltage level allocation vectors given by *A*. The constraint shown in (4.3) is that the valid DVS solution between *U(t)* or *L(t)* defined in (51) and (52).

One can easily prove that the problem defined in (53) to (54) is a linear programming problem. Hence, with this formulation, solving this LP problem leads to the optimal solution for offline DVS problem. Note that this formulation is *pervasive*: the operating voltages can take on any set of discrete values, and there is no requirement for the power-frequency model (no need for a convex power-frequency function). Furthermore, this formulation is also applicable to other delay-sensitive DVS problems of real-time applications.

3.4 Online Robust Linear Programming DVS

For online multimedia applications, where jobs are received through a network, we often do not know the precise complexity and arrival times of each decoding job. Nevertheless, the idea of mapping DVS into a linear programming problem in Section 3.3.2 can still be used for online DVS. We solve the stochastic online DVS problem by sequentially solving robust linear programs (rLP). We label our proposed algorithm as SLP/r.

The main idea of SLP/r is: we predict the stochastic complexity of decoding jobs in a future time window by using the means and variances of jobs, and solve an rLP problem to obtain the scheduling solution for the predicted decoding jobs in the window. After completing a decoding job (or several decoding jobs), we move the window forward based on the current time, adjust the predictions in the future window, and repeat the rLP based on possibly new statistics.

3.4.1 Consideration of Stochastic Complexity

The prediction of future decoding job complexities (in the sliding window) is crucial to our real-time DVS solution. In real time video transmission, this can be accomplished by having the encoder send complexity specifications, such as the mean and variance of each job class for each video scene, prior to transmitting the corresponding frames, as introduced in Section 3.2.2 [82]. Note that using only the mean for prediction may lead to a high miss rate. To reduce the probability of misses, we incorporate the variance into each job to estimate the bounded “worst case” complexity in a probabilistic manner. Due to the central limit theorem, when uncertainties accumulate over many jobs (See Figure 13 for workload distributions of single jobs.), the total workload tends toward a Gaussian distribution. In this case, the mean and variance for each job class can explicitly determine the miss rate probability under different adjustments of $U(t)$ and $L(t)$. The adjustments are based on a confidence level α to adjust the new bounds. Note that for jobs far into the future of a prediction window, the accumulated variance over many jobs may be large. Hence, a scaled coefficient α (possibly 0, such that only the mean is considered) can be used to guarantee feasibility. Using a small coefficient for jobs far into the future does not necessarily increase the miss rate, since the rLP solution will only determine the DVS schedule for imminent jobs, after which the rLP is solved again for the future jobs based on the decoding results.

The rLP problem for a given prediction window is as follows:

Problem Formulation 2:

$$\min\ E = \sum_{i=1}^{W}\sum_{j=0}^{K}(A_{ij}\bullet P_j\bullet\varphi) \tag{56}$$

such that

$$0 \le A_{ij} \le 1,\ for\ 0 \le j \le K\ and\ \sum_{j=0}^{K} A_{ij} = 1 \tag{57}$$

$$L(I_n) \le \sum_{i=1}^{n}\sum_{j=0}^{K}(F_j\bullet A_{ij}\bullet\varphi) \le U(I_n), \forall\ 1 \le n \le W \tag{58}$$

Where φ is the display interval, *W* is the prediction window size. Adaptation intervals *I, U(t)* and *L(t)* are defined as follows (detailed description is in section 5.2):

$$I = \{I_0,...,I_W\}, I_i = i\bullet\varphi \tag{59}$$

$$U(t) = \sum_{j=1}^{k}(\widetilde{C}_{W_0+j}), I_{k-1} < t \le I_k,\ 1 \le k \le W \tag{60}$$

$$L(t) = \max(0, U(t-\theta\bullet\varphi)) \tag{61}$$

Where W_0 is the current adaptation interval and $\widetilde{C}_i$ is stochastic complexity of job *i* based on the measured means and variances. Specifically we have:

$$\widetilde{C}_{W_0+j} \le \rho_{W_0+j} + \alpha_j\bullet\sqrt{v_{W_0+j}} \tag{62}$$

$$\alpha_j = \max(0, \alpha\bullet(R-j+1)/R), 1 \le j \le W \tag{63}$$

where ρ_i and v_i is the mean and variance of stochastic complexity of job *i*, α is the confidence level set by the user and *R* is a constant. (63) indicates that, α_j is linearly scaled between α and 0 over time. Note that a tradeoff between miss rate and energy consumption could be achieved by tuning α. For example, increasing α will make the bounds tighter and lead to more energy consumption but a lower miss rate.

One can easily show that the problem defined by equations (56) to (63) is an rLP problem. Note that with stochastic complexity model, the proposed online algorithm applies to other real-time applications although we only use video decoding as an example.

After we finish decoding one job, we need to adjust *U(t)* and *L(t)* dynamically. The idea is shown in Figure 18. The gray area indicates the variance part of prediction, the dotted line indicates the adaptation intervals and the dotted area indicates the bound adjustments. Figure 5.1(a) shows the solution from robust linear programming using mean and variance of each job class as a prediction. The real complexity of each job is shown in Figure 5.1(b). In this particular case, we used three adaptation intervals to finish decoding job No.1. As our granularity for recalculating the solution is one adaptation interval, we may consume more complexity than that of the specific job. As shown in Figure 5.1(b), after finishing job No.1, we also finished job No.2 and part of job No.3. Hence, we need to adjust the prediction dynamically to give more accurate prediction. As shown in Figure 5.1 (c), the dotted area indicates that part of job No.3 is already finished in the previous interval, and when we move the window forward, we need to adjust *U(t)* and *L(t)* accordingly.

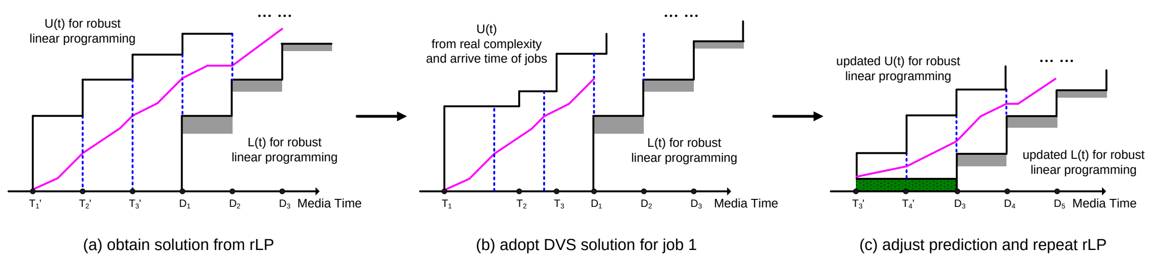

U(t) for robust linear programming
... ...
L(t) for robust linear programming
T1' T2' T3' D1 D2 D3 Media Time
(a) obtain solution from rLP
U(t) from real complexity and arrive time of jobs
... ...
L(t) for robust linear programming
T1 T2 T3 D1 D2 D3 Media Time
(b) adopt DVS solution for job 1
updated U(t) for robust linear programming
... ...
updated L(t) for robust linear programming
T3' T4' D3 D4 D5 Media Time
(c) adjust prediction and repeat rLP

Figure 18 Dynamic adjust of prediction for rLP

3.4.2 Extension to Variable Communication

For SLP/r, another problem is that we need to deal with the variance of network bandwidth, because we do not know the exact arrival time of each job. The idea is that we assume that a network buffer at the decoding side collects packets and dispatches jobs to the decoder according to the display frame rate. We fix the dispatch time as θ display intervals before the display deadline of the job. This means that we predict adaptation intervals using only display intervals. In this fashion, we can reduce the number of adaptation intervals (and also the size of the rLP problem). In this case, the adaptation intervals *I*, *U(t)* and *L(t)* are defined as (5.4) to (5.6). If a job arrives before our scheduling solution (i.e. the real *U(t)* is higher than the complexity consumption line), we just switch voltages as guided by rLP. Sometimes, a job may arrive very late due to insufficient bandwidth of network, which may occur in unreliable wireless networks. In this case, if a job arrives too late to be processed, power gating can be used to shut down the processor until a new job arrives, based on which *U(t)* and *L(t)* are adjusted for the next rLP.

3.5 Online Queuing-based DVS Algorithm

Recall that online DVS algorithms, due to varying workloads, cannot always achieve 0% miss rate. In this section, we introduce a queuing theoretic approach to online processor power adaptation. While feedback mechanisms using queuing theory may not perform as optimally as the rLP DVS algorithm, as queuing theory determines mainly resulting miss rates and power consumptions in *steady state* rather than for transients, queuing theory nevertheless provides us with a tool to analyze delay characteristics based on precise stochastic models. Accurate delay estimation also enables us to construct better scheduling approaches (e.g. priority scheduling) that perform well under different energy consumption rates, such that scalable energy-quality tradeoffs can be made. In this section, we introduce two methods for estimating the miss rate given a fixed processor power level. We then propose two variations of a queuing-based online DVS algorithm that utilizes this delay estimation to provide feedback and adapt the processor operating level.

3.5.1 Delay Analysis

As discussed previously, we assume a network buffer periodically provides the decoder with jobs in the order of their display deadlines, i.e. releases the jobs for execution. We deploy a $D/G/1$ cyclic multi-class queuing system with single-service discipline for modeling a deadline-driven DVS system as shown in Figure 19. In cyclic scheduling with single service discipline, a job in class $i+1$ is always serviced directly after a job in class i, and a job of class 1 (in the next GOP) is always serviced directly after a job in class I (of the current GOP). The service policy for each class in this system can also be viewed as a single service discipline with a vacation period [60]. In particular, whenever a job of class i finishes service, the processor "goes on vacation" by servicing one job in each of the other classes, and "returns to service" to class i only after it has completed processing the jobs in other classes. Based on this service discipline and class-dependent service time distributions obtained through offline training or from an analytical model (Section III.3.2.1), we can determine the delay distribution for each class of jobs (Figure 19). The delays can then be used to drive power scheduling to ensure that jobs are decoded before their deadlines.

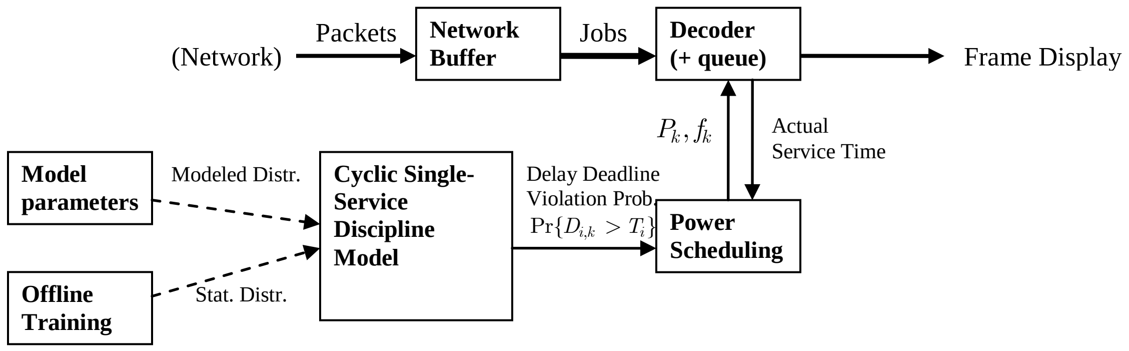


Figure 19: Queuing model for Deadline Driven DVS.

In order to analyze the queuing system, we first divide the interarrival time τ between jobs into N equally spaced time intervals, thereby obtaining discrete units of time that are τ / N seconds apart. Hence, each job interarrival period contains time indices $n \in \{0,1,...,N-1\}$. Let $S_{i,k}(m)$ be a random variable representing the discrete service time of the $i\text{th}$ job of the $m\text{th}$ GOP at processor speed f_k, and $V_{i,k}(m)$ the following vacation period. Let $W_{i,k}(m)$ be the waiting time for the $i\text{th}$ job of the $m\text{th}$ GOP. The service, vacation, and waiting times are shown for a GOP structure with two jobs in Figure 20.

We suggest two methods for computing the delay distribution of jobs. The first method is based on the vacation time distribution. Suppose the processor runs at a constant speed f_k until steady state is reached. We denote the steady state (or time average) random variables as $S_{i,k}$, $V_{i,k}$, and $W_{i,k}$ with distributions $s_{i,k}(n)$, $v_{i,k}(n)$, $w_{i,k}(n)$. Waiting times approximations for a $D/G/1$ cyclic service queue and $D/G/1$ queues with vacations have been analyzed extensively [21-23]. For example, given a discrete service time distribution $s_{i,k}(n)$ and vacation time distribution $v_{i,k}(n)$, the average waiting time is [59]:

$$E[W_{i,k}] = \frac{H''_{i,k}(1)}{2H'_{i,k}(1)} + \sum_{r=1}^{N-1} \frac{1}{1-z_r} - \frac{N(N-1) - \left[G'_{i,k}(1) + 2G'_{i,k}(1)H'_{i,k}(1) + H''_{i,k}(1)\right]}{2\left[N - G'_{i,k}(1) - H'_{i,k}(1)\right]} \tag{64}$$

where $G_{i,k}(z)$ and $H_{i,k}(z)$ are the z-transforms of $s_{i,k}(n)$ and $v_{i,k}(n)$ respectively, and z_r are the unique roots of the polynomial $z^N - G_{i,k}(z)H_{i,k}(z)$ that are on or inside the unit circle, except $z = 1$.

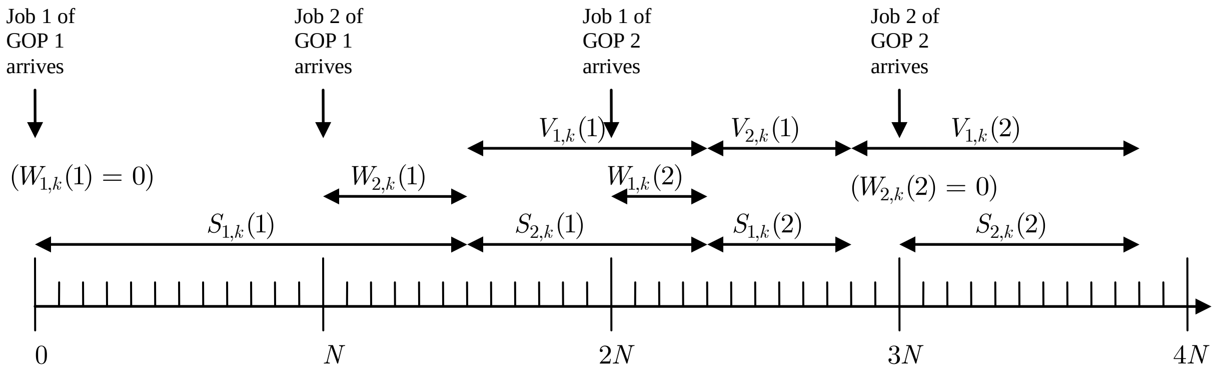


Figure 20: An example of discrete service, vacation, and waiting times for cyclic service and 2 classes per GOP. Jobs of each class arrive at multiples of $2N$, with the first job of class 1 arriving at time 0.

The second method, which is shown below, is an analytical queuing approach based on service times directly. Without loss of generality, assume that we have the waiting time distribution for class 1 jobs, $w_{1,k}(n)$. The total delay of class 1 jobs, $D_{1,k} = S_{1,k} + W_{1,k}$, follows the distribution:

$$d_{1,k}(n) = w_{1,k}(n) * s_{1,k}(n) \tag{65}$$

where $*$ indicates convolution. Now, define $\tilde{D}_{1,k}$ for class 1 to have the following distribution:

$$\tilde{d}_{1,k}(n) = \begin{cases} d_{1,k}(n) & , n > N \\ 1/(1-\alpha_1) & , n = N \\ 0 & , n < N \end{cases} \tag{66}$$

where $$\alpha_1 = \Pr\left\{D_{1,k} > N\right\}. \tag{67}$$

Intuitively, $\tilde{D}_{1,k}$ is the delay of a class 1 job, conditioned on the total time being greater than N, which is when job 2 arrives. Hence, this conditional delay also defines the waiting time for the following class 2 job, i.e.

$$w_{2,k}(n) = \tilde{d}_{1,k}(n-N) \tag{68}$$

For all other classes $i = 2,...,I$, define the following distributions in the same way:

$$\tilde{d}_{i,k}(n) = \begin{cases} d_{i,k}(n) & ,n > iN \\ 1/(1-\alpha_i) & ,n = iN \\ 0 & ,n < iN \end{cases} \tag{69}$$

where $$\alpha_i = \Pr\left\{D_{i,k} > N\right\} \tag{70}$$

and $$w_{i,k}(n) = \tilde{d}_{i-1,k}(n-N). \tag{71}$$

The final result $w_{1,k}(n) = \tilde{d}_{I,k}(n-N)$ defining a recursive relationship for $w_{1,k}(n)$. In fact, the vacation time $v_{i,k}(n)$ can also be expressed by its transform:

$$H_{i,k}(z) = \frac{\Lambda_{i-1,k}(z)}{z^{(I-1)N}\Psi_{i,k}(z)G_{i,k}(z)} \tag{72}$$

where $\Lambda_{i-1,k}(z)$ and $\Psi_{i,k}(z)$ are the z-transforms of $d_{i-1,k}(n)$ and $w_{i,k}(n)$. Because we are mainly interested in the probability that the waiting time exceeds some time t, we refer to [35, 36] for the waiting time tail approximation given below:

$$\Pr\{W_{i,k} > t\} = \rho_k \exp\left(-\frac{\rho_k t}{\mathrm{E}\left[W_{i,k}\right]}\right), \tag{73}$$

where $$\rho_k = \sum_{i=1}^{I} \frac{E[S_{i,k}]}{N} \tag{74}$$

is the average load on the system.

The waiting time tail approximation can be derived based on the approach shown in Table 6, which iterates through equations (66) and (69) until the expected waiting time converges. The distributions are truncated at a sample size $N_{\max}$ under which the expected waiting time will be accurate. Because the waiting time tail distribution is truncated by $N_{\max}$, we use the approximation in (73) to estimate the waiting time tail distribution. The complexity of the waiting time estimation is proportional to $I \cdot N_{\max} \cdot N \cdot iter$, where I is again the total number of cyclic classes, and $iter$ is the number of iterations through the algorithm in Table 6.

Table 6 Obtaining the waiting time distribution and probability of violating a delay deadline.

```
1.  Initialize $W_{1,k}^{(0)} = 0$, $iter = 0$.
2.  Do {
3.     Calculate $\tilde{s}_{2,k}^{(iter)}(n)$, $\tilde{s}_{2,k}^{(iter)}(n)$, …, $\tilde{s}_{2,k}^{(iter)}(n)$ from (66) and (69), $n \le N_{\max}$.
4.     Set $w_{i,k}^{(iter)}(n) = \tilde{s}_{i-1,k}^{(iter)}(n)$ for $i = 2,...,I$.
5.  } while ($\left|E[W_{i,k}^{(iter)}] - E[W_{i,k}^{(iter-1)}]\right| > \delta$);
6.  Apply (75) to $E[W_{i,k}^{(iter)}]$ to determine tolerable delay under error prob. $\varepsilon_i$.
```

The last step of the algorithm in Table 6 is used to estimate waiting time tail distributions. However, the tail distribution of $D_{i,k}$ is of greater importance to our system, since the delay determines the probability that a hard deadline at time $t + T_i$ is missed for a job arriving at time t. However, since the tail distribution for $W_{i+1,k}$ has the same shape as the tail distribution of $D_{i,k}$ shifted by $-N$ in time (as can be shown from the relationship in (71)), we can apply the waiting time tail approximation in (73) to estimate the probability of violating delay deadline T_i:

$$\Pr\{D_{i,k} > T_i\} = \Pr\{W_{i+1,k} > T_i - N\} \approx \rho_k \exp\left(-\frac{\rho_k (T_i - N)}{\mathrm{E}[W_{i+1,k}]}\right) \tag{75}$$

3.5.2 Queuing-based DVS Algorithms

The processor captures its service time and waiting time distributions for different classes of jobs at various power levels P_k according to the algorithm described in section III.3.5.1. The decoding system then produces a lookup table containing the net load ρ_k and expected delay $E[D_{i,k}]$ for each class i. Using (75), the processor can quickly estimate the probability of violating delay deadlines T_i. Finally, the video application sets upper bounds ε_i for the probability of dropping class i jobs, and determines a power schedule such that $\Pr\{D_{i,k} > T_i\} < \varepsilon_i$. Note that ε_i can be set to vary depending on the distortion impact of the respective job class for a particular sequence. For example, if an MPEG sequence has high motion, B-frames will have a larger overall quality impact on the video, and hence the probability of missing its deadline ε_i should be decreased.

As mentioned in [64], for a fixed time τ to complete c cycles, the optimal energy saving

Optimization Problem 1: Minimize a fixed power level subject to a target miss rate.

$$\begin{aligned} &\min_k P_k \\ &s.t. \Pr\{D_{i,k} > T_i\} < \varepsilon_i \end{aligned} \tag{76}$$

schedule is to run the processor at a minimal constant speed, which is $f* = \frac{c}{\tau}$. For discrete frequency levels, $f*$ can be achieved through timesharing between two adjacent frequency levels. While the delay constraints do not allow the processor to always run at the optimal average power level, the processor may usually run at a constant power level, and occasionally increase to higher power levels when the queue size (and aggregate delay) becomes large, such that jobs need to be processed quickly. Hence we consider the following low complexity optimization problem for selecting processor frequencies:

We proposed two variations of an adaptive algorithm that performs power adaptation to achieve energy savings while meeting deadlines with high probability. Based on control parameter α, Algorithm 1 below adjusts its power such that the decoding deadlines for all classes will be met with high probability. Hence, it solves (1) after processing each job. Algorithm 2, on the other hand, adjusts the power based only on the class about to be processed, such that that particular class i will meet its decoding deadline with probability $1-\varepsilon_i$. Unlike existing algorithms, such as laEDF, where the complexity grows linearly with the number of released jobs, both queuing-based algorithms have significantly lower complexity, where Algorithm 1 has complexity equal to the number of classes of jobs, while Algorithm 2 has constant complexity.

Algorithm 1 Adjusting power for all classes based on service time overshoot or undershoot

```
1.  Solve problem 2.
2.  While jobs are available,
3.     Set time $t$ to $0$ for each (soft) arrival point.
4.     If job $i$ finishes at time $t = \tau + \Delta\tau$
5.        Change delay bounds for all classes $j$ to $\Pr\{D_{j,k} > T_j - \alpha\Delta\tau\} < \varepsilon_j$, $0 \le \alpha \le 1$
6.        Solve problem 2 under new constraints.
7.     end
8.  end
```

Algorithm 2 Adjusting power per class based on service time overshoot or undershoot

```
1.  Solve problem 2. Set $P_{k_1} = P_{k^*}$ for job 1, where $P_{k^*}$ is solution to problem 2.
2.  While jobs are available,
3.     Set time $t$ to $0$ for each (soft) arrival point.
4.     If job $i$ finishes at time $t = \tau + \Delta\tau$
5.        Change delay bound for class $i+1$ to $\Pr\{D_{i+1,k_i} > T_{i+1} - \alpha\Delta\tau\} < \varepsilon_i$, $0 \le \alpha \le 1$
6.        Find $P_{k_{i+1}}$ by solving problem 2 for class $i+1$ with only the $i+1th$
7.     end
8.  end
```

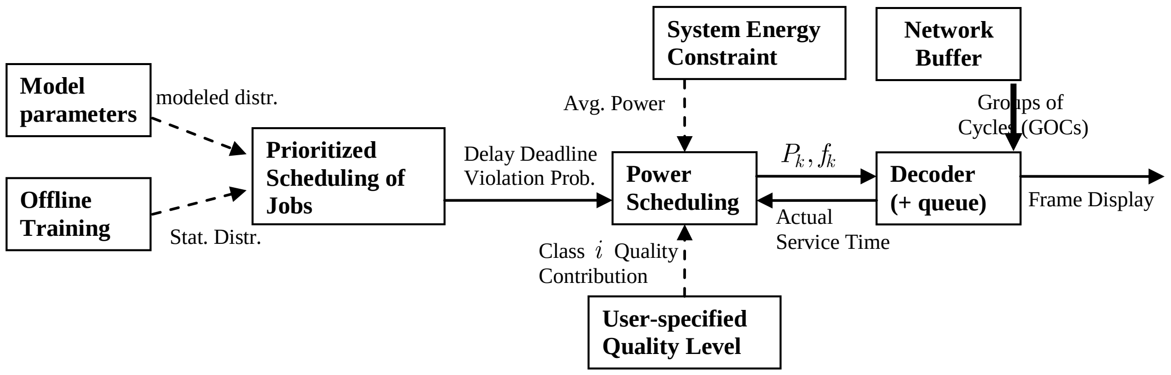
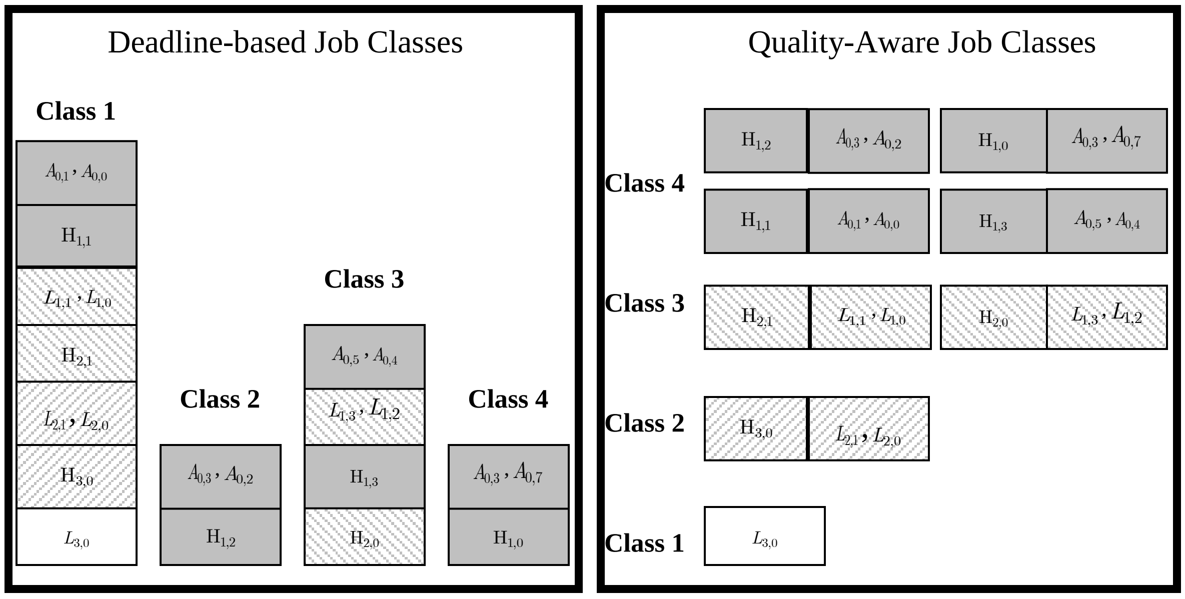


Figure 21: Queuing model for Priority Scheduling based DVS.



(a) (b)

Figure 22: (a) Job decomposition based on deadlines for 3 level MCTF. (b) Job decomposition based on quality-aware priority classes.

3.6 Quality-aware Priority Scheduling-based DVS

In this section, we introduce the concept of a joint DVS and priority-based job scheduling algorithm (Figure 21). By dropping less important jobs, we can gracefully degrade the quality by operating at lower power levels. We will show examples of this graceful degradation in our results section.

3.6.1 Incoming Traffic Model and Service Model

Unlike the deadline-driven decomposition of jobs, we now consider jobs that are decomposed based on their contribution to quality. Figure 22 shows an example of how jobs can be decomposed into wavelet frames. There are also many other levels of decompositions, such as subbands, macroblocks, etc. In particular, the MCTF wavelet coder has the property of sorting jobs into priority classes, where a job that contributes more to quality belongs to a higher priority class.

Consider again a buffer that releases jobs for decoding. By lightly varying the way the individual job cycles are streamed to the decoder, we can derive a model where ED complexity arrives in groups of cycles (GOCs) of fixed size according to a mixed arrival process. Because ED complexity is closely related to the arrival rate of bits, ED GOCs can be seen as a complexity representation of packets. Below, based on some simplifying assumptions, we will show that ED complexity can be modeled by GOCs that arrive according to a Poisson arrival process plus a general arrival process.

Proposition 6: Let us assume:

- Jobs of class i arrive in periodic time intervals of size τ_i (as in Section 3.3).
- The bit-rate r_i per job of each class i is quasi-constant.
- The shifted and scaled Poisson distribution for ED complexity per frame holds.
- Different temporal level transform frames corresponding to the same job have independent ED complexity statistics (as a result of motion compensation).
- The buffer can feed the decoder with bits that arrive according to an arbitrary distribution, as long as the total number of bits that arrive within time τ_i is r_i.

Then ED complexity can be modeled by a Poisson arrival process plus a general arrival process.

Proof: It is a well known fact that a Poisson arrival process with i.i.d. exponential interarrival times $a(t) = \lambda_i e^{-\lambda_i t}$ can be decomposed into a doubly stochastic model based on a Poisson distributed number of arrivals within a fixed time period τ, and a uniform distribution of arrivals within that period [71], i.e.

$$N_A \sim p_A(n) = \frac{(\lambda_i \tau_i)^n e^{-\lambda_i \tau_i}}{n!} \ , n \geq 0 \tag{77}$$

$$\bar{U}_{N_A} = (u_{1,N_A}, u_{2,N_A}, ..., u_{N_A,N_A}) \sim f(\bar{U}_{N_A}) = \frac{1}{\tau^{N_A}} \ , 0 \leq u_{N,j} < \tau, j = 1,...,N \tag{78}$$

where λ is the arrival rate of the process, N is the random number of arrivals within the period, and $\bar{U}_N$ is the uniform distributed vector over an N_A-dimensional hypercube of all combinations of possible arrival times. (Note that components of $\bar{U}_{N_A}$ do not necessarily have to arrive in order.)

Now, consider the case where ED complexity for a frame follows a Poisson distribution with mean ν_{fr}. The buffer, which can arbitrarily stream bits that it contains for a given job, will stream in such a way that GOCs of size 1 (cycle) arrive at rate

ν_{fr} / τ_i uniformly distributed in time interval $[0, \tau_i]$. Hence, the ED complexity for the frame follows a Poisson arrival process. Since wavelet transformed frames are often independent, we may assume the ED GOCs associated with each frame or job to be independent. The sum of independent Poisson arrival processes is another Poisson arrival process, hence the ED complexity per each job, which may include entropy decoding several frames, is a Poisson process. Denote the average ED complexity per job of class i as ν_i. Based on the above construction per frame, the buffer now streams bits in a manner such that, based on the ED complexity associated with each bit, the ED complexity is "streamed" as a Poisson process in GOCs of size 1 (cycle) with arrival rate ν_i / τ_i with total job complexity:

$$C_{i,ED} \sim p_{i,ED}(n) = \frac{\nu_i^n e^{-\nu_i}}{n!}, n \geq 0 \tag{79}$$

Now consider $C_{i,ED} = a_i \hat{C}_{i,ED} + b_i$, where $\hat{C}_{i,ED}$ follows the distribution in (79). First, $a_i \hat{C}_{i,ED}$ can be modeled as a Poisson process as above, but with GOCs of size a_i. Secondly, we can model the complexity b_i as arriving across an independent complexity stream, where exactly b_i / a_i GOCs of size a_i arrive within time τ_i. Hence, the total arrival process is a mixture of Poisson and a general arrival process. ■

For the remainder of this section, we simplify the model for ED complexity to be a pure Poisson arrival process. We use $\lambda_1, \lambda_2, ..., \lambda_I$ to denote the quantized ED arrival rate for priority classes $1, ..., I$, and $\lambda = \sum_{i=1}^{I} \lambda_i$ is the total job arrival rate.

3.6.2 Proposed DVS Service Policy and Model

Based on the decomposition of jobs into ED GOCs in section 3.6.1, we propose a DVS system that uses priority scheduling to process the incoming GOCs. We model the service of GOCs as a non-preemptive $M / G / 1$ priority queuing system. In other words, whenever the system finishes servicing a GOC, it will then process the highest priority GOC waiting in the queue at the time. However, while a GOC is being processed, a GOC of higher priority can not interrupt the ongoing service (i.e. non-preemptive).

Based on priority scheduling, the system will ensure that even if not all jobs can be processed before the display deadline, the higher priority jobs will be processed first, so that they are more likely to satisfy their deadline constraints. Effectively, a lower quality video can be streamed by decoding (in time) only jobs in higher priority classes without having to process jobs from every priority class. This creates a quality and energy tradeoff, as we can lower the average processor power to create a video of lower quality. To model the service rate per GOC, we divided the total complexity associated with the decoding of each frame by the complexity of entropy decoding.

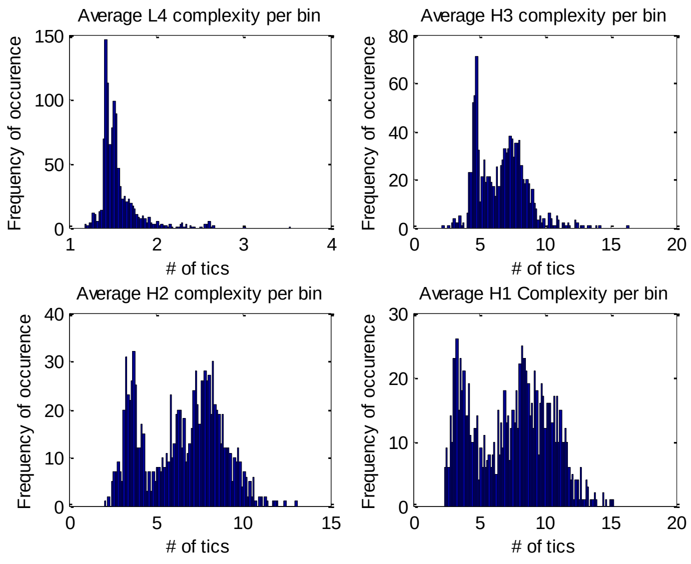


Figure 23 Example of total complexity per arriving ED GOC for various frames in a 4 temporal level MCTF GOP. The statistics are averaged over several sequences.

3.6.3 Delay and Idle Time Analysis

Let $D_{i,k}$ be the delay of processing a GOC of class i, and define $\Pr\{D_{i,k} > T_i\}$ to be the probability that a GOC arriving at time t can not be processed before deadline $t + T_i$. Note that in reality, all GOCs of the same job have the same hard deadline regardless of their arrival times t, so the delay bound T_i would not be fixed for every GOC of a job. However, considering that GOCs need to be processed in FIFO order to complete the job, the deadlines for the first GOCs in the job may be set earlier to accommodate the processing time delay induced on later GOCs. For the purpose of analysis, we approximate the delays tolerated by all GOCs within a class to be approximately equal.

In order to determine the probability of violating the delay deadline for a non-preemptive priority queuing system, we first define the load on the system induced by priority class i with service time $S_{i,k}$ as:

$$\rho_{i,k} = \lambda_i E[S_{i,k}] \tag{80}$$

Let $\sigma_{i,k} = \sum_{j=1}^{i} \rho_{j,k}$ be the total load of traffic coming from priority classes 1 to i, and let $\mu_{i,k}$ be the average service rate for a class i job in processor operating mode k. The average waiting time in the queue for priority class i GOCs can then be expressed as [73]:

$$\mathrm{E}[W_{i,k}] = \frac{\sum_{j=1}^{I} \frac{\rho_{j,k}}{\mu_{j,k}}}{2(1-\sigma_{i-1,k})(1-\sigma_{i,k})} \tag{81}$$

From the average waiting time, we can obtain an approximation for the probability that the waiting time exceeds some time t. We use the waiting time tail approximation to estimate the tail of the delay:

$$\Pr\{D_{i,k} > T_i\} = \Pr\{W_{i,k} + S_{i,k} > T_i\} \approx \rho_k \exp\left(-\frac{\rho_k T_i}{\mathrm{E}[W_{i,k}] + E[S_{i,k}]}\right) \tag{82}$$

The fraction of idle time in an $M/G/1$ queuing system is the time average probability that the system is empty:

$$p_{0,k} = 1 - \rho_k \tag{83}$$

3.6.4 Priority Scheduling Optimization Problems and Algorithms

In this section, we formulate and analyze a number of optimization problems based on probabilistic delay constraints. We begin with a simple optimization problem, where a processor continues running an idle processing thread even if there are no jobs in system:

Optimization Problem 2: Minimize the Average Active Power given an Average Video Quality

$$\min_{\alpha=(\alpha_1,\dots,\alpha_K)} \sum_{k=1}^{K} \alpha_k P_k$$
$$s.t. \sum_{k=1}^{K} \alpha_k \sum_{i=1}^{I} \frac{\lambda_i}{\lambda} \Delta_i \Pr\{D_{i,k} \le t_i\} \ge Q_{\text{avg}} \tag{84}$$
$$\sum_{k=1}^{K} \alpha_k = 1$$

where

$$Q_k = \sum_{i=1}^{I} \frac{\lambda_i}{\lambda} \Delta_i \Pr\{D_{i,k} \le t_i\} \tag{85}$$

is the average quality of the decoded sequence at power level P_k. Here, α is a vector with components that are the fraction of time the processor is set to operate at power level P_k, and Δ_i is the quality slope parameter for priority i GOCs (i.e. the average quality contributed to video by a priority i GOC.) as introduced in [66]. Note that $\frac{\lambda_i}{\lambda}$ is the fraction of GOCs of priority i received from the bitstream. Thus, the first constraint requires that the average quality of the video is at least Q_{avg}. This problem turns out to be a linear programming problem, since P_k and Q_k are constants. We can thus solve this via the simplex method. However, an even simpler closed-form solution exists if we explicitly consider the properties of power with respect to quality.

Proposition 7: If quality is a concave increasing function of ED complexity, and there are a finite number of power/frequency levels, the optimal solution to is to run the processor always at a single power level, or to perform time sharing between two adjacent power levels.

Proof: Let $\hat{Q}$ be a discrete random variable which takes on quality levels Q_k with probability α_k. Power is a convex function of frequency [64]. Likewise, complexity (and thus the average processor frequency per unit time) is a convex function of quality, which has been shown theoretically [25, 26] given a concave PSNR curve with respect to rate. Hence, the power is a convex function of the required average quality. From [72], it is shown that for a convex quality to power function $P(q)$, the distribution of $\hat{Q}$ with $E\left[\hat{Q}\right] = Q_{\text{avg}}$ that minimizes the expected value of the function $E\left[P(\widehat{Q})\right]$ is to choose the $\widehat{Q} = Q_{\text{avg}}$ with probability 1 if $P\left(Q_{\text{avg}}\right) \in \{P_1,\ldots,P_K\}$, or else:

$$\widehat{Q} = \begin{cases} Q_{k^*} & w.p.\ \frac{Q_{\text{avg}} - Q_{k^*}}{Q_{k^*+1} - Q_{k^*}} \\ Q_{k^*+1} & w.p.\ \frac{Q_{k^*+1} - Q_{\text{avg}}}{Q_{k^*+1} - Q_{k^*}} \end{cases} \tag{86}$$

where $Q_{k^*} < Q_{\text{avg}} < Q_{k^*+1}$. $\widehat{Q}$ then minimizes $E\left[P(\widehat{Q})\right]$, which gives us the solutions α_k to . ■

If we now consider the case where the processor may shut down during idle times and expend essentially zero energy, we have a quality-constrained, energy-minimizing

Optimization Problem 3: Minimize the Average Power given an Average Video Quality

$$\begin{aligned} &\min \sum_{k=1}^{K} \alpha_k \rho_k P_k \\ &s.t. \sum_{k=1}^{K} \alpha_k \sum_{i=1}^{I} \frac{\lambda_i}{\lambda} \Delta_i \Pr\{D_{i,k} \le t_i\} \ge Q_{\text{avg}} \\ &\sum_{k=1}^{K} \alpha_k = 1 \end{aligned} \tag{87}$$

problem:

This problem is the same as (84) but under a different objective function which is not necessarily convex. Since the optimal mode of operation should keep the system nonempty with high probability, such that the processor power should run at a nearly constant power level [64], we propose a simplified problem that can be solved with complexity $O(I \cdot \log K)$ and can be used by a DVS algorithm to reactively adapt the

Problem 4: Minimize a Fixed Power given an Average Quality

$$\begin{aligned} &\min P_k \\ &s.t. Q_k \ge Q_{\text{avg}} \end{aligned} \tag{88}$$

power level based on a minimum desired average quality:

Based on Problem 4, we propose several simple priority scheduling and power scheduling algorithms for DVS. The first algorithm chooses a constant power based on the arrival rate and service time statistics by solving with various levels of Q_{avg}. The second algorithm is the same as the first, but periodically purges the queue of expired jobs, thereby reducing the average waiting time for different classes. Finally, we present a combined quality-aware priority and look-ahead algorithm (Algorithm 3), which temporarily increases the power whenever important jobs are about to expire. Whenever a job in a class i is within δ seconds of being expired, the system will increase the processor power according to the job's priority by some $\Upsilon(i)$, thereby increasing the chance of that job being decoded in time.

Algorithm 3 Priority scheduling with last second power increase

```
1.  Solve problem 4 for $Q_{\text{avg}}$ to obtain $P_{init}$.
2.  While jobs are available,
3.     For the highest priority class $i$,
          s.t. deadline of a job in class $i$ expires in less than $\delta$ time
4.           Set $P = P_{init} + \Upsilon(i)$.
5.     end
6.     Process highest priority job in FIFO order. Record service time $s$
7.     Subtract deadline of all other jobs by $s$.
8.     If deadline of a job $j$ is less than 0, purge job $j$.
9.  end
```

3.7 Experimentation and Results

3.7.1 Experimental Setup

For our experiments, we adopted the power and frequency model for the 70nm technique node in [78][79]. We considered discrete voltage levels V_{dd} between 0.6V and 1.0V with voltage step sizes of 0.1V, and present the clock frequencies and power for different V_{dd} levels in table 6.1.

Table 6.1. Frequency and power for different V_{dd} levels

Vdd (V)	0.6	0.7	0.8	0.9	1.0
Frequency (GHz)	0.79	1.27	1.81	2.42	3.09
Dynamic Power (10^{-5}W)	0.12	0.27	0.50	0.84	1.33
Leakage Power (10^{-5}W)	0.21	0.29	0.40	0.54	0.72
Total Power (10^{-5}W)	0.33	0.56	0.90	1.38	2.05

For our application scenario, we combined 11 video sequences of 16 GOPs each (e.g. *Coastguard*, *Foreman*, *Harbor*, *Mobile*, *Silent*, *Stefan*, and several others) into a long sequence, which was then decoded. We set the hard deadlines for Algorithm 1 and Algorithm 2 to be 8 frames after the (soft) periodic arrivals, and we collected workload traces for 4 temporal level MCTF sequences [3]. We measured the complexity for each decoding job in terms of clock cycles and use the measurement for offline scheduling. We also tuned the stochastic model using the measurement for the proposed online algorithm SLP/r.

Furthermore, we generated more data for simulation based on Monte Carlo method to present a more general simulation. The experiment observation was almost the same as the result from real data. Hence, we only present the result from the real data here.

To simulate a real-time video decoding environment with sequences that have a frame rate of 30Hz, we fix the hard display deadlines. We assume that the (soft) frame arrivals from the network follow a normal distribution as in [83] to simulate a wireless network, and we applied the same generated arrival times of jobs for all algorithms in our experiments. For all algorithms, we calculate energy with the same power model considering leakage power. Since the absolute value of energy is not important, we report normalized energy with respect to energy consumed by the optimal solution.

3.7.2 Optimality Study

We chose to compare the performances of the rLP, the queuing-based algorithm 2, and laEDF DVS [41]. We selected queuing-algorithm 2 for comparison as it outperforms algorithm 1 experimentally. To be fair, the laEDF DVS also considers separate WCET estimates for different classes of jobs, and hence it is not entirely application-agnostic. We then compare the performances of different priority scheduling algorithms and discuss the quality-energy adaptation points achieved by Algorithm 3. We also revised the power models of laEDF [41] to consider leakage power and sleep modes for a fair comparison.

The parameters of each algorithm were tuned to obtain different trade-off points for energy consumption and job miss rate. For the queuing based algorithm, we tuned the delay sensitivity parameter ε, and for laEDF, we used different WCETs. The result is shown in Figure 24. The energy achieved by the optimal offline LP solution (e.g. the lower bound) is normalized to 1. Note that based on our formulation, the optimal solution always has zero miss rate. The result shows that for a zero miss rate, laEDF consumes approximately 15% more than the optimal and queuing based algorithm 2 consumes approximately 4% more than optimal.

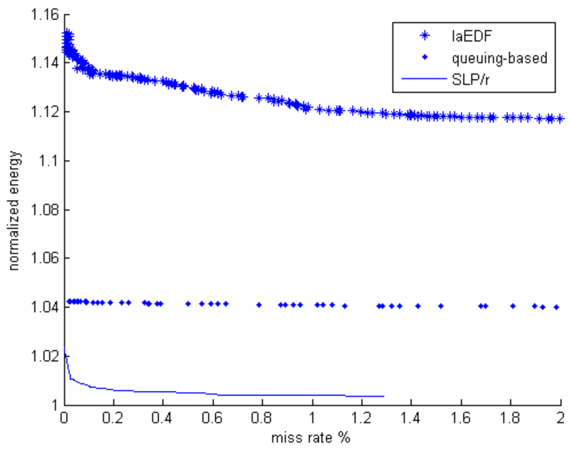


Figure 24 Energy and miss rate

3.7.3 Quality-aware Priority Scheduling DVS Implementation and Results

For the priority scheduling approach, we decomposed jobs of the MCTF GOP structure in the same way shown in Figure 20. While there are other ways to prioritize jobs based on different classes for single frames, for resolution levels within frames, or even for subbands within resolution levels in order to achieve results with finer granularity, the priority classification method depicted in Figure 20 was sufficient to highlight the performance of our priority-driven DVS implementation.

Based on various average power levels for the processor, we compared the probability of dropping jobs of different classes based on the strict priority scheduling policy, the periodic queue purging priority policy, and our DVS strategy in Algorithm 3. Table 7 includes the results from the combined sequence encoded by 4 temporal level MCTF based on the 5 different operating frequencies $f_1,...,f_5$. For Algorithm 3, we used $\Upsilon(1) = f_{k+2}, \Upsilon(2) = f_{k+1}$. While Algorithm 3 may expend slightly more power than pure priority queuing due to speeding up when jobs are about to expire, it performs better than pure priority queuing due to reactively rushing jobs through at the last minute. In the case of job classes being frames or groups of frames, the effect of dropping different priority classes is the same as reducing the frame rate of the corresponding GOP. Finally, Table 8 shows how different power levels correspond to different frame rates, energies, and quality levels. Notice that as long as the frame rate is sufficiently high (e.g. 10fps), there is only a loss in quality of less than 1.5 dB when the power is scaled down to 10%! Likewise, when the energy is scaled down to 25%, the quality degradation is less than 1.0 dB, which demonstrates that Algorithm 3 achieves high-scalability in terms of quality and power tradeoffs.

Table 7 Comparisons of performances of various quality-aware priority scheduling algorithms in terms of the percentage of deadlines met for various priority classes for 4 temporal level decomposition (f_0 indicates the minimum processor power.).

Jobs decoded in time (%)	Frequency Level	f_0	f_1	$3f_0$	$4f_0$	$5f_0$
Priority Scheduling	class 1	99.72	100	100	100	100
	class 2	67.33	99.91	100	100	100
	class 3	0	98.48	99.91	100	100
	class 4	0	0	0	99.05	99.95
	class 5	0	0	0	0	0
Priority Scheduling with Queue Purging	class 1	99.62	100	100	100	100
	class 2	68.18	99.91	100	100	100
	class 3	13.54	99.72	100	100	100
	class 4	0	14.32	57.50	99.67	99.95
	class 5	0	0	6.46	26.95	41.28
Priority Scheduling with Last Second Power Increase for 3 Priority Classes	class 1	99.91	100	100	100	100
	class 2	91.29	99.91	100	100	100
	class 3	31.91	99.43	100	100	100
	class 4	0	14.02	58.43	99.67	99.95
	class 5	0	0	6.63	26.95	41.28

Table 8 Comparisons of average energy consumption and quality levels for algorithm 2 and quality-energy adaptation points of algorithm 3 for the *Coastguard* and *Stefan* sequences decoded a bit rate 768kbps. The average frame rates (frames per second) over all GOPs are given for different adaptation points for algorithm 3.

	Frame rate (fps):		**Energy consumed:**		**PSNR (dB):**	
Algorithm	***Coastguard***	***Stefan***	***Coastguard***	***Stefan***	***Coastguard***	***Stefan***
2	30.00	30.00	2.63E	2.41E	33.24	27.35
3	26.48	23.67	2.15E	2.15E	32.98	27.01
3	20.04	18.05	1.26E	1.25E	32.51	26.70
3	16.17	15.23	0.65E	0.65E	32.23	26.48
3	14.53	10.08	0.28E	0.29E	32.05	25.94
3	8.09	7.27	0.09E	0.09E	30.68	25.55
3	5.04	3.63	0.02E	0.03E	29.44	24.62

3.8 Conclusions

Current multimedia compression algorithms and standards provide only very coarse levels of complexity, thereby neglecting the vast resource diversity and heterogeneity of state-of-the-art systems. Also, current systems lack good complexity models for resource management, and hence statistics must be collected and updated frequently online in order to reactively adapt to time-varying source and coding structures. This chapter provides a (fine) granular complexity model that enables systems to plan ahead and optimize their scheduling algorithms. We demonstrate that significant energy savings can be achieved through application-specific stochastic modeling of complexity by proposing two DVS approaches that achieve near optimal performance for video decoding applications. Finally, we proposed an adaptive architecture combining both power and job scheduling to obtain energy-quality tradeoffs. Our results indicated that the priority-scheduling based DVS algorithms can save a significant amount of energy with only a small reduction to the quality level.

CHAPTER 4
Informationally-Decentralized System Resource Management for Multiple Multimedia Tasks

4.1 Introduction

With the advent of web TV, YouTube, peer-to-peer multimedia streaming, video conferencing, etc., multiple autonomous multimedia processing, compression, transcoding, and streaming tasks need to be executed simultaneously on the same system. Because multimedia systems must readily cope with time-varying resource availabilities and demands, an automated resource management solution for gathering on-the-fly application requirements, and reconfiguring the system in a timely manner, is highly desirable [85][91]. Moreover, multimedia applications are highly *resource-aware*, such that they can achieve different video/audio quality levels given different amounts of system resources (e.g. processing power, memory, battery power, bandwidth, etc.) [99][11]. Hence, the principles and methodology for determining how to *optimally* divide limited system resources among multiple multimedia applications must also be addressed [87].

To this extent, various analytical solutions have been proposed for allocating limited system resources to multiple resource-aware applications [87]-[91]. In these works, the quality of a resource allocation scheme is assessed by one of several *social welfare* functions, often characterized by a weighted sum or a weighted product of individual application utilities (or multimedia qualities) [86] [93]. In [87]-[89], centralized resource allocation solutions are provided for maximizing the social welfare of multiple applications that have multiple quality-of-service dimensions (e.g. error rate, delay, etc.), and consume multiple types of system resources. In [90], a centralized solution is provided for maximizing the social welfare of multiple applications running on a multiprocessor system. Note that centralized solutions implicitly require a *resource manager* (RM) to gather application utility-resource functions and solve a centralized optimization problem. To reduce the informational requirement at the RM, a decentralized, auction-based middleware solution called CARISMA is proposed for maximizing social welfare [91]. By maintaining a valid representation of the execution *context* (i.e. the types of resources required by the applications), the CARISMA middleware provides applications with the necessary information to bid on various system policies (or configurations). Hence, the middleware acts as an RM that simply collects bids from each application and chooses the policy that maximizes the sum of the bids.

However, the above approaches do not perform efficiently for multimedia applications due to several reasons. First, multimedia applications are diverse, and include algorithms such as coding, error concealment, resizing, deinterlacing, deblocking, and denoising, which can function at a fine-granular set of operating points based on various modes and parameters [99] [101] [102]. Hence, the RM must be either complex enough to map out and optimize over a large number of system configurations, or intelligent enough to choose only a subset of potentially optimal candidate policies based explicitly on the application's algorithm. Unfortunately, the computational overhead associated with enumerating possibly hundreds or thousands of policies, and choosing the optimal policy (through centralized optimization or decentralized bidding), is unsuitable for multimedia applications with highly *dynamic* utility functions, since the system would need to be reconfigured frequently during runtime [11]. On the other hand, designing an intelligent RM that can provide candidate policies based on multimedia algorithms is an even more daunting task, since the RM needs to be updated every time a new algorithm is ported to the system. While the complexity of determining candidate policies may not be an issue for a system that supports only 1-2 types of multimedia algorithms, systems nowadays need to support many different types of algorithms (e.g. see [92] for a list of common video codecs), and hence the size of the RM would need to be very large.

Secondly, and perhaps more importantly, multimedia applications that share a system are often developed by many different companies [92]. In many cases, these competing companies may have incentives to develop products that can exploit information from other applications in order to "selfishly" improve their own performances at the cost of social welfare. Hence, multimedia applications often require autonomy over the distribution of *private information*, such as details about their coding algorithms and utilities, in order to protect themselves from malicious applications (and a potentially exploitative system). Unfortunately, the algorithms proposed in [87]-[90] require multimedia applications to submit their utility-resource functions to the RM. Likewise, CARISMA's auction-based protocol requires each application to bid *according to its utility* for each candidate policy, such that each application would be required to report its utility to the middleware for possibly many different system configurations [91]. In summary, prior works do not offer *informationally*-decentralized solutions that can protect the private information of autonomous multimedia applications.

In this chapter, we present a low-complexity, resource management solution for allocating resources to multiple autonomous multimedia applications, without requiring the applications to share private information about their utilities. By employing a *continuous* representation of system resources, we introduce a *message exchange protocol* (MEP) that enables each application to perform its own reconfiguration and utility optimization in a decentralized manner,[2] while using a RM to communicate system resource demands and costs to each application. Importantly, our optimization framework allows the *decision making* process for resource allocation to be decentralized, where the RM updates costs solely based on resource availability and demand, while applications update resources requirements based on their private

[2] Note that many multimedia applications already have features built in to maximize their qualities subject to resource constraints [95]. Thus, our decentralized resource management solution requires little additional complexity provided that individual application utilities are optimized by the applications themselves.

utilities. Indeed, this design also allows the computational overhead of the algorithm to be distributed across possibly *remote* (networked) applications, which can significantly reduce the system's computational overhead. (For example, remote video encoders can make coding decisions for the corresponding decoders on the system in order to satisfy the system's computational resource bounds [11].) Our main contributions are summarized below:

- We introduce a formal, analytical framework for the MEP. We show that repeatedly exchanging messages between the RM and each application can effect optimal resource allocations in a decentralized manner without requiring applications to disclose private information about their utilities.
- We show that the MEP is highly versatile and can be used to implement informationally-decentralized algorithms that achieve a variety of system objectives, such as maximizing the social welfare of applications, minimizing system energy consumption, performing workload balancing, and performing joint power scheduling for interdependent multimedia jobs.
- Based on a derived stochastic model for dynamic video quality functions, we provide insight into the rate of adaptation for MEP-based decentralized algorithms in dynamic environments. We also demonstrate how temporal correlations in the quality functions of multimedia applications can be exploited by our algorithms to reduce the complexity of resource allocation. We show experimentally that our proposed algorithms converge quickly to their respective optimal solutions and are therefore ideal for resource adaptation in dynamic environments.

The organization of this chapter is as follows: Section 4.2 provides an overview of the system, and introduces the MEP. In Section 4.3, we provide MEP-based algorithms for jointly maximizing social welfare and minimizing energy consumption. Section 4.4 provides MEP-based algorithms for achieving several other miscellaneous objectives. Section 4.5 analyzes how parameters in the MEP can be tuned to adapt quickly to time-varying, but temporally-correlated video quality functions. Section 4.6 provides simulations that compare the performances of the algorithms, and Section 4.7 concludes the chapter.

4.2 The Message Exchange Protocol

In this section, we introduce the analytical framework for the MEP that is used to communicate information between applications and the system. An example of the MEP for maximizing social welfare on a single processor system is then provided as an illustration.

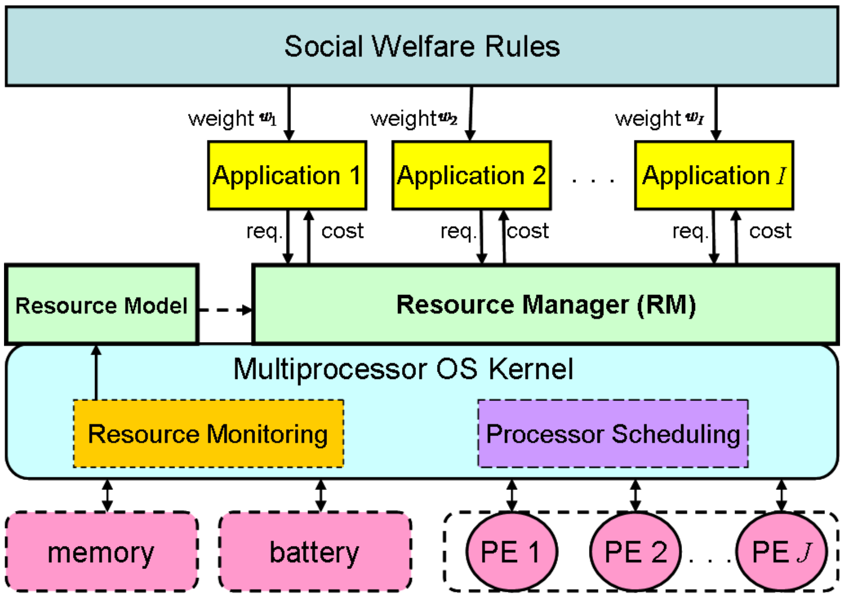


Figure 25 An overview of the system stack, with the resource manager and resource model.

4.2.1 Formal Representation of the Message Exchange Protocol

The focus of this chapter is the design of a resource management solution that can provide optimal resource allocation schemes for multiple multimedia applications in an informationally-decentralized environment. (See Figure 25 for a high-level view of the architecture.). The OS kernel, which can accurately monitor system resources, does not have knowledge of each application's quality-resource function, and hence cannot schedule resources to maximize social welfare-related metrics (e.g. sum of application utilities). On the other hand, the applications are not aware of system configurations and resource costs/constraints, or of other applications that may be sharing the system. The problem is further exacerbated by the autonomous applications which are, in general, unwilling to communicate private information about their utilities to the system, or to other competing applications.

To bridge the informational gap between the OS and applications while meeting information privacy requirements, we introduce a message exchange protocol that makes use of a system resource manager (RM) to pass messages between applications and the OS. While the RM does not have knowledge of application utilities, it makes use of a continuous model of system resources to intelligently effect various resource allocation solutions by charging applications various *costs* for consuming system resources. The RM generates costs in the form of *tax functions,* which capture the congestion levels or consumption rates of various utilized system resources (e.g. CPU utilization, communications bandwidth, energy availability, etc.). Based on the tax functions provided by the RM, multimedia applications can factor the costs into their utilities, reconfigure their algorithms, and update their resource requirements, which effectively leads to a new system configuration. The message exchange protocol (MEP) is presented formally as follows:

Stage 1: Initialization.

The RM provides each task i, $i \in \{1,2,...,I\}$, with a parametrized tax function $t_i(x_i, m_{-i})$, where x_i is task i's resource demand, and m_{-i} is a parameter (or *message*) that is transmitted and updated by the RM.[3] The message can be used to communicate the system resource condition, such as CPU utilization, system energy consumption, bandwidth availability, etc. Upon initialization, the RM also transmits an initial message $m_{-i}^{(0)}$.

In general, it makes sense to define tax functions with $t_i(0, m_{-i}) = 0$. In other words, a task does not have to pay if it chooses not to utilize system resources.

Stage 2: Message Exchange.

During the message exchange phase, the RM and the tasks will iteratively perform optimizations and message exchanges (as shown below) until the resource allocation

[3] The tax charged to each task may be in the form of real money, or of tokens for future system usage.

scheme converges.

Tasks to RM: During iteration n, each task i solves the following local optimization problem:

$$x_i^{\langle n \rangle} = \arg\max_x \left\{ Q_i(x) - t_i\left(x, m_{-i}^{(n-1)}\right) \right\}, \tag{89}$$

where the *net utility* function of task i is the task quality function $Q_i(x)$ minus the tax $t_i\left(x, m_{-i}^{(n-1)}\right)$ that the system charges to the task. After computing its new resource demand $x_i^{\langle n \rangle}$, it submits $x_i^{\langle n \rangle}$ to the RM.

RM to Tasks: Based on the resource demands from all tasks, the RM relays messages $m_{-i}^{\langle n \rangle} = f_i\left(\mathbf{x}_{-i}^{\langle n \rangle}, \gamma^{\langle n \rangle}\right)$ back to each task i, where $\mathbf{x}_{-i}^{\langle n \rangle}$ is a vector of the resource demands from all other tasks during iteration n, $\gamma^{\langle n \rangle}$ is a step size parameter that affects the value of the message at every iteration, and f_i is a function that is designed in a way to provide sufficient information to each task while minimizing the amount of information exchanged (i.e. the RM may not need to transmit the entire demand vector $\mathbf{x}_{-i}^{\langle n \rangle}$ and parameter $\gamma^{\langle n \rangle}$ to each task i). Minimizing the message size can be beneficial both in terms of reducing the informational overhead, as well as better protecting information about individual applications' resource demands. The messages $m_{-i}^{\langle n \rangle}$ will be discussed in more detail in Section 4.3.

Stage 3: Task Resource Allocation.

For static environments, *convergence* of the MEP to an optimal resource allocation solution $\mathbf{x}^* = (x_1^*, \ldots, x_I^*)$ is highly desirable. Formally, convergence can be defined as follows: For any $\varepsilon > 0$, there exists an integer N such that the resource allocation vector at any iteration $n \geq N$, given by $\mathbf{x}^{\langle n \rangle}$, satisfies the property $\left\| \mathbf{x}^* - \mathbf{x}^{\langle n \rangle} \right\|_2 < \varepsilon$ [100]. In practice, an approximate convergence point can be chosen for any given ε by terminating the algorithm when the change in the resource demand vector is less than ε, i.e. $\left\| \mathbf{x}^{\langle n \rangle} - \mathbf{x}^{(n-1)} \right\|_2 < \varepsilon$. After terminating the MEP, each task i is allocated $x_i^{\langle n \rangle}$ resources.

For dynamic environments on the other hand, a highly desired objective is minimizing the *adaptation time*. The adaptation time can be defined as the number of iterations for the algorithm to provide a solution within ε of the optimal resource allocation, given that the application utility functions are time-varying. Importantly, unlike the convergence time in a static environment, the algorithm must constantly adapt to different application utility functions over time. However, the RM can also take advantage of temporal correlations in the utility functions to reduce the adaptation time (discussed in more detail in Section 4.5).

We note that in general, MEP-based algorithms will require *multiple* iterations to converge due to a lack of centralized utility information. This is unlike the auction-based protocol in CARISMA [91], where the RM collects application utilities as sealed bids, and then chooses the optimal policy in a single iteration. However, because multimedia quality-resource functions are often concave [99], the MEP often requires each multimedia application to solve only a low-complexity, convex optimization problem during each iteration [100]. Hence, as long as only a few iterations are required for convergence, the algorithm has very low complexity compared to bidding on hundreds or thousands of policies.

4.2.2 An Illustrative Example of MEP Implementation: The Social Welfare Maximizing Algorithm

To illustrate the MEP protocol, consider the simple problem of social welfare maximization for multiple video applications on a single processor system, subject to CPU utilization constraints. For simplicity, we consider a static environment where application utility functions are fixed (Dynamic functions will be explored in Section 4.5). We express the social welfare function as a weighted product of the video distortions $d_i(x_i)$ (or errors) achieved under allocated computational resources x_i for each task i [87] [88]:

$$U(\mathbf{x}) = \prod_{i=1}^{I} \left(\frac{1}{d_i(x_i)} \right)^{w_i}, \tag{90}$$

or a weighted sum form of the dB PSNR qualities $Q_i(x_i)$ [91] [93] [94]:

$$U(\mathbf{x}) = \sum_{i=1}^{I} w_i Q_i(x_i), \tag{91}$$

where w_i are weights (priorities) specifying the importance of each task. For illustration purposes, we set $w_i = 1$ (equal priorities), such that the social welfare optimization problem is:

$$\begin{aligned}&\max \sum_{i=1}^{I} Q_i\left(x_i\right)\\&\text{s.t.} \sum_{i=1}^{I} x_i \leq R\ ,\\&x_i \geq 0, \forall i \in I\end{aligned} \tag{92}$$

where R indicates the total computational resources available (i.e. the schedulability constraint). The social welfare-maximizing (SWM) algorithm is given as follows:

Stage 1: Initialization.

Each task i is provided a linear tax function $t_i\left(x_i, m_{-i}\right) = m_{-i}x_i = px_i$, where p is a global cost per unit resource (e.g. cost per CPU cycle) charged to each task. To initialize the algorithm, the RM determines an initial cost $m_{-i}^{(0)} = p^{(0)}$ for each task.

Stage 2: Message Exchange.

Tasks to RM: During each iteration n, each task $i \in \{1,2,...,I\}$ calculates its new resource demand $x_i^{(n)}$ and submits it to the RM. The calculation is based on:

$$x_i^{(n)} = \arg\max_{x \geq 0} \left\{Q_i(x) - p^{(n-1)}x\right\}. \tag{93}$$

where $p^{(n-1)}$ is the cost determined by the RM at iteration $n-1$.

RM to Tasks: After receiving messages from *all* tasks, the RM updates the cost $p^{(n)}$ by:

$$p^{(n)} = p^{(n-1)}\left(1 + \left(\sum_{i=1}^{I} x_i^{(n)} - R\right) / \gamma^{(n)}\right) + \chi_+\left(\mathbf{x}^{(n)}, \gamma^{(n)}\right), \tag{94}$$

where $\gamma^{(n)}$ is the step size parameter that affects the change in the cost at each iteration, and $\chi_+\left(\mathbf{x}^{(n)}, \gamma^{(n)}\right) = \max\left(0, \left(\sum_{i=1}^{I} x_i^{(n)} - R\right) / \gamma^{(n)}\right)$ is a subgradient metric to force the cost to be non-zero if the cost is initialized at zero, and resource demands are infeasible [104]. The RM then submits the message $m_{-i}^{(n)} = p^{(n)}$ back to each task $i \in \{1,...,I\}$.

One way to understand this algorithm is that if the cost $p^{(n)}$ is too low, each task will demand too much computational resource and overutilize the system. If the cost is too high, the tasks will not demand enough resources to fully utilize the system. Hence, using (94), the cost is increased if the excess resource demand $\sum_{i=1}^{I} x_i^{\langle n \rangle} - R$ is positive (i.e. system is overutilized), and decreased when the excess resource demand is negative (i.e. system is underutilized).

Stage 3: Allocation Stage.

After the messages have been exchanged repeatedly, and the resource demands converge to a feasible vector $\mathbf{x}^*$, each task i is then allocated resource quantity x_i^*. Similar problems in distributed algorithm design have shown that convergence can be guaranteed if the value of $\gamma^{\langle n \rangle}$ is picked at every stage of the process from an increasing sequence, as long as the sum of the sequence generated by $1/\gamma^{\langle n \rangle}$ is infinite (e.g. $\gamma^{\langle n \rangle} = n$) [103] [104].

4.3 MEP-based Algorithms for Energy Minimization

4.3.1 An Energy-Minimizing Solution for an Always-Active System

To demonstrate how the MEP can be applied for energy minimization, in this section we propose a MEP-based algorithm to simultaneously maximize the social welfare of applications while minimizing energy consumption for dynamic voltage scalable (DVS) systems. A DVS-enabled processor can change its operating frequency by adjusting its voltage level, which affects its energy consumption rate. The active energy consumed by a DVS-enabled processor can be modeled as a *convex* increasing function of the processor workload [40][43][64]. Since modeling system behavior is not the focus of this chapter, we will forego energy consumption and dissipation details that can vary between different single- and multi-core architectures, and model the total energy consumption as a convex function of the *total workload* across the system.[4] The convex, multi-objective optimization problem can be given by:

[4] We note that this is an incomplete model, since the energy function may not be purely convex, or purely a function of the total workload across multiple cores. Future work can explicitly address overheads associated with processor-sharing and memory-access.

Social Welfare-Maximizing Energy-Minimizing (SWMEM) Optimization Problem:

$$\max_{x_i} \sum_{i=1}^{I} Q_i(x_i) - \lambda E_{\text{tot}}^{\text{act}}\left(\sum_{i=1}^{I} x_i\right), \quad \text{s.t. } x_i \geq 0 \tag{95}$$

where x_i denotes the computational resources allocated to task i, $E_{\text{tot}}^{\text{act}}(R)$ is the minimum *active* energy of a multiprocessor system given the total computational resources allocated R, and $\lambda > 0$ is a weighting factor that determines the relative importance of system energy to social welfare utility. Note that the *passive* energy is assumed to be constant and therefore not considered in the equation. However, the passive energy becomes important for multiprocessor systems where each of the multiple processing elements (PEs) can enter sleep mode, as discussed in the next subsection.

The MEP tax function that achieves the optimal solution to (95) is:

Excess Energy Minimizing (EEM) Tax Function:

$$t_i(x_i) = \lambda E_{\text{tot}}^{\text{act}}(x_i + d_i) - \lambda E_{\text{tot}}^{\text{act}}(d_i), \tag{96}$$

where x_i is task i's total computational resource demand, and d_i is task i's "perceived" total computational resource demand from all other tasks (which will be discussed later). The EEM tax function has the following interpretation: the system charges each task the amount of energy the system consumes when the task runs on the system, minus the amount of energy the system consumes if the task does not run on the system. Note that the second term in the tax function, $\lambda E_{\text{tot}}^{\text{act}}(d_i)$, does not depend on x_i, but is there to ensure that each task i is not taxed if it does not comsume system resources, i.e. $t_i(0, m_{-i}) = 0$.

As in the SWM algorithm, the RM can introduce a step size $\gamma^{(n)}$ to ensure convergence of the EEM algorithm. In this case, assuming that $x_i^{(n-1)}$ is the computational demand from each task i during the previous iteration, and $x_i^{(n)}$ the demand during the current iteration, the RM submits the following message to task i to update the tax function:

$$d_i = m_{-i}^{(n)} = \sum\nolimits_{l \neq i}\left(x_l^{(n-1)}\right) + \frac{1}{\gamma^{(n)}} \sum\nolimits_{l \neq i}\left(x_l^{(n)} - x_l^{(n-1)}\right), \tag{97}$$

such that the "change in demand" seen by each task is scaled down by $\gamma^{(n)}$. The step size parameter $\gamma^{(n)}$ can be chosen from an increasing sequence to guarantee convergence. We omit a rigorous proof of convergence here, as a similar proof can be found in [93], but we will prove that for strictly concave task quality functions (in terms of x_i) and convex energy functions, the EEM tax function generates an optimal solution for the SWMEM objective upon convergence. This is an important result for multimedia applications, where the quality-complexity functions are concave [99]. The convergence time will be further analyzed in the results section (Section 4.6).

Proposition 8: For strictly concave quality functions and convex energy functions, the EEM tax function generates an optimal solution to the SWMEM upon convergence.

Proof: See Appendix A. We note that the proof provides us with an additional guideline for designing tax functions that achieve globally optimal solutions, i.e. tax functions must be designed such that at equilibrium, the Karush-Kuhn-Tucker (KKT) conditions are uniquely and simultaneously met for each task's local optimization problem as well as the system's global optimization problem.■

4.3.2 Energy-constrained SWM for Multi-processor Systems with Sleep Modes

In this section, we consider a social welfare-maximizing solution for a multiprocessor system with a fixed total energy constraint. The PEs are DVS-enabled and can switch between active and sleep modes. We assume that a PE can go to sleep during a control interval of length T and consume negligible energy if it is not assigned any jobs to process. However, a PE cannot be both active and sleeping within the same control interval. For an active, DVS-enabled PE j, it is shown that, given a total computation requirement (e.g. cycles) r_j during an interval of length T, the PE should run at a constant frequency $f_j = r_j / T$ throughout the interval in order to minimize energy consumption [64]. Thus, the sleep mode-enabled energy function for each PE j can be given by:

$$E_j(r_j) = \begin{cases} 0 & , r_j = 0 \\ TP_j^{\text{pass}} + \alpha_j T P_j^{\text{act}}\left(\frac{r_j}{T}\right) & , r_j > 0 \end{cases}. \tag{98}$$

where the passive power for PE j, P_j^{pass}, is constant, and the active power $P_j^{\text{act}}(f_j)$ is a convex function of the operating frequency f_j [40][43] .[5]

Note that the energy function for each PE is no longer convex due to the discontinuity at $r_j = 0$ in (98). However, because the function is piecewise convex, we can still identify all the local minima in the multiprocessor energy function, and search through them in a combinatorial fashion to find the globally optimal point. Prior to running the SWM, we determine, for a given energy constraint, the subset of PEs to turn on in order to maximize the available computational resources. The optimal configuration is the one with the loosest computational resource constraint. The algorithm is summarized in Table 9.

4.4 MEP-based Algorithms for Miscellaneous System Objectives

4.4.1 Assigning tasks to PEs using tax functions

A practical concern for multiprocessor systems is how to assign tasks to different PEs in order to minimize energy consumption. Many multi-core systems provide software programmers with the flexibility to choose system configurations that can better exploit thread-level parallelisms in their applications [84] [85]. While this can improve the performance of individual parallelizable applications when resource availability is high, arbitrary resource usage can hurt the overall performance of an energy-constrained system, since many applications might compete over the same set of over-utilized PEs.

One possible solution is to remove this programming flexibility and allow the RM to centrally schedule applications across the PEs. However, we propose an alternative idea which keeps the flexible programming feature, but introduces tax functions to *compel* applications to run on PEs that are either underutilized, or have lower energy

[5] Note that when there are finitely many voltage levels for a PE, the power as a function of CPU requirements is piecewise continuous and increasing. Since energy consumption is the integral of power with respect to time, it follows that energy is a piece-wise linear (and convex) function due to the power being a non-decreasing function.

consumption rates. We note that determining this optimal assignment of PEs is beyond the scope of this thesis. The main purpose of this section is to provide an alternative, practical method for the RM to assign applications to PEs, given that an efficient assignment has already been computed.

To provide an illustration of our algorithm, we construct an example tax function that compels individual tasks to run on individually assigned PEs. If a task is assigned to a particular PE, it is taxed the amount of energy consumed as if no other tasks were running on the PE. However, if a task is not assigned to a particular PE, it must pay a penalty equal to the total energy increase it causes to all other tasks sharing that PE. Without loss of generality, we assign task i to a PE $j = i$, where we have assumed that $I \leq J$, such that every task has a uniquely assigned PE. If task i makes resource demands across J PEs given by $\mathbf{r}_i = (r_{i,1}, r_{i,2}, \ldots, r_{i,J})$, its quality is modeled by a concave increasing function $Q_i\left(\sum_{j=1}^{I} r_{i,j}\right)$. Using an appropriate step size $\gamma^{\langle n \rangle}$, the RM sends the following messages to each task i during each iteration n:

$$m_{-i}^{\langle n \rangle} = \left(d_{i,1}, d_{i,2}, \ldots, d_{i,i-1}, d_{i,i+1}, \ldots, d_{i,J}\right),$$
$$\text{where } d_{i.j} = \sum\nolimits_{l \neq i}\left(r_{l.j}^{\langle n-1 \rangle}\right) + \frac{1}{\gamma^{\langle n \rangle}} \sum\nolimits_{l \neq i}\left(r_{l,j}^{\langle n \rangle} - r_{l,j}^{\langle n-1 \rangle}\right). \tag{99}$$

where $r_{i,j}^{\langle n \rangle}$ is task i's resource demand on processor j during iteration n. The application then maximizes its utility based on the following tax function:

The PE Assigning Tax Function (PA):

$$t_i^{\mathrm{PA}}\left(\mathbf{r}_i, m_{-i}^{\langle n \rangle}\right) = \alpha_i E_i^{\mathrm{act}}\left(r_{i,i}\right) + \sum_{j \neq i} E_j^{\mathrm{act}}\left(r_{i,j} + d_{i,j}\right) - \sum_{j \neq i} E_j^{\mathrm{act}}\left(d_{i,j}\right). \tag{100}$$

Here, task i's perceived total resource demand on PE j by all other tasks is given by $d_{i,j}$. Note that similar to the EEM tax function, the perceived change in demand is scaled by the step size $1/\gamma^{\langle n \rangle}$.

The PA algorithm satisfies the condition where a task does not have to pay taxes for any processor that it does not use. It can also be shown that the decentralized PA algorithm often converges to a solution where a task will never run on another task's assigned processor, unless the derived benefit is very large. (The proof is similar to

Appendix A and is therefore omitted.) Note that while we have constructed a tax function to assign a single processor to each task in this subsection, a similar tax function exists for assigning multiple processors to each possibly multi-threaded task. This can be identically achieved by penalizing a task less for a *set* of dedicated processors.

4.4.2 Power Scheduling for Interdependent Multimedia Jobs

In this subsection, we propose a power scheduling solution for maximizing the social welfare of multiple applications with interdependent jobs. For example, in MPEG, if an I-frame is decoded at a lower resolution using less resources, the following P-frames and B-frames will also be decoded under extra distortion regardless of the amount of resources allocated to them, since they depend on the distorted I-frame to reproduce their respective video frames. Hence, proper power scheduling is an important issue for optimizing the performance of real-time multimedia systems where jobs do not only have stringent delay deadlines, but the contribution of jobs to the overall quality may be highly interdependent.

We formulate the scheduling problem as follows: Define a super-interval T_S as consisting of N_S time intervals of size T, where T is the time between successive video frames, and N_S is the size of an interdependent group of frames/pictures (GOP). For the $m\text{th}$ time interval in super-interval T_S, a corresponding quantity of computational resources $x_{i,m}$ is allocated to task i. The resulting quality function with N_S time intervals per super-interval then takes on the form:

$$Q_i(\mathbf{x}_i) = Q_i\left(x_{i,1}, x_{i,2}, \ldots, x_{i,N_S}\right), \tag{101}$$

where $\mathbf{x}_i = \left[x_{i,1}, x_{i,2}, \ldots, x_{i,N_S}\right]^{\mathrm{T}}$. For simplicity, we allow only the time interval $[(m-1)T, mT)$ in the super-interval for decoding the $m\text{th}$ job in the GOP.

The system objective is then to simultaneously maximize the sum of task qualities while saving energy by jointly allocating resource shares to each task, and adjusting the operating level of the DVS-enabled processor for each time slot. The optimization function is as follows:

Joint SWMEM and Scheduling Optimization Problem (SWMEM-S):

$$\max_{\mathbf{x}_i} \sum_{i=1}^{I} U_i(\mathbf{x}_i, \lambda) = \max_{\mathbf{x}_i} \sum_{i=1}^{I} Q_i(\mathbf{x}_i) - \lambda \sum_{m=1}^{N_S} E_{\text{tot}}^{\text{act}}\left(\sum_{i=1}^{I} x_{i,m}\right), \quad \text{s.t. } \mathbf{x}_i \geq \mathbf{0} \tag{102}$$

Since SWMEM-S is simply the multidimensional version of the SWMEM, by a proof similar to Proposition 8, we can show that the MEP tax function which achieves this global objective function is the corresponding multidimensional version of the EEM:

Joint EEM and Scheduling Tax Function (EEM-S):

$$t_i^{\text{EEM-S}}(x_{i,m}, d_{i,m}) = \lambda \sum_{m=1}^{N_S} \left[E^*(x_{i,m} + d_{i,m}) - E^*(d_{i,m})\right], \tag{103}$$

where $d_{i,m} = \sum_{l \neq i}\left(x_{i,m}^{(n-1)}\right) + (1/\gamma^{(n)})\sum_{l \neq i}\left(x_{i,m}^{(n)} - x_{i,m}^{(n-1)}\right)$ is computed by the RM from the resource demands $x_{i,m}^{(n)}$ for each interval m during algorithm iteration n.

4.4.3 Summary of MEP Algorithm Design Requirements

The examples in Sections 4.4 and 4.5 were provided to illustrate a formal method for decentralized algorithm design via tax functions. To clarify our approach, three main requirements are summarized below.

First, an analytical optimization problem must be defined. What is the social welfare function that we are trying to maximize? What are the system resources (energy, workload on specific PEs, etc.) that we are trying to conserve? What are the resource constraints?

Secondly, to guarantee convergence in static environments, a proper step size sequence $\gamma^{(n)}$ must be chosen [103]. As we will discuss in the next section, different choices of $\gamma^{(n)}$ can also affect the adaptation time for our algorithms in dynamic environments.

Finally, to guarantee *optimality* upon convergence, the KKT conditions must be simultaneously and uniquely met for the system objective function and each task's local optimization function. Hence, the analytical tax functions must be designed according to the goal that the system is trying to achieve.

4.5 Efficient Resource Allocation for Temporally-correlated Quality Functions

In this section, we propose adaptive resource allocation solutions that explicitly take advantage of the temporal correlations exhibited by the time-varying quality-resource functions of multimedia applications. For illustration purposes, we focus on the MEP-based SWM algorithm for video decoding applications. First, we propose a Markov model for video decoding quality functions based on the collected statistics from various video sequences (Subsection 4.5.1). We then discuss the information required by the RM to minimize the adaptation time, and we present how to statistically choose optimal step sizes for the SWM algorithm based on information known about the dynamics of the video quality functions (Subsection 4.5.2). Finally a simple algorithm with very low informational overhead is proposed (Subsection 4.5.3), which fixes the cost function during intervals where video quality functions do not vary significantly.

4.5.1 Modeling the Quality Functions for Dynamic Video Applications

Video sequences exhibit highly time-correlated characteristics, and are often encoded using similar GOP structures over time. Hence, we can model video decoding quality-resource functions as discrete time Markov random processes $Y_i(x_i, n)$, where each index n corresponds to the GOP number in the sequence, and the complexity x_i in each period n corresponds to the complexity of the entire GOP. The characteristics of the Markov process, such as its distribution, can be obtained using training data. For example, the distribution of video decoding quality for several video sequences are given in Figure 26 using the coder in [99]. We found that the Gaussian distribution approximates well the PSNR statistics (in dB) of each sequence for fixed complexity levels (Figure 26(c-d)). Hence, we model the video decoding quality function for each task i as a Gaussian Markov chain. The Gaussian Markov chain can be perfectly described by the mean quality function $Q_i(x_i)$, the perturbation variance $\sigma_i^2(x_i)$, and the autocorrelation coefficient ρ_i between adjacent GOPs in the sequence (See Figure 26(a-b).), i.e.:

$$\begin{aligned} Y_i(x_i,n) &= Q_i(x_i) + \sqrt{\rho_i}\left(Y_i(x_i,n-1) - Q_i(x_i)\right) + \sqrt{1-\rho_i}W_i(x_i) \\ &= (1-\sqrt{\rho_i})Q_i(x_i) + \sqrt{\rho_i}Y_i(x_i,n-1) + \sqrt{1-\rho_i}W_i(x_i) \end{aligned}, \tag{104}$$

where $W_i \sim N\left(0,\sigma_i^2(x_i)\right)$ is a white Gaussian vector with variance $\sigma_i^2(x_i)$. Note that according to our measured statistics (Figure 26(a-b)), ρ_i depends little on x_i, and can therefore be modeled as a constant rather than a function of the complexity. Hence, based on any complexity level x_i, ρ_i can be estimated by:

$$\rho_i = \frac{E[(Y_i(x_i,n) - Q_i(x_i))(Y_i(x_i,n-1) - Q_i(x_i))]}{\sigma_i^2(x_i)}. \tag{105}$$

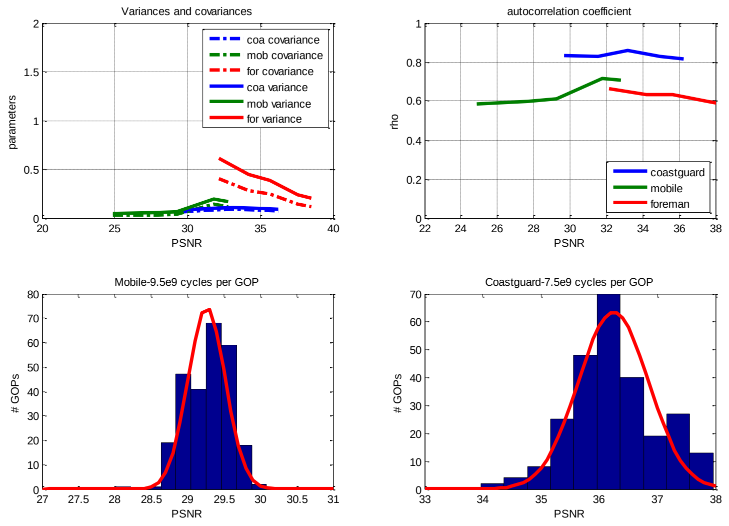


Figure 26 a) Variances and covariances and b) autocorrelation coefficient ρ for quality perturbations in the *Coastguard*, *Mobile*, and *Foreman* sequences. c-d) Gaussian fits to the PSNR distributions (in dB) for *Mobile* and *Coastguard* when operating at particular complexity levels. The complexity is the number of entropy decoding and inverse transform cycles per GOP.

The quality function $Y_i(x_i,n)$ will change its shape based on the first derivative of $\sigma_i(x_i)$. We approximate this perturbation variance using the following linear model:

$$\sigma_i(x_i) = a_i x_i + b_i. \tag{106}$$

Hence, the change in the first derivative of the quality function from the previous time interval is:

$$\varsigma_i(n) = \frac{dY_i(x_i,n)}{dx_i} - \frac{dY_i(x_i,n-1)}{dx_i}, \tag{107}$$

where $\varsigma_i(n)$ is a Gaussian random variable with standard deviation on the order of $\sqrt{\rho}a_i$, which is independent of x_i based on the model in (106). Note that for the SWM algorithm, the larger the variance of $\varsigma_i(n)$, the more suboptimal the allocation will be if the cost per unit resource p is not accurately updated.

4.5.2 Informational Requirement for Single Iteration Adaptation, and Optimal Step Sizes

In general, if the RM knows the second derivative $Q_i''(x_i)$ of each task's quality function, the optimal solution can be determined by the following two steps. First, the RM can project a cost per unit resource for the tasks, and each task's response $\hat{x}_i$ can be used to reconstruct the first derivative function given $Q_i'(\hat{x}_i) = p$. Once $Q_i'(x_i)$ is known, the optimal cost can be obtained, and a centralized solution can be used for resource allocation. Hence, the second derivative provides the RM with sufficient information to optimally update the resource allocation in a single iteration.

However, $Q_i''(x_i)$ is generally not known by the RM. Nevertheless, if some information about $Q_i''(x_i)$ can be gathered by the RM, better convergence time can be guaranteed for the SWM by dynamically adjusting the step size parameter γ (Note that the step size is no longer chosen from a predetermined increasing sequence $\gamma^{(n)}$, but is updated based on the behavior of the video quality functions.) To demonstrate our approach, we propose the following modification to the SWM algorithm:

SWM for Updating Resource Allocations (SWM-U):

Based on the model in (104) and (107), we have the following task level optimization function for time interval n:

$$\begin{aligned} &\max_{x_i,p_i} Y_i(x_i,n) - t_i \\ &= \max_{x_i,p_i} Y_i(x_i,n-1) + \varsigma_i(n)x_i - t_i(x_i,m_{-i}) \\ &= \max_{x_i,p_i} Y(x_i,n-1) + \varsigma_i(n)x_i - px_i. \end{aligned} \tag{108}$$

(Since the costs are already assumed to be non-zero in the previous iteration, we exclude the last term $\chi(\bullet)$ from (94).) As can be seen, the random quality perturbation function $W_i(x_i)$ causes the equilibrium point to shift, as the perturbations now affects each task's perceived cost per unit resource to be $p - \varsigma_i$. To construct a single iteration algorithm that accurately updates resource allocation, a good choice of γ is needed. The following toy example provides insight on how to determine the best choice for γ.

Consider the case when all tasks have identical quality functions $Q(x)$, with initially balanced resource allocations x. At the next time interval, each task identically update his function to $Q_{new}(x) = Q(x) + \varsigma x + \xi$, where $\varsigma, \xi \in \mathbb{R}$. Hence, the new derivative can be written as:

$$Q'_{new}(x) = Q'(x) + \varsigma \tag{109}$$

Based on this new function and the old cost $p = Q'(x)$, a task will update its resource demand to x', where $Q'_{new}(x') = Q'(x') + \varsigma = Q'(x)$. Each task now has an identical excess demand of $\Delta x = x' - x$. The second derivative of $Q(x)$ can be approximated between the points x and x' as:

$$Q''(x) \approx \frac{Q'(x + \Delta x) - Q'(x)}{\Delta x} = \frac{Q'(x') - Q'(x)}{x' - x} = \frac{-\varsigma}{\Delta x} \tag{110}$$

Based on the modified SWM-U tax form (108), setting $\gamma = -\varsigma / \Delta x$ is the optimal choice for that time interval. Note that γ depends on ς and the excess demand; hence if ς is known for every interval, γ can be chosen close to optimal. For the general case with many tasks and different quality-resource functions, γ can be set to the average value of all individually calculated γ_i's. Even when precise information is unavailable, if the system can still obtain an estimate of the quality perturbation autocorrelation coefficient ρ and variance σ over several time intervals, a distribution of ς can be obtained. By estimating the excess demand Δx over several intervals, we can obtain a statistically optimal $\gamma \approx \mathrm{E}[\varsigma / \Delta x]$.

4.5.3 A Fixed Cost Algorithm

Finally, we propose a low-complexity *fixed cost algorithm* (FCA), which updates resource allocations *without* any communications overhead aside from determining an initial cost per unit resource p. The assumption is that when ς is small, the cost of the SWM-U remains approximately fixed. Hence, the only overhead associated with this algorithm occurs when source characteristics change significantly, and a new cost needs to be determined using a few iterations of the SWM. Otherwise, each task can run the FCA in isolation.

In the FCA, based on the initial cost p obtained by the SWM at the beginning of a video scene, tasks simply determine their demand for each time interval n using the cost p without communicating with the RM, i.e.:

$$x_i^{\mathrm{demand}}(n) = \arg\max_{x_i} Y(x_i, n) - x_i p\,. \tag{111}$$

Notice from Figure 26(b) that the perturbation variance is fairly constant across different complexities for each sequence, which means the slope parameter a_i is very small. Even for sequences such as *Mobile*, the perturbation parameter a_i is about 10% the slope of its quality function. Moreover, the high correlation coefficient ρ further reduces the effect on the cost change between adjacent intervals, such that the perturbation-to-quality ratio is approximately $\sqrt{\rho}\sigma_i(x_i)/Q_i(x_i)$. Hence, as long as the initial cost converges, the FCA should have a cost approximately equal to that of the SWM-U, which leads to approximately identical resource allocations. We will verify this in the results section.

4.6 Simulations and Results

To measure the performance of our resource management algorithms, we used the video coder from [99] in our simulations to encode various sequences at different bit rates, and we gathered the number of cycles for each decoding job on a Pentium IV processor. We note that while we have chosen a specific coder, our methodology can just as easily apply to any other quality-resource scalable video coding algorithms (e.g. encoding with different macroblock sizes and prediction modes in H.264). In order to verify that our algorithms are suitable for DVS-enabled PEs, we used the StrongARM processor profile [105].

4.6.1 SWM and FCA Behavior for Dynamic Quality Functions

In Figure 27, the iterations of the SWM for computational resource allocations are plotted for 2 *Foreman*, 2 *Coastguard*, and 1 *Mobile* sequences sharing the same system.

Note that only one curve for the *Foreman* and *Coastguard* sequences are plotted, since resource allocations are identical for the same sequences. Negative indices refer to iterations for global adaptation (via the SWM-based algorithm), such that the allocations can converge before the video starts playing at iteration/time 0. The MEP iterations for the SWM-U and FCA are then synchronized with the video sequence during runtime, such that one iteration of the MEP is performed between each GOP. (The video quality function changes after each GOP.) Note that we purposely ran 8 iterations of the SWM algorithm during global adaptation before running the FCA to allow the cost to converge. On the other hand, 4 iterations were sufficient for SWM global adaptation before running SWM-U, since the SWM-U continued to adapt its cost metric and resource allocations accurately during runtime under *very low complexity* (i.e. only a single MEP iteration is required for accurately updating the resource allocation between each GOP!). Moreover, because the cost fluctuates little upon convergence (Figure 27(c, f)), the FCA, which requires no communications overhead except at the beginning of the video sequence, performs almost equally to the SWM-U.

4.6.2 EEM Convergence and Tradeoff Parameter λ

The convergence of the decentralized EEM algorithm to the optimal SWMEM solution is shown in Figure 28, based on the StrongARM energy model [105], where the voltage V is proportional to the frequency f, and hence the energy $E \propto V^2 \propto f^2$. The same five video tasks above were also used for this experiment. The step size parameter is fixed at $\gamma = 1$, such that the real resource demand is projected for each application (see (97)), and the energy weight is set to $\lambda = 2 \times 10^{-3} \mathrm{PSNR/Joule}$. The EEM converges quickly (after only 5 iterations) and is therefore ideal for resource allocation in dynamic environments.

In Figure 29, we plotted the video qualities and the energy consumed for different values of λ in the SWMEM/EEM, and we compared it against an application-agnostic fair scheduling scheme. Note that the EEM allocates more resources to the *Mobile* sequence since its performance increases drastically with increased resources. On the other hand, *Coastguard*, which benefits less from increased resources, receives less total resources and therefore has lower quality. Overall, the sum of PSNRs for all tasks, shown in Figure 29b, is 5-10 dB higher for the EEM than for fair share resource allocation at low-power regions, which translates to a significantly higher average video quality per task (1-2 dB PSNR). We also performed a comparison between the application-agnostic fair scheduling scheme using the Nash product social welfare objective (a utility fair scheme), given by the product of application utilities (Figure 30) [86]. This can be achieved by requiring each task to use a utility function that is the *log* of its quality function, i.e. $\ln(Q_i(x_i))$. Note that the Nash product-based EEM algorithm obtains a much fairer allocation of resources than simple resource-based fair scheduling, since the quality of the *Mobile* sequence, which requires far more computational resources than the *Foreman* and *Coastguard* sequences, is greatly increased.

4.6.3 Allocation Behavior for the Processor Assigning Algorithm

We simulated the performance of the PA algorithm by assigning the 5 video decoding tasks to 5 separate, identical DVS-enabled processors. Note that from Table 10 and Figure 31a, for 5 tasks and 5 processors, the final resource distribution allocates most of each task's complexity to its assigned processor after only 4 iterations of the PA algorithm, which demonstrates that using tax functions can compel tasks to run mostly on their assigned processors.

We also performed simulations to analyze the "suboptimality" of the PA algorithm compared to the ideal SWMEM solution when tasks can be arbitrarily parallelized across all PEs (See Figure 31b). Note that for very small λ, or high energy availability regions, the processor dedicating algorithm achieves approximately the same performance as the SWMEM. This may be explained by the fact that when energy is cheap, the video qualities achieved are near maximum and increase very little per unit energy allocated. Consequently, resource allocation is not an important issue when resources are highly available. However, the PA algorithm yields considerably less quality than the ideal SWMEM solution when the cost per unit energy λ is high, since the slope of video quality functions are much steeper and differ much more between sequences at low energy/quality levels. Indeed, the poor performance of the PA algorithm is largely a result of poor PE-to-task assignment, which indicates that smartly assigning/sharing processors is very important when energy is scarce.

4.6.4 Joint SWMEM and Scheduling Simulation

The SWMEM-S simulations were performed for a 3 level hierarchy of frames, and the quality-resource curves were used for the different bitplane-truncation points for each frame. Note that if a base-layer frame is decoded down to a certain bitplane, decoding the second and third layers to a finer quantization level does not contribute much to the overall quality, since the distortion created by the base-layer frame still exists. Hence, the quality of the decoded frames are interdependent. As shown in Table 11, by decoding the different priority frames during different time intervals and adapting the processor frequency in every interval based on application job dependencies, we can achieve better performance than rate-monotonic DVS, which is the optimal application-agnostic policy for saving energy given hard deadlines. Using the same 5 video decoding tasks as above, we showed that the SWMEM-S achieved approximately the same video quality as rate-monotonic DVS while providing over 28% in energy savings.

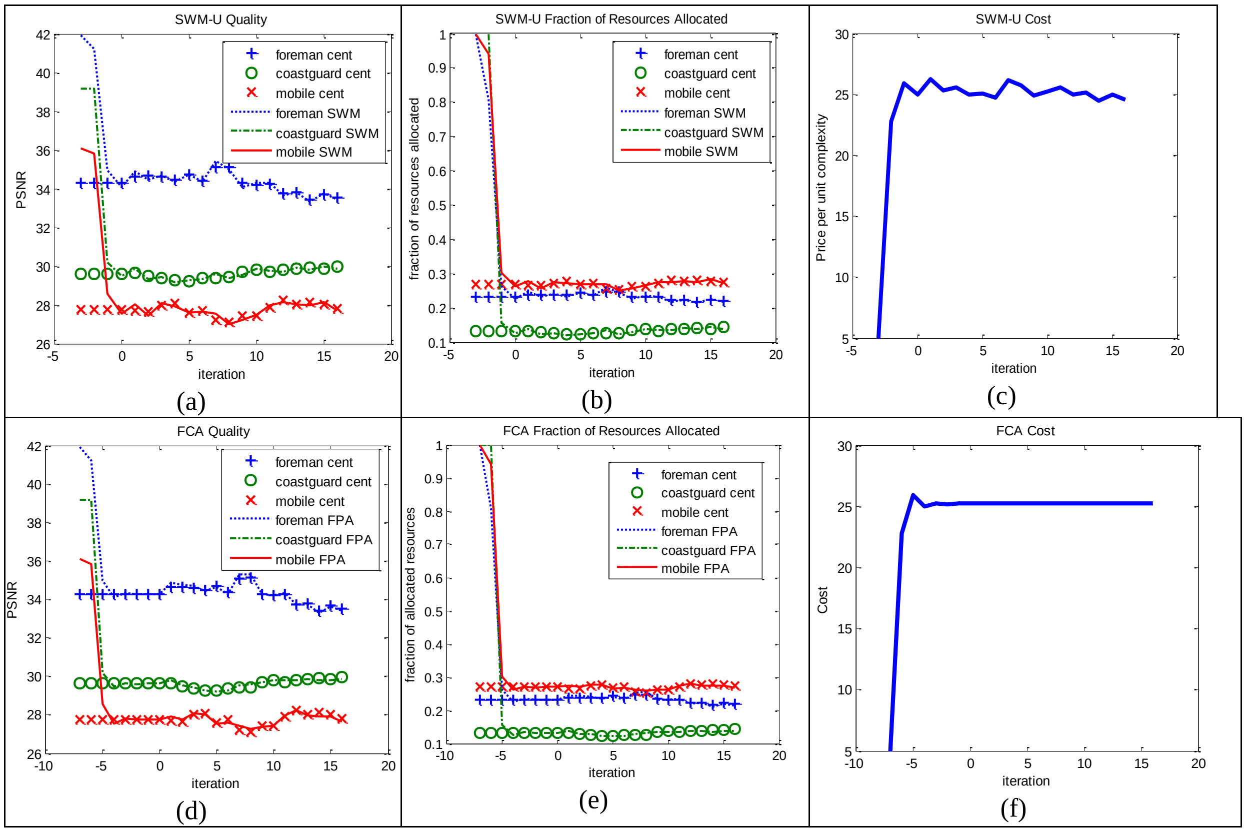


Figure 27 The (a) convergence of decentralized allocation qualities to centralized solutions, (b) the corresponding fraction of resources allocated to each task, and (c) the costs are shown for SWM-U. (d-f) show corresponding plots for the fixed cost algorithm (FCA).

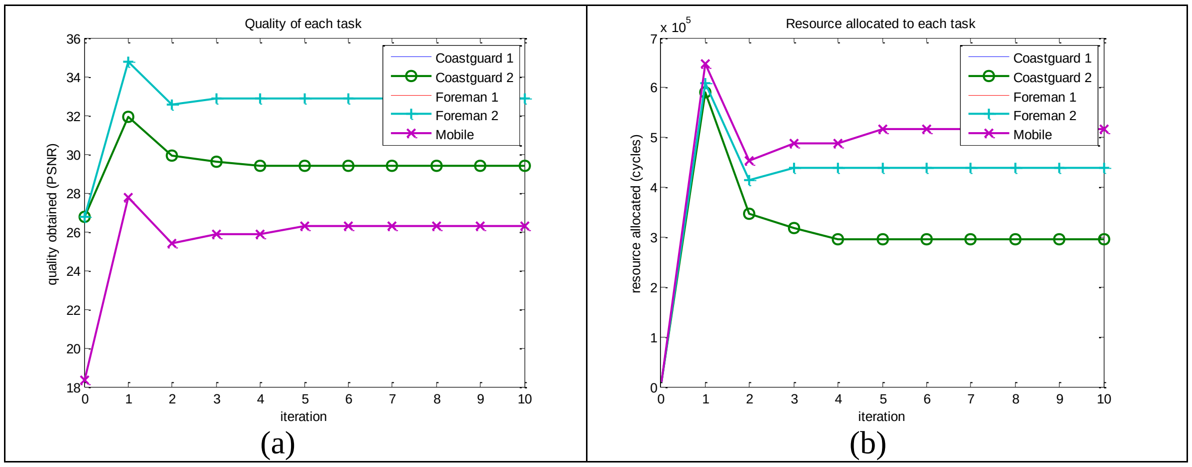


Figure 28 (a-c) The convergence of quality and computational resource allocation of EEM with energy function $E \propto x^2$.

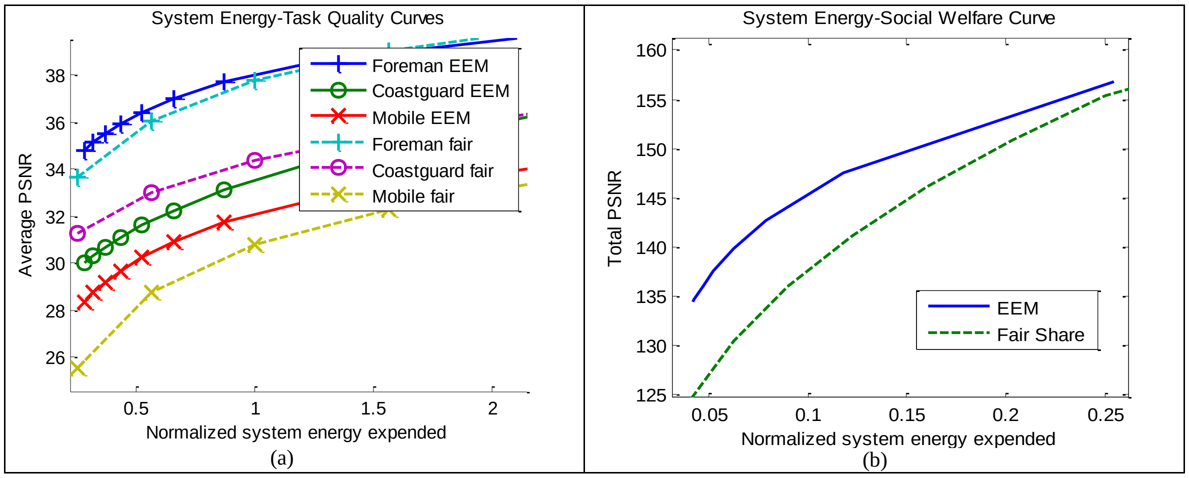


Figure 29 (a) The energy-quality curves for video sequences in the EEM algorithm compared against the fair share scheduling algorithm. (b) The plot of the sum of qualities (social welfare).

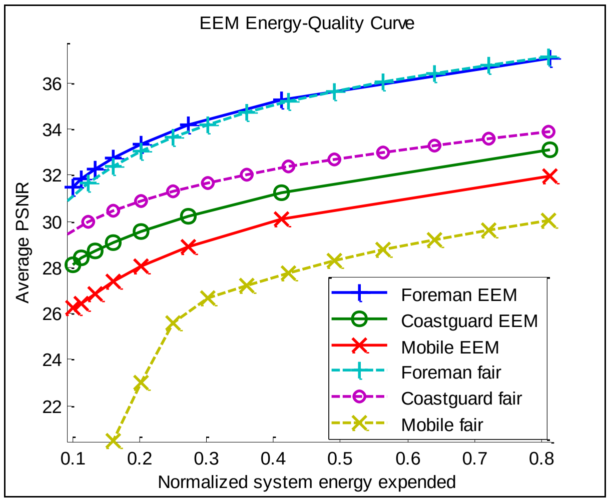


Figure 30 The energy-quality curves for video sequences using Nash product fairness, compared against the fair share scheduling algorithm.

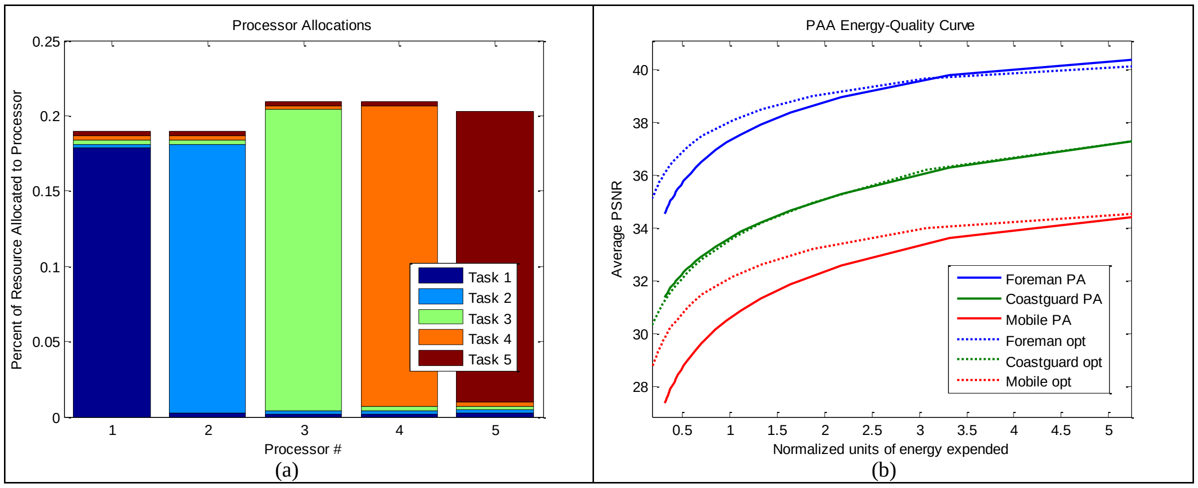


Figure 31 (a) Resulting processor resource distribution after 4 iterations of the PA algorithm. (b) Energy-quality curves for the PA algorithm and the optimal SWMEM solution.

Table 9 Sleep Mode Initialization for SWM

```
1.  // Initialize before runtime.
2.  Fix total energy constraint $E_{\text{tot}}$.
3.  Set $S = \{1, 2, ..., J\}$ to be the set of PEs.
4.  Construct the power set (set of all possible subsets) of PEs, $2^S$.
5.  For each subset of PEs $N \in 2^S$
6.    Calculate the active energy constraint: $E_{\text{tot}}^{\text{act}} = E_{\text{tot}} - \sum_{j \in N} E_{\text{pass}}^{N}$.
7.    Determine the computational resource constraint $\Upsilon_N = R_N\left(E_{\text{tot}}^{\text{act}}, N\right)$ by using the inverse of
      the energy model $E_{\text{tot},N}^{\text{act}}(R)$ for only the PEs in $N$.
8.  End For
9.  Turn on the PEs corresponding to the $N$ with the largest $\Upsilon_N$.
10. // During runtime.
11. Run SWM with complexity constraint $\Upsilon_N$.
```

Table 10 Average number of cycles per frame for task to PE Allocations (4 iterations of the PA algorithm).

Task/Processor	PE 1	PE 2	PE 3	PE 4	PE 5
Foreman 1	267982	3564	4333	4333	4019
Foreman 2	3564	267982	4334	4333	4019
Coastguard 1	3237	3237	299839	3885	3571
Coastguard 2	3237	3237	3884	299842	3571
Mobile 1	3310	3310	4077	4078	289221

Table 11: Comparison of SWMEM-S and rate monotonic (normalized) complexity/energy allocations for various time intervals.

	Rate Monotonic			SWMEM-S/EEM-S		
Time Interval / Priority	1	2	3	1	2	3
Task 1	75.0	60.0	120.4	108.2	74.0	43.9
Task 2	75.0	60.0	120.4	108.2	74.0	43.9
Task 3	200.0	190.2	154.8	201.6	161.1	128.7
Task 4	200.0	190.2	154.8	201.6	161.1	128.7
Task 5	200.0	249.7	199.7	199.3	160.5	129.0
Total Complexity	750.0	750.0	750.0	818.9	630.7	474.1
Total Energy	12.66E			9.07E		
Average PSNR	29.75 dB			29.80 dB		

4.7 Conclusion

In this chapter, we presented a low-complexity, informationally-decentralized resource management solution for multiple multimedia applications sharing a resource-constrained system. By formalizing a message exchange protocol between applications and a system resource manager, we verified that our solution can be used to optimally achieve various objectives, such as maximizing the social welfare of applications, minimizing system energy consumption, assigning processors to applications, and efficiently updating the resource allocation in dynamic environments. In summary, our MEP solution can be used to construct robust resource allocation solutions, even when applications are unwilling to reveal private information about their utilities.

An avenue for future work is to construct and evaluate informationally-decentralized algorithms based on more sophisticated (and possibly non-convex) resource models that deal explicitly with various elements and features that can exist within a heterogeneous system (e.g. shared memory and caches between PEs, special-function processors, voltage islands, etc.).

CHAPTER 5
Resource-constrained Configuration of Classifier Cascades in Distributed Stream Mining Systems

5.1 Introduction

In this chapter, we introduce the problem of configuring cascaded classifier topologies in resource-constrained, distributed stream mining systems. Many stream classification and mining applications implement topologies (ensembles such as trees or cascades) of low-complexity binary classifiers to jointly accomplish the task of complex classification [145]. It has been shown that boosting trees of weak classifiers enables the successive identification and filtering of multiple attributes in the data, and leads to improved accuracy over single classifier systems [146][109].

Distributed stream mining systems provide a scalable infrastructure capable of supporting such classification applications, since different classifiers can be naturally placed across different processing nodes, depending on the classifier workload and processing node resource availabilities. Nevertheless, when voluminous data streams need to be processed, resource constraints pose a major challenge for optimizing the performance of cascades of classifiers. A commonly used approach is load-shedding, where algorithms determine when, where, what, and how much data to discard given the observed data characteristics, desired Quality of Service (QoS) requirements [112]-[116], and delay constraints [117]. While naïve load shedding performs well for simple data management jobs such as aggregation, for which the quality of job results depend only on the sample size, this is generally not the case for jobs involving *classification* of data. Hence, recent work on intelligent load shedding [118] attempts instead to maximize certain Quality of Decision (QoD) measures based on the predicted distribution of feature values in future time units. Nevertheless, this approach considers only information pertaining to a single classifier, and hence can be highly suboptimal for an application that requires the use of an entire cascade of (distributed) classifiers. Moreover, without a joint consideration of resource constraints at all downstream classifiers in the chain, the end-to-end processing delay for a chain of classifiers can become intolerable for real-time applications [134][135].

To address these issues, we consider an optimization concept for load shedding proposed in recent work, which involves reconfiguring the *operating point* of each classifier in a cascaded topology to ensure that constraints are met at each classifier [120]. For example, SVM-based classifiers can adjust different thresholds for detection, which in turn influence both the detection and false alarm probabilities, as well as the load forwarded to the next classifier [148]. Configuring the operating points of each classifier enables the chain to maintain high accuracy of information retrieval under resource constraints, since the data returned by the chain consists of higher confidence data, while lower confidence data is more likely to be shed. In this chapter, we will first review this methodology for configuring classifiers chains [119][120]. We then propose a novel solution extending this problem to configuring classifier trees, which go significantly beyond linearly cascaded classifiers by providing greater flexibility in data processing, while also posing different challenges in terms of resource constrained configuration. Specifically, while excess load can be easily handled within the optimization framework for a binary classifier chain by adapting the operating point of each classifier, using a single operating point for each classifier in a tree generates two output streams with a total sum rate that is fixed. Hence, it may not be possible to simultaneously meet tight processing resource constraints for downstream classifiers along both output edges when using only one operating point (e.g. threshold).

Instead, we propose configuring each classifier in a binary tree topology using *multiple* operating points (i.e. one for each output), e.g. with multiple overlapping and non-overlapping decision thresholds for the different classes. This directly enables intelligent discard of low-confidence data across output edges of each classifier when resources are scarce. Additionally, this also enables the intelligent *replication* of low-confidence data across both positive and negative edges when excess resources are available, which can significantly reduce the number of classification misses.

Additionally, we note that in this chapter, the problem we are analyzing is for stream mining systems that are owned by the same company or reside within the same administrative domain (e.g. IBM's System S stream processing core [108]). Hence, it can be assumed that classifiers in the topology are designed in a way to obey simple (often semantic) subset relationships, as shown in Figure 32 for a sports image concept classification tree [148]**Error! Reference source not found.**. In this tree, data is successively filtered from broad concepts to narrow concepts, such that large volume streams can be successively filtered along each branch of the tree. We will show that this problem already contains many interesting application specific features and details that require careful analysis and evaluation. However, Chapter 6 will introduce a more challenging scenario, where an application requires the use of classifiers that are distributed across different autonomous sites, and classifiers do not necessarily obey simple subset relationships. Thus, while the problem introduced in this chapter can be solved using conventional optimization techniques, the problem introduced in the next chapter will require a careful application of our proposed framework (i.e. a combination of modeling, information gathering, and multi-agent learning solutions) in order to provide efficient optimization solutions.

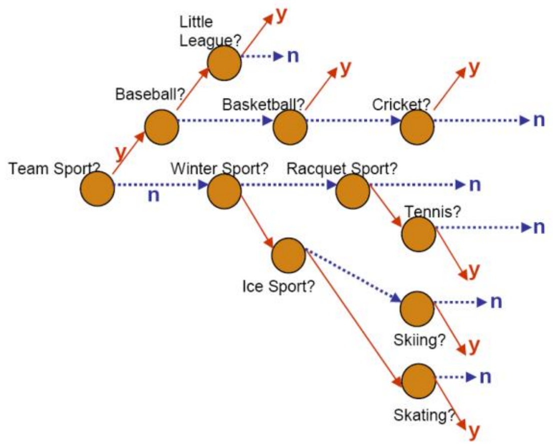


Figure 32 A hierarchical classifier system that identifies several different sports categories and subcategories.

This chapter is organized as follows. In Section 5.2, we discuss the model for a binary chain of classifiers, and the cost associated with its configuration. We extend the model to a binary classifier tree in Section 5.3, and formulate the misclassification cost minimization problem under resource constraints. We also include a discussion on the assertion that multiple operating point configurations outperforming single operating point configurations. In Section 5.4, we present experimental results using our algorithms for a sports image classification application, and discuss insights derived from the results. We conclude the chapter in Section 5.5 with some insights, as well as extensions to be addressed in the following chapter.

5.2 Model for Binary Chain of Classifiers

First, we review the configuration of operating points in a binary classifier chain, as shown in Figure 33. Each binary classifier v_i processes an input stream by classifying each stream data object (SDO) as belonging to a positive class H_i, or a negative class $\bar{\mathrm{H}}_i$. Each SDO that is classified as belonging to H_i is forwarded from a classifier node v_i to the next classifier node v_{i+1} in the chain. Otherwise the SDO is dropped from the stream. SDOs generated by the source v_0 are only received at the terminal v_N if they are forwarded by every classifier in the chain (i.e. they are relevant for the application).

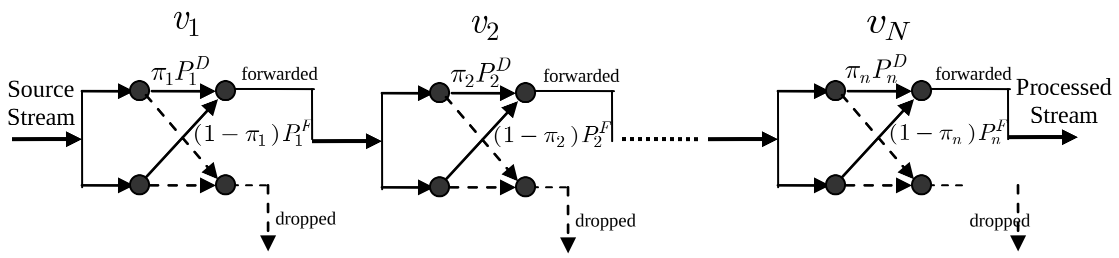


Figure 33 Classifier chain with probabilities labeled on each edge.

Given the ground truth X_i for an input SDO to classifier v_i, denote the classification decision on the SDO by $\hat{X}_i$. The proportion of correctly forwarded samples is captured by the *probability of detection* $(p_D)_i = \Pr\{\hat{X}_i \in \mathrm{H}_i \mid X_i \in \mathrm{H}_i\}$, and the proportion of incorrectly forwarded samples is captured by the *probability of false alarm* $(p_F)_i = \Pr\{\hat{X}_i \in \mathrm{H}_i \mid X_i \notin \mathrm{H}_i\}$.

Suppose that the input stream to classifier v_i has *a priori probability* (APP) π_i of being positive. The probability of forwarding an SDO to the next classifier (i.e. throughput) can be given by:

$$t_i = \pi_i \cdot (p_D)_i + (1 - \pi_i) \cdot (p_F)_i . \tag{112}$$

Moreover, the probability of *correctly forwarding* data (i.e. the goodput) to the next classifier is:

$$g_i = \pi_i \cdot (p_D)_i . \tag{113}$$

A filtering classifier is often designed such that the probability of detection is maximized subject to a false alarm probability constraint [125] [126]. Hence, by varying the false alarm constraint, different probabilities of detection can be obtained, and different volumes of the stream can be forwarded. Assuming that each classifier operates at a fixed complexity level, a detection-error-tradeoff (DET) curve, or a curve relating p_D^i to p_F^i, can be obtained for each classifier v_i. Hence, given the APP π_i and the DET curve, t_i and g_i become deterministic functions of p_F^i.

We can characterize the performance of a single classifier system by a weighted sum of its misses and false alarms, i.e.:

$$C = c_M (\pi_i - g_i) + c_F (t_i - g_i) . \tag{114}$$

This cost is practical in the sense that it identifies, for each data unit processed, the cost of making a mistake, which can be applied directly to various scenarios such as real-time manufacturing process control [150].

It is not as straightforward to determine a practical cost for a binary classifier chain, since the relationships between classifiers in the chain may be unknown. However, when successive classifiers filter out subset data, the principle of exclusivity can be applied [109]. Exclusivity implies that data that does not belong to the positive class of classifier v_i will not belong to the positive class if v_j for all $j > i$. Under this assumption, the end-to-end throughput and goodput for a classifier chain can be computed by the following recursive relationships:

$$\begin{bmatrix} t_{i+1} \\ g_{i+1} \end{bmatrix} = \mathbf{T}_{i+1} \begin{bmatrix} t_i \\ g_i \end{bmatrix}, \tag{115}$$

where
$$\mathbf{T}_i = \begin{bmatrix} (p_F)_i & \phi_i \cdot ((p_D)_i - (p_F)_i) \\ 0 & \phi_i \cdot (p_D)_i \end{bmatrix}, \tag{116}$$

and ϕ_i is the conditional a priori probability that positive data for classifier v_i also belongs to the positive class of v_{i+1}. For simplicity, we can normalize the input rate to 1, such that $t_0 = g_0 = 1$. Based on this, the end-to-end throughput and goodput can be computed, and the cost given as:

$$C = c_M(\pi_N - g_N) + c_F(t_N - g_N). \tag{117}$$

5.3 Minimizing Misclassification Cost in Binary Classifier Trees

5.3.1 Binary Classifier Tree Model and Misclassification Cost

Consider now a set of N binary classifiers labeled $1,\ldots,N$ cascaded in a tree topology, an example of which, for $N = 2$ is shown in Figure 34. This topology of classifiers may be used to identify data from 3 (in general K) end-to-end classes of interest. Each binary classifier n partitions input data objects into two classes, a ``yes" class $H_n^0$ and a ``no" class H_n^1, and forwards the classified data along respective output edges. For each respective class, denote the probability of correct detection by $(p_D^0)_n$ and $(p_D^1)_n$, and the probability of false alarms by $(p_F^0)_n$ and $(p_F^1)_n$. Note that when the classifier uses one operating point (e.g. thresholding) to label each data item as H_n^0 or H_n^1, we have the following relationships: $(p_D^1)_n = 1 - (p_F^0)_n$ and $(p_F^1)_n = 1 - (p_D^0)_n$. This coupling is, however, removed when we have multiple operating points (e.g. a different threshold for each output class).

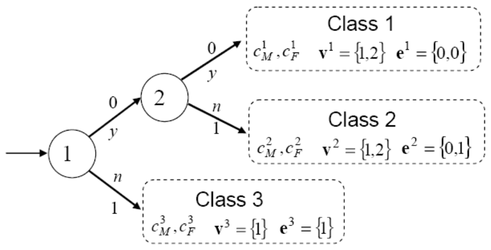

Class 1
c_M^1, c_F^1 $\mathbf{v}^1 = \{1,2\}$ $\mathbf{e}^1 = \{0,0\}$
Class 2
c_M^2, c_F^2 $\mathbf{v}^2 = \{1,2\}$ $\mathbf{e}^2 = \{0,1\}$
Class 3
c_M^3, c_F^3 $\mathbf{v}^3 = \{1\}$ $\mathbf{e}^3 = \{1\}$
1
2
y
n
0
1

Figure 34 Example of a depth-2 tree of classifiers with 3 terminal classes.

Each end-to-end class k is determined after a set of cascaded local classifications. Class \$k\$ is characterized by the following: N_k - the number of classifiers that data from that class need to pass through, $\mathbf{v}^k$- the sequence of classifiers in the path (e.g. each component from the set of classifiers $1,...,N$), $\mathbf{e}^k$ - the sequence of branch ``types" in the path (class 0 or 1), and c_M^k, c_F^k - the misclassification costs per miss and false alarm in class k. Assuming a (normalized) unit input data rate, the total misclassification cost for each class k can be computed from the total rate of data labeled as k (*throughput*) $\overline{t}^k$, the total rate of correctly labeled data in class k (*goodput*) $\overline{g}^k$, and the a-priori probability of data belonging to class k, $\overline{\pi}^k$, by the relation:
$C^k = c_M^k\left(\overline{\pi}^k - \overline{g}^k\right) + c_F^k\left(\overline{t}^k - \overline{g}^k\right)$ where the first term denotes the cost of misses, and the second term denotes the cost of false alarms for class k. The average cost of misclassification for the entire tree per unit input data rate can be computed as:

$$C = \sum_{k=1}^{K} C^k \ . \qquad (118)$$

To determine $\overline{t}^k$ and $\overline{g}^k$, we model the impact of filtering at classifier v_i^k on the received data stream at classifier v_i^{k+1} by a conditional a priori probability $\phi_{v_{i+1}^k}^{e_{i+1}^k, e_i^k}$, where $\left(e_{i+1}^k, e_i^k\right)$ represents the four possible combinations (0,0), (0,1), (1,0), and (1,1) corresponding to the “yes” and “no” answers along the two successive branches [6]. The throughputs $\left(t_{v_{i+1}^k}^0, t_{v_{i+1}^k}^1\right)$ and goodputs $\left(g_{v_{i+1}^k}^0, g_{v_{i+1}^k}^1\right)$, outputted by classifier v_i^k can be

[6] For a tree topology only one data stream enters each classifier. Hence we do not need to include v_i^k while parameterizing ϕ .

computed from $t_{v_i^k}^{e_i^k}$ and $g_{v_i^k}^{e_i^k}$ using a set of recursive relationships described by the following transfer matrices:

$$\begin{bmatrix} t_{v_{i+1}^k}^{e_{i+1}^k} \\ g_{v_{i+1}^k}^{0} \end{bmatrix} = \mathbf{T}_{v_{i+1}^k}^{e_{i+1}^k} \begin{bmatrix} t_{v_i^k}^{e_i^k} \\ g_{v_i^k}^{0} \end{bmatrix}, \tag{119}$$

where
$$\mathbf{T}_{v_{i+1}^k}^{0} = \begin{bmatrix} \left(p_F^0\right)_{v_{i+1}^k} & \phi_{v_{i+1}^k}^{0,e_i^k}\left(\left(p_D^0\right)_{v_{i+1}^k} - \left(p_F^0\right)_{v_{i+1}^k}\right) \\ 0 & \phi_{v_{i+1}^k}^{0,e_i^k}\left(p_D^0\right)_{v_{i+1}^k} \end{bmatrix}, \tag{120}$$

and
$$\mathbf{T}_{v_{i+1}^k}^{1} = \begin{bmatrix} \left(p_D^1\right)_{v_{i+1}^k} & \left(\phi_{v_{i+1}^k}^{1,e_i^k} - 1\right)\left(\left(p_D^1\right)_{v_{i+1}^k} + \phi_{v_{i+1}^k}^{0,e_i^k}\left(p_F^1\right)_{v_{i+1}^k}\right) \\ 0 & \phi_{v_{i+1}^k}^{1,e_i^k}\left(p_D^1\right)_{v_{i+1}^k} \end{bmatrix}. \tag{121}$$

Note that, in the above expressions the "yes" (0) and "no" (1) output edges have different transfer matrices due to classifier exclusivity [109][7]. Hence, the throughput expression for the ``no" output branch consists of three separate terms that correspond to different types of data: bad data from classifier v_i^k that is correctly rejected by classifier v_{i+1}^k, good data from classifier v_i^k that is falsely rejected by classifier v_{i+1}^k, and good data from classifier v_i^k that is correctly rejected by classifier v_{i+1}^k. Using the recursive relationships, the end-to-end throughput and goodput $\overline{t}^k$, $\overline{g}^k$ for class k can be computed as:

$$\begin{bmatrix} \overline{t}^k \\ \overline{g}^k \end{bmatrix} = \left(\prod_{i=1}^{N_k} \mathbf{T}_{v_i^k}^{e_i^k}\right)\begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{122}$$

Similarly, the end-to-end a priori probability $\overline{\pi}^k$ for class k is given as:

$$\overline{\pi}^k = \prod_{i=1}^{N^k} \phi_{v_i^k}^{e_i^k, e_{i-1}^k} . \tag{123}$$

5.3.2 Resource Consumption and Constraints

Due to the high complexity operations that need to be performed by each classifier on each data object, limited computational resources in the system impose a heavy constraint on the performance of the classifier tree when the volume of the incoming data stream is large. Since each classifier generally performs the same set of functions

[7] Note that for the tree topology, exclusivity in the classifiers implies $\phi_{v_{i+1}^k}^{0,1-e_i^k} = 1 - \phi_{v_{i+1}^k}^{1,1-e_i^k} = 0$.

on each data object, we model computational resource requirements for each individual classifier as being directly proportional to the rate of data entering it. Hence, we may define its resource consumption as:

$$r_{v_i^k} = \alpha_{v_i^k} t_{v_{i-1}^k}^{e_{i-1}^k}. \tag{124}$$

where $\alpha_{v_i^k}$ is the amount of resources required per unit rate for classifier v_i^k. Since some of the classifiers for each class overlap, i.e. for multiple classes k, $v_i^k = n$, we concisely denote the entire tree configuration vector by indexing the configuration for each output edge of each of the N classifiers:

$$\mathbf{p}_F = \left[\left(p_F^0 \right)_1, \left(p_F^1 \right)_1, ..., \left(p_F^0 \right)_N, \left(p_F^1 \right)_N \right]. \tag{125}$$

and the resulting resource consumption vector for each classifier:

$$\mathbf{r}(\mathbf{p}_F) = [r_1, ..., r_N]^T. \tag{126}$$

Suppose that each classifier is uniquely placed on one of M different processing nodes. Let $A_{M \times N}$ be the binary node assignment matrix that maps each classifier onto a processing node, with:

$$A_{m,n} = \begin{cases} 1 & \text{if classifier } n \text{ is placed on node } m \\ 0 & \text{otherwise} \end{cases}. \tag{127}$$

Given that processing node m has R_m available resources, any feasible configuration of classifiers in the tree needs to satisfy the constraint:

$$\mathbf{Ar}(\mathbf{p}_F) \leq \mathbf{R}, \text{ where } \mathbf{R} = [R_1, ..., R_M]^T. \tag{128}$$

This problem can be solved using convex programming techniques such as sequential quadratic programming [144]. However, due to its non-convexity, it is often necessary to try different starting points for the algorithm to converge to the globally optimal solution. In practice, due to the sharpness of the DET curve, the global minimum can often be found by selecting a starting point near the origin, $\mathbf{p}_F = \mathbf{0}$.

5.3.3 Discussion: Single versus Multiple Operating Points per Classifier

Recall that using multiple operating points decouples the two output rates by configuring each output class separately, i.e. p_F^0 and p_F^1 are configured independently. This enables flexibility in terms of allowing data to be intelligently discarded or replicated across both branches based on the cost functions. For example, suppose a classifier uses thresholding on the resulting data prediction scores and forwards all data with scores above 0 across the ``yes" branch, and all data with scores below $d$ across the ``no" branch. If $d < 0$, all data between d and 0 is dropped from both branches, leading to load shedding. If $d > 0$, all data between 0 and d is transmitted across both branches, leading to replication.

On the other hand, when a classifier usesone threshold or operating point, the sum of output rates for each classifier is equal to the input rate entering it. Under tight resource constraints, such a strategy may not be feasible, and may require the downstream classifiers to discard input data
using ``arbitrary load shedding". Arbitrary load shedding effectively moves the operating point below the DET curve, towards the origin $(p_F, p_D) = (0,0)$. Using multiple operating points, on the other hand, can always outperform ``arbitrary load shedding" by allowing each output edge to be configured independently. The proof is obvious, since for any point below the DET curve for an output edge, moving the point to the left (until it intersects the DET curve) reduces the false alarm probability while maintaining the same detection probability, thereby decreasing the overall cost. Moreover, this move strictly reduces the throughout for the respective edge, and hence downstream resource constraints are always feasibly met.

In Figure 35, we highlight four different centralized algorithms for comparison in simulations: 3 using a single operating point per classifier, and 1 which uses multiple operating points. The algorithms are as follows:

- *Algorithm A* uses the equal error rate (EER) configuration for each classifier, i.e. the point where the DET curve crosses the line. This ensures that the probability of false alarm, and the probability of misses across both output edges are equal. This may seem an intuitive approach when the costs are equal for all classes.
- *Algorithm B* determines the operating point of each classifier to minimize the overall cost without considering resource constraints. Consequently, whenever a classifier is

overloaded, arbitrary load shedding brings the effective operating point below the DET curve.

- *Algorithm C* uses a single operating point for each classifier as in *Algorithm B*, but jointly determines the point on the DET curve, and the percentage of output load to shed (randomly) across each branch, such that the resulting resource consumption is feasible, and the overall cost is minimized.
- *Algorithm D* selects multiple operating points per classifier to independently filter and replicate data across each output edge.

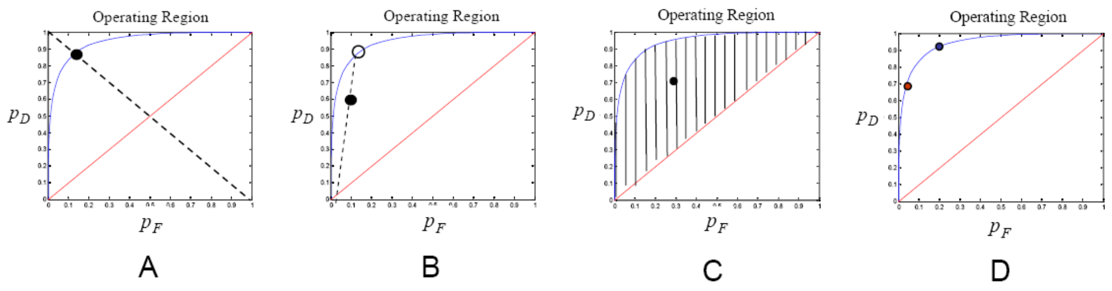


Figure 35 Pictorial representations of the configuration choices based on algorithms A-D.

While solving a centralized SQP problem is tractable for small classifier trees, since only a few classifier configurations need to be optimized, the complexity both in terms of informational and computational overhead can be very high for a large system with many classifiers, especially when the system needs to be reconfigured frequently in a dynamic environment. Moreover, centralized algorithms have a single point of failure, such that if the central controller fails, the system can no longer adapt to time-varying data streams or nodal resource constraints. As a practical alternative to the centralized approach, we propose several distributed approaches to enable classifiers in the tree to iteratively adapt to locally optimal configurations based on local information exchanges and low complexity operations.

5.4 Experimental Results

5.4.1 Application Scenario: Classifying Sports Images

We performed experiments by applying our algorithms to a tree topology of classifiers constructed for a sports image retrieval system [148]. Based on the natural hierarchy in data characteristics, we constructed a classifier tree given in the introduction (Figure 32). Each classifier is implemented as a support vector machine (SVM) trained specifically to the characteristic it detects, and uses up to 82 features, with complexity on the order of ~4000-21000 support vectors. The DET curves for individual classifiers were experimentally measured by testing the classifier on a set of images disjoint from the training set. Based on simulations, we observe that the complexity of processing one image is approximately proportional to the number of support vectors. In Table 12, we list the approximate amount of processing complexity (normalized) per image for different classifiers [8]. Here, C is a normalization constant for the complexity, given by the product of the average time of processing each image, and the speed of the processor. The total image set consists of approximately 20000 sports image scenes that are streamed at a data rate of 1 image per second.

Table 12 Processing complexity per image for each classifier.

Classifier:	*Complexity:*	*Classifier:*	*Complexity:*
Team Sports	$0.3884 \times C$	**Winter Sports**	$0.3199 \times C$
Baseball	$0.1761 \times C$	Ice Sports	$0.2223 \times C$
Little League	$0.1307 \times C$	Skating	$0.2403 \times C$
Basketball	$0.0772 \times C$	Skiing	$0.2608 \times C$
Cricket	$0.2006 \times C$	**Racquet Sports**	$0.1276 \times C$
		Tennis	$0.1720 \times C$

5.4.2 The Effect of Resource Constraints on Classifier Configurations

We tested the 4 algorithms for 3 different types of system conditions and placements under equal cost for misses and false alarms. The first type assumes system resources are

[8] We note that each image also required a one-time step for 82-dimensional feature extraction before support vectors could be used to obtain prediction scores for each classifier. We assume that the extraction is performed on a given node, and the remaining resources on that node is used for the classification process.

abundant and hence rate constraints (and placement) do not need to be considered while configuring classifiers. The second type involves placing classifiers on heavily resource constrained processing nodes in a manner that reduces cross-talk between nodes, i.e. traffic across the network. The third type involves placing classifiers on nodes in a hierarchical fashion to enable fault tolerance, where more important (upstream) classifiers are placed on more reliable nodes. The placements for types 2 and 3 are shown in Figure~\ref{fig:ExpClassifierPlacement}.

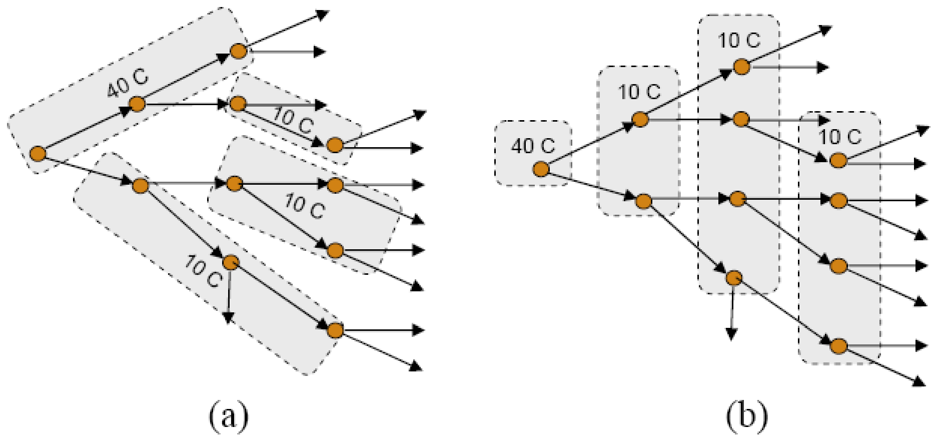


Figure 36 Placement of classifiers (a) to minimize cross-talk between nodes, and (b) to ensure some level of failure resiliency. Note that different nodes have different processing constraints. The constraints are measured in terms of the processor speed (in cycles/second).

Table 13 Costs of algorithms under different resource constraints and classifier placements

Algorithm	*No Resource Constraints*	*Placement in Fig.4a*	*Placement in Fig.4b*
A	1.9563	1.2971	1.3604
B	0.7742	0.9226	0.9442
C	0.7907	0.9158	0.8964
D	0.6959	0.8640	0.8419

To evaluate our algorithms, we ran *Algorithms B-D* using sequential quadratic programming from 50 different randomized starting points, and provided the minimum costs incurred by the application over all trials in Table 13. Note that *Algorithms B and C* show significant reduction in cost over the seemingly intuitive EER configuration for all scenarios. In the non-resource constrained case, the cost of EER was approximately 2.5 times that of *Algorithms B and C*. For resource constrained cases, the cost of EER was 40-50\% greater than *Algorithms B and C*.

Note that under resource constraints, *Algorithm C* shows minor improvement over *Algorithm B*, since it explicitly configures classifiers based on knowledge of the utility reduction as a result of load shedding. However, when resource constraints are loose, there is no benefit using *Algorithm C* since no load needs to be shed. In all cases, enabling multiple operating points (*Algorithm D*) saves 6-12% in cost over the *Algorithms B and C* due to intelligent filtering and replication.

5.4.3 Effect of Unequal Costs on Classifier Configurations

To further highlight the benefit of multiple operating points, we consider the effects of different cost functions on the performance of each algorithm. In the first scenario, we set the cost functions to be $c_M^k = 1$, $c_F^k = 4$ for all classes k. In the second scenario, we set the cost functions to be $c_M^k = 4$, $c_F^k = 1$ for all classes. The experiments were performed under loose resource constraints to highlight the effects of cost.

Table 14 Costs associated with various algorithms under different cost functions.

Alg.	$c_M^k = 1$, $c_F^k = 4$	$c_M^k = 4$, $c_F^k = 1$
A	3.8906	3.8906
B	1.9356	1.9355
C	0.9655	1.9365
D	0.8703	1.5438

Table 14 depicts the gains derived for each type of cost metric. For high costs of false alarms, we discovered significant savings when load shedding at the output was considered (*Algorithm C*). The reason for this large gain is that, unlike *Algorithm B*, which always keeps the entire output load from each classifier, *Algorithm C* can completely shed the output load whenever the quality of decision (e.g. goodput to throughput ratio) falls below a certain threshold. In our simulations, the load was completely shed by *Algorithm C* at the edges going into classifiers "Winter Sports" and

"Cricket". Nevertheless, using multiple operating points performed the best because rather than shedding the entire load down certain branches, it could successively shed data objects along each path that had poor quality of decision.

For high costs of misses, a huge decrease in cost resulted from using multiple operating points (approximately 21%). This is due to the intelligent replication of data, which reduces the probability of miss for each class. For example, we discovered that approximately 18% of the data from "Team Sports" was replicated and transmitted along both the “yes” and “no” output edges, while 10% of the data from "Baseball" was replicated, and 9% of data from "Winter Sports" was replicated.

5.5 Conclusion and Motivation for Learning Solutions

In this chapter, we introduced the paradigm of jointly configuring binary classifier trees to minimize misclassification costs under resource constraints. By using multiple thresholds for each classifier, we showed that significant cost savings can be achieved through the intelligent filtering and replication of data for semantic trees of classifiers.

In the following section, we will also consider the same method of configuring operating points for classifiers in cascaded topologies, but in an autonomous, informationally-distributed environment. In particular, suppose two autonomous sites train classifiers separately that detect unrelated semantic features, or features with unknown relationships, e.g. an “outdoor” image and a “basketball” image. When the two sites do not share analytics, optimizing the overall performance requires distributed, multi-agent solutions for learning the optimal configuration, especially for dynamic streams. In the following two sections, we will discuss in detail the problems, limitations, and solutions for configuring such classifier topologies in informationally-distributed environments.

CHAPTER 6
Learning Solutions for Configuring Chains of Classifiers in Distributed Environments

6.1 Motivation and Introduction of Learning Solutions

The joint optimization of a chain of classifier located across autonomous sites is a very difficult problem, since the analytics used to perform successive classification/filtering may be separately trained and owned by different companies [122] [137]. These analytics may have complex relationships, and often cannot be unified into a single repository due to legal, proprietary, or technical restrictions [153] [155]. Another challenge is that data streams often have time-varying rates and characteristics and thus, they require frequent reconfiguration to ensure acceptable classification performance. The approach introduced in Chapter 5 only optimally configure classifiers under *fixed* stream characteristics [120], and can perform poorly when stream characteristics are highly time-varying.

In this section, we introduce two multi-agent learning solutions for configuring chains of classifiers in distributed systems. We first introduce an experimentation algorithm that enables classifiers to determine the optimal configurations for a static stream. Then we show that this algorithm can fit within a larger multi-agent optimization framework, shown in Figure 37. The framework combines modeling, distributed estimation and information gathering schemes, and multi-agent learning solutions, as follows:

- *Modeling:* By using Poisson models for the stream traffic at each classifier (since the system is typically networked), we introduce a utility metric for real-time stream processing applications that explicitly considers the throughput, classification accuracy, and end-to-end delay of filtered streams.

- *Estimation:* Important local information, such as the estimated a priori probabilities (APP) of positive data from the input stream at each classifier, and processing resource constraints, are gathered to determine the utility of the stream processing system. We introduce a method for distributed information gathering, where each classifier summarizes its local observations using a single scalar called the local utility metric [153]. The local utility metric can be exchanged between nodes in order to obtain an accurate estimate of the overall stream processing utility, while keeping the communications overhead low and maintaining a high level of information *privacy* across sites.
- *Reconfiguration:* Classifier reconfiguration can be performed by using an algorithm that analytically maximizes the stream processing utility based on the processing rate, accuracy, and delay. Note that while in some cases, a centralized scheme can be used to determine the optimal configuration [119], in informationally-distributed environments, it is often impossible to determine the performance of an algorithm until sufficient time is given to estimate the accuracy/delay of the processed data [153]. Such environments require the use of randomized or iterative algorithms that converge to the optimal configuration over time. However, when the stream is dynamic, it often does not make sense to use an algorithm that configures for the current time interval, since stream characteristics may have changed during the next time interval. Hence, having *multiple algorithms* available enables us to choose the optimal algorithm based on the expected stream behavior in future time intervals.
- *Modeling of Dynamics:* To determine the optimal algorithm for reconfiguration, it is necessary to have a model of stream dynamics. Stream dynamics affect the APP of positive data arriving at each classifier, which in turn affects each classifier's local utility function. In our work, we define a *system state* to be a quantized value over each classifier's local utility values, as well as the overall stream processing utility. We propose a Markov-based approach to model state transitions over time as a

function of the previous state visited and algorithm used. This model enables us to choose the algorithm that leads to the best expected system performance in each system state.

- *Rules-based Decision-making:* We introduce the concept of rules, where a rule determines the proper algorithm to apply for system reconfiguration in each state. We provide an adaptive solution for using rules when stream characteristics are initially unknown. Each rule is played with a different probability, and the probability distribution is adapted to ensure probabilistic convergence to an optimal steady state rule. Furthermore, we provide an efficiency bound on the performance of the convergent rule when a limited number of iterations are used to estimate stream dynamics (i.e. imperfect estimation). As an extension, we also provide an evolutionary approach, where a new rule is generated from a set of old rules based on the best expected utility in the following time interval based on modeled dynamics. Finally, we discuss conditions under which a large set of rules can be decomposed into small sets of local rules across individual classifier sites, which can then make autonomous decisions about their locally utilized algorithms.

This chapter is organized as follows. In Section 6.2, we review related works in distributed decision-making in dynamic environments. In Section 6.3, we introduce distributed classifier chains and propose a delay-sensitive utility function. In Section 6.4, we discuss a distributed information gathering approach to estimate the utility when each site is unwilling to share proprietary data. In Section 6.5, we introduce a multi-agent experimentation solution for optimally configuring classifiers under the aforementioned informational constraints. We provide the overarching rules-based framework in Section 6.6 for dynamic environments. Extensions to the rules-based framework, such as the decomposition of rules across distributed classifier sites, and evolving a new rule from existing rules, are discussed in Section 6.7. Simulation results from a speech classification application are given in Section 6.8, and some concluding remarks in Section 6.9.

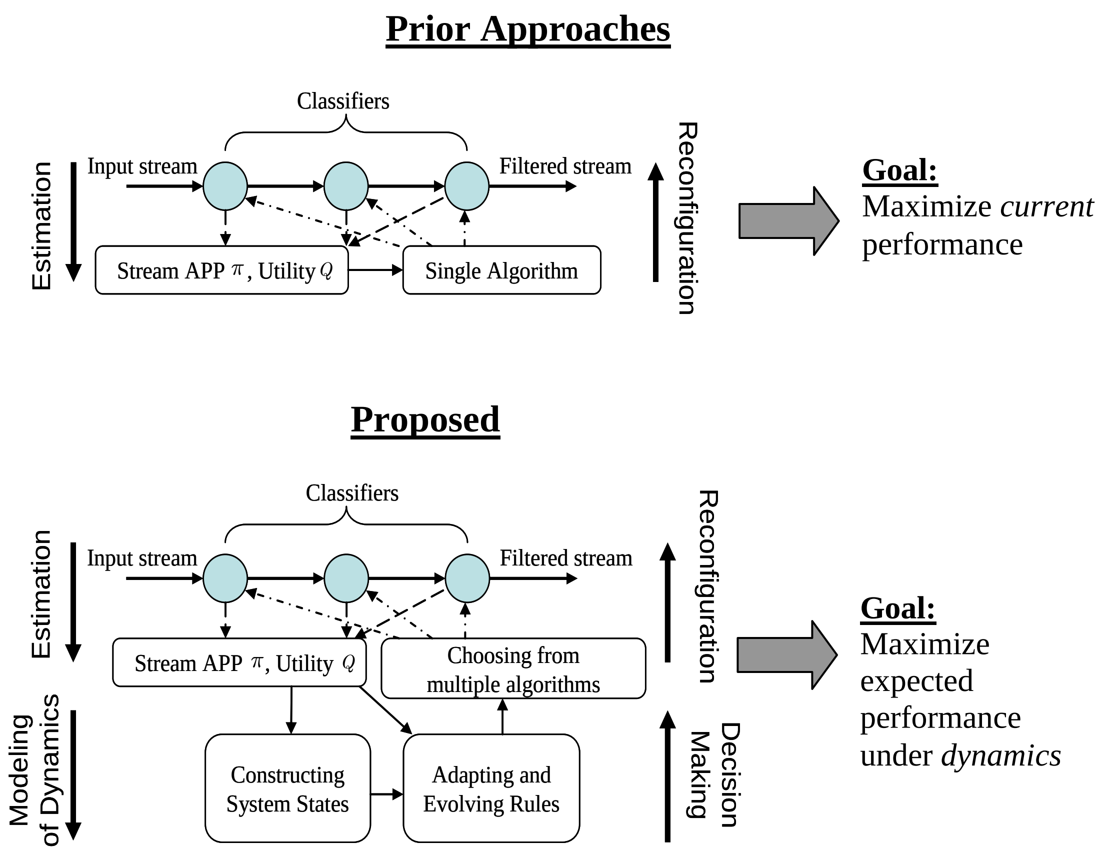

Prior Approaches
Classifiers
Input stream
Filtered stream
Estimation
Reconfiguration
Stream APP π, Utility Q
Single Algorithm
Goal:
Maximize *current* performance
Proposed
Classifiers
Input stream
Filtered stream
Estimation
Reconfiguration
Stream APP π, Utility Q
Choosing from multiple algorithms
Modeling of Dynamics
Constructing System States
Adapting and Evolving Rules
Decision Making
Goal:
Maximize expected performance under *dynamics*

Figure 37 Comparison of prior approaches and the proposed rules-based framework.

6.2 A Review of Related Works for Decision-making in Dynamics

A widely used framework for optimizing the performance of dynamic systems is the Markov decision process (MDP) [157], where a Markov model is used for state transitions as a function of the previous state and action (e.g. configuration) taken. In an MDP framework, there exists an optimal *policy* (i.e. a function mapping states to actions) that maximizes an expected *value function*, which is often given as the sum of discounted future rewards (e.g. expected utilities at future time intervals). When state transition probabilities are unknown, reinforcement learning techniques can be applied to determine the optimal policy, which involves a delicate balance between *exploitation* (playing the action that gives the highest estimated value) and *exploration* (playing an action of suboptimal value) [158].

While our rules-based framework is derived from the MDP framework (e.g. rules map states to algorithms while policies map states to actions), there is a key difference between traditional MDP-based approaches and our proposed rules-based approach. Unlike the MDP framework, where actions must be specified by quantized (discrete) configurations, algorithms are explicitly designed to perform iterative optimization over previous configurations [104]. Hence, their outputs are not limited to a discrete set of configurations/actions, but rather converge to a locally or globally optimal configuration over the real (continuous) space of configurations. Furthermore, algorithms avoid the complication involving how the configurations (actions) should be quantized in dynamic environments, e.g. when stream characteristics change over time.

Finally, there have been recent advances in collaborative multi-agent learning between distributed sites related to our proposed work. For instance, the idea of using a playbook to select different rules or strategies, and reinforcing these rules/strategies with different weights based on their performances, is proposed in [154]. However, while the playbook proposed in [154] is problem specific, we envision a broader set of rules capable of selecting optimization algorithms with inherent analytical properties leading to utility maximization of not only stream processing, but distributed systems in general. Furthermore, our aim is to construct a purely automated framework for both information gathering and distributed decision making, without requiring supervision, as supervision may not be possible across autonomous sites, or can lead to high operational costs.

6.3 A Delay-sensitive Utility Function for a Chain of Classifiers

The goal of a stream processing application is not only to maximize the amount of processed data (the *throughput*), but also the amount of data that is correctly processed by each classifier (the *goodput*). However, increasing the throughput also leads to an increased load on the system, which increases the end-to-end delay for the stream. In [153], we proposed a stream utility function to capture the tradeoff between system performance and the incurred delay for a chain of classifiers. The performance of each classifier v_i is estimated by a cost for misclassifying information, i.e. $\pi_i(1-(p_D)_i)+\theta_i(1-\pi_i)(p_F)_i$, where θ_i is the weight ratio between false positives and false negatives [132]. By inverting the cost, a metric for each classifier $F_i = g_i - \theta_i(t_i - g_i)$ can be used, which represents a performance reduction given by the weighted difference between the percentage of good and bad data forwarded by each classifier. Since exclusivity can no longer be assumed, the true end-to-end cost cannot be determined. Hence, we provided instead an approximation of the overall performance of the chain by the product of individual classifier performance reductions, $F = \prod_{i=1}^{n} F_i$ [153].

To factor in the delay, we multiply the resulting processing quality F with an end-to-end processing delay penalty $G(D) = e^{-\varphi D}$, where φ reflects the application's delay sensitivity [134] [135]. To determine $G(D)$, we follow the $M/M/1$ queuing model often used for networks and distributed stream processing systems [130] [131]. Denote the total SDO input rate, and the processing rate for each classifier v_i, by λ_i and μ_i, respectively. Note that each classifier acts as a filter that drops each SDO with i.i.d. probability $1-t_i$, and forwards the SDO with i.i.d. probability t_i to the next-hop classifier. Based on this model, the resulting output to each next-hop classifier is also given by a Poisson process [71], where the arrival rate of input data to classifier v_i is given by $\lambda_i = \lambda_0 \prod_{j=1}^{i-1} t_j$. Because the output of an $M/M/1$ system has i.i.d. interarrival times, the delays for each classifier in a classifier system, given the arrival and service rates, are also independent [128]. Hence, the expected delay penalty $G(D)$ for the entire chain can be calculated from the moment generating function [73]:

$$E[G(D)] = \Phi_D(-\varphi) = \prod_{i=1}^{n}\left(\frac{\mu_i - \lambda_i}{\mu_i - \lambda_i + \varphi}\right) \tag{129}$$

Thus, the overall utility of real-time stream processing, as given in our prior work [153], is:

$$Q\left(\mathbf{P}^F\right) = E[F \cdot G(D)] = F \cdot \Phi_D(-\varphi)$$
$$= \prod\nolimits_{i=1,\ldots,n} F_i \cdot \Phi_{D_i}(-\varphi) = \prod\nolimits_{i=1,\ldots,n} (g_i - \theta_i(t_i - g_i))\left(\frac{\mu_i - \lambda_i}{\mu_i - \lambda_i + \varphi}\right). \tag{130}$$

6.4 Informationally-Distributed Utility Calculation

6.4.1 Distributed Information Gathering

Note that while classifiers may be willing to provide information about $(p_F)_i$ and $(p_D)_i$, the APP π_i at every classifier v_i is, in general, a complicated function of the false alarm probabilities of all previous classifiers, i.e. $\pi_i = \pi_i(\mathbf{p}_F)_{j<i}$. This is because setting different thresholds for the false alarm probabilities at previous classifiers will affect the incoming source distribution to classifier v_i. Because analytics trained across different sites may not obey simple relationships (e.g. subsets), constructing a joint classification model is very difficult if sites do not share their analytics. Due to legal and proprietary restrictions, it can be assumed that in practice, the joint model cannot be constructed, and hence the objective function $Q(\mathbf{p}_F)$ is unknown.

While the precise form of $Q(\mathbf{p}_F)$ is unknown, and is most likely changing due to stream dynamics, the utility can still be *estimated* over a short time interval if classifier configurations are held *fixed* over the length of the interval. This is summarized in Figure 38 and discussed in more detail in [153]. First, the average service rate μ_i is fixed (*static*) for each classifier and can be exchanged with other classifiers upon system initialization. Second, the arrival rate into classifier v_i, λ_i, can be obtained by simply measuring (or *observing*) the number of SDOs in the input stream. Finally, the goodput and throughput ratios $\wp_i$ and ℓ_i are functions of the configuration $(p_F)_i$ and the APP. The APP can be estimated from the input stream using maximum a priori (MAP) schemes. Consequently, *every parameter* in (130) can be easily estimated based on some locally observable data. By exchanging these locally obtained parameters and configurations across all classifiers, each classifier can then estimate the overall stream processing utility.

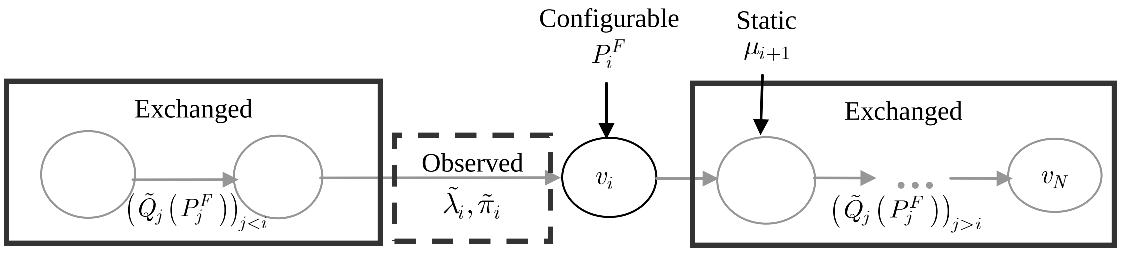

Configurable
P_i^F
Static
μ_{i+1}
Exchanged
$\left(\tilde{Q}_j\left(P_j^F\right)\right)_{j<i}$
Observed
$\tilde{\lambda}_i, \tilde{\pi}_i$
v_i
Exchanged
$\left(\tilde{Q}_j\left(P_j^F\right)\right)_{j>i}$
v_N

Figure 38 The various parameters in relation to v_i .

On the other hand, autonomous sites may have even stronger *privacy* requirements that prevent them from exchanging such local parameters (e.g. DET curves, configurations, etc). Alternatively, even if there are no such requirements, the classifiers may not be able to exchange this information due to the large communication overheads involved. To deal with the informationally-decentralized nature of the classifying system, as shown in [153], some calculations can be performed locally, and then a *local utility* metric scalar can be exchanged between classifiers. The local utilities are constructed by decomposing the utility function as follows:

$$\begin{aligned} Q(\mathbf{p}_F) &= \prod_{i=1}^{n}(g_i - \theta_i(t_i - g_i))\prod_{i=1}^{n}\left(\frac{\mu_i - \lambda_i}{\mu_i - \lambda_i + \varphi}\right) \\ &\approx \underbrace{\left(\frac{\mu_1 - \lambda_1}{\mu_1 - \lambda_1 + \varphi}\right)}_{\text{constant}}\prod_{i=1}^{n-1}\underbrace{\left(\tilde{g}_i - \theta_i\left(\tilde{t}_i - \tilde{g}_i\right)\right)\left(\frac{\mu_{i+1} - \tilde{\lambda}_i\tilde{t}_i}{\mu_{i+1} - \tilde{\lambda}_i\tilde{t}_i + \varphi}\right)}_{\text{known at } v_i}\underbrace{\left(\tilde{g}_n - \theta_n\left(\tilde{t}_n - \tilde{g}_n\right)\right)}_{\text{known at } v_n}, \\ &= K\prod_{i=1}^{n}\tilde{Q}_i\left((p_F)_i\right) \end{aligned} \tag{131}$$

where K is a constant, and the local utilities are given by:

$$\begin{aligned} &\tilde{Q}_i\left((p_F)_i\right) = \left(\tilde{g}_i - \theta_i\left(\tilde{t}_i - \tilde{g}_i\right)\right)\frac{\mu_{i+1} - \tilde{\lambda}_i\tilde{t}_i}{\mu_{i+1} - \tilde{\lambda}_i\tilde{t}_i + \varphi},\ 1 \le i < N, \\ &\tilde{Q}_N\left((p_F)_N\right) = \left(\tilde{g}_N - \theta_N\left(\tilde{t}_N - \tilde{g}_N\right)\right). \end{aligned} \tag{132}$$

Here, the symbol $\tilde{x}$ is used to indicate that the parameter or function x is obtained based on observation or estimation. Importantly, based on the observed arrival rate $\tilde{\lambda}_i$, the estimated APP $\tilde{\pi}_i$, the resource constraint/service rate of the next hop classifier μ_{i+1}, and the configuration P_i^F, each classifier v_i can compute $\tilde{Q}_i\left((p_F)_i\right)$ locally and exchange this scalar with other classifiers in the chain in order to compute the product of local utilities, which is the overall utility. Table 15 summarizes the various parameter types, their descriptions, and examples in our problem.

Table 15: Summary of parameter types and a few examples.

Type of parameter for v_i :	Description:	Examples:
Static Parameters:	Fixed parameters, exchanged during initialization	$\mu_j, j \neq i$
Observed Parameters:	Can be measured or estimated by classifier v_i	$\tilde{\lambda}_i$, $\tilde{\pi}_i$
Exchanged Parameters:	Traded with other classifiers	$\tilde{Q}_i((p_F)_i)$
Configurable Parameters:	Configured by classifier v_i	$(p_F)_i$

6.4.2 Optimization Framework using Distributed Information

In this subsection, we summarize our proposed framework for optimizing based on estimating the global utility during each time interval.

<u>Distributed Framework for Maximizing Stream Utility:</u>

1) Initialize the configuration $(p_F)_i^0$ at time 0 and exchange static parameters.

Each classifier v_i sets $(p_F)_i^0$ to some initial configuration.

2) For each iteration (or integer time) t, measure/estimate the arrival rate $\tilde{\lambda}_i(t)$ from the last elapsed control interval $[t-1,t)$.

The arrival rate from the previous-hop, $\tilde{\lambda}_i(t)$, is obtained by observing either the number of arrivals at classifier v_i during the interval $[t-1,t)$, or forming an estimate based on time-averaging, discounting, etc. of previous measurements.

3) Estimate the APP via maximum a posteriori (MAP) $\tilde{\pi}_i(t)$ from past time intervals.

Based on the location of the a posteriori points of arriving SDOs during time interval $[t-1,t)$, find the APP of each SDO and update $\tilde{\pi}_i(t)$ based on time-averaging, discounting, etc.)

4) For each classifier v_i, v_j, $j \neq i$, exchange local utilities $\tilde{Q}_j^t$ with all other classifiers just prior to time t to obtain an estimate of the $\tilde{Q}^t = \prod_{i=1}^{n} \tilde{Q}_i^t$.

5) Based on empirical analysis, modeling, experimentation, etc., choose the best configuration $p_F(t)$ to maximize the following:

$$(\mathbf{p}_F)^t = \arg\max_{\mathbf{p}_F} E\left[\tilde{Q}^{t+1}(\mathbf{p}_F)\right], \tag{133}$$

where (133) is the *predicted* stream utility at time $t+1$. Since the $(\mathbf{p}_F)^t$ affects the performance in the following interval $[t, t+1)$, solving (133) gives the (predicted) optimal configuration.

Of course, the key question is determining (133) for both static and dynamic environments. Because we do not know how a new configuration will affect the expected utility, learning solutions are needed to enable classifiers to converge to the optimal configurations.

6.5 A Distributed Learning Algorithm for Static Streams

6.5.1 Safe Experimentation for Discrete Configuration Sets

As an alternative to the model-based approaches, we first introduce a low-complexity, *model-free* learning approach called *safe experimentation* [138] for choosing the best configuration for classifiers (i.e. Step 5 of the distributed framework). Safe experimentation was first proposed for large, distributed, multi-agent systems, where each player is unable to observe the actions of all other players (due to informational or complexity constraints) and hence cannot build a model of other players. The player therefore adheres to a “trusted” action at most times, but occasionally “explores” a different action in search of a potentially better action. Essentially, safe experimentation does not require coordinating actions between players, or in our case, between autonomous sites.

Our stream processing system falls under such a category and is equivalent to a common interest game [139], where distributed classifiers (i.e. players) want to configure themselves (i.e. perform actions) to maximize the same utility function. The safe experimentation algorithm for reconfiguring classifiers is given as follows:

1) Initialization: At iteration $t = 0$, each classifier randomly selects a configuration $(\mathbf{p}_F)^0$ from a *discrete* action set A_i, which is set as the *baseline* configuration $\left(\mathbf{p}_F^b\right)^1$. After exchanging information about the derived local utilities from the initial configurations, each classifier’s baseline utility at iteration 1 is initialized as $u^b(1) = Q\left((\mathbf{p}_F)^0\right)$.

2) Configuration Selection: At each subsequent iteration, each player selects his baseline configuration with probability $(1-\varepsilon_t)$ or experiments with a new random configuration with probability ε_t. Hence, $(\mathbf{p}_F)^t = (\mathbf{p}_F^b)^t$ with probability $(1-\varepsilon_t)$, and $(\mathbf{p}_F)^t$ is chosen uniformly over A_i with probability ε_t. ε_t is denoted the *exploration rate* at iteration t.

3) Baseline Configuration and Baseline Utility Update: Each player compares the utility received, $Q((\mathbf{p}_F)^t)$, with his baseline utility $u^b(t)$, and updates his baseline configuration and utility as follows:

$$(p_F^b)_i^{t+1} = \begin{cases} (p_F)_i^t\ , & Q((\mathbf{p}_F)^t) > u_i^b(t) \\ (p_F^b)_i^t, & Q((\mathbf{p}_F)^t) \le u_i^b(t) \end{cases}, \tag{134}$$

$$u_i^b(t+1) = \max\left(u_i^b(t), Q\left(\mathbf{P}^F(t)\right)\right), \tag{135}$$

4) Return to step 2 and repeat.

The reason why this learning algorithm is called "Safe Experimentation" is because the baseline utility is non-decreasing with respect to time (or the number of iterations), and hence the performance of the algorithm only improves over time. We now provide a sufficient condition for finding the optimal solution.

Theorem 2: The following two conditions for the exploration rate ε_t are sufficient to guarantee that the Safe Experimentation algorithm for common interest games converges to the global optimal solution with probability 1:

$$\lim_{t\to\infty} \varepsilon_t = 0, \tag{136}$$

$$\lim_{t\to\infty} \prod_{\tau=1}^{t} \left[1 - \left(\frac{\varepsilon_\tau}{|A_1|}\right)\left(\frac{\varepsilon_\tau}{|A_2|}\right)\cdots\left(\frac{\varepsilon_\tau}{|A_n|}\right)\right] = 0. \tag{137}$$

Proof: The proof is similar to the proof for Theorem 3.1 in [138]. We provide a short sketch of the proof to highlight properties of the exploration rate. First, the exploration rate must converge to 0 as $t \to \infty$, as indicated by (136), such that the algorithm will play its baseline configuration with probability 1. Moreover, note that each multiplicative term in (137) represents the probability that the joint configuration played at time τ is *not* the optimal joint configuration, *or* all classifiers experimented at time τ. Hence, the left-hand side of (137) forms an upper bound on the probability that the optimal joint configuration is *not* played before time t. Thus Eq. (137) provides a sufficient condition for the optimal joint configuration to be eventually played with probability 1. ■

Note that we have shown (137) to be a *sufficient* (though not necessary) condition on the exploration rate for convergence to optimality, but only in a *discrete* action set. More importantly however, the proof provides no bounds for the convergence time of safe experimentation. In general, the method converges very slowly, and the expected time for finding the optimal solution can be bounded below by $|\mathcal{A}| = |A_1||A_2|...|A_n|$, which is the expected time for finding the optimal solution via i.i.d. uniform sampling (i.e. $\varepsilon_t = 1, \forall t$) of the action set. Because the action set for each classifier can be large when configurations are finely quantized, safe experimentation becomes impractical for dynamic environments where good configurations are promptly needed. In the next section, we will provide an alternative to safe experimentation for continuous configurations.

6.5.2 Combining Safe Experimentation with Randomized Local Search

To overcome the slowness of the convergence time for the discrete Safe Experimentation algorithm, as well as the suboptimality from choosing from a finite, discrete set of actions, we propose a simple stochastic algorithm that fits within the framework of two-phase methods [140]. In this approach, we combine a uniform random search for a baseline configuration over a continuous feasible set, and we perform a *randomized* local search algorithm around the baseline configuration. Unlike pure safe experimentation, which uses random sampling over the entire configuration space, the local search procedure take advantage of smoothness and continuity properties that exist in the global utility function, thereby allowing classifiers to converge to locally optimal points near their baseline configurations. The algorithm is given by modifying step 2 of safe experimentation as follows:

2) Configuration Selection: At each subsequent iteration, each player selects his baseline configuration with probability $(1-\varepsilon_t)$ or experiments with a new random configuration with probability ε_t. If the baseline configuration is selected, it is perturbed by a small random variable (e.g. Gaussian, uniform, etc.) $Z_i(t)$. Hence $(p_F)_i^t$ is chosen uniformly over the feasible configuration space with probability ε_t, and $(p_F)_i^t = \left(p_F^b\right)_i^t + Z_i^t$ with probability $(1-\varepsilon_t)$, where Z_i^t is a zero-mean random variable, with $\lim_{t\to\infty} Z_i^t = 0$.

If the size of the local search random perturbations do not decay too quickly (e.g. $E\left[\left|Z_i^t\right|^2\right] = K/t^2$, where $K > 0$ is a constant), it can be shown that (in a stationary setting) the local search algorithm converges to a local maximum with probability 1 [141]. Moreover, if the exploration rate is sufficiently high, the algorithm will converge to the globally optimal point with probability 1.

6.5.3 A Distributed Algorithm without Information Exchange

Recall that global utility estimation requires information exchange across all classifiers. What if such information could not be exchanged between classifiers? The best solution would be a simple distributed algorithm where each classifier v_i maximizes only its local utility function $\tilde{Q}_i\left(P_i^F\right)$.

Local Utility-based Distributed Algorithm:

1-3) Perform steps 1-3 in the distributed algorithm framework to obtain local utilities.

4) Based on the estimated parameters from past intervals, each classifier chooses $(p_F)_i^t$ to maximize its local utility function.

It can be shown that when stream characteristics are stationary, the local utility-based distributed algorithm converges to a fixed solution, since each classifier sequentially fixes its configurations (and hence the resulting APP and arrival rate into the next classifier), starting from P_1^F to P_n^F. However, the convergence point is suboptimal, since the distributed algorithm optimally configures each classifier based on a joint consideration of its own performance, and the delay penalty incurred on only its next-hop classifier, but does not consider the effect of its configuration on the performances and delays of classifiers further downstream. In the simulations section, we will compare the performance of the local utility-based distributed algorithm with safe experimentation to evaluate the gain achieved by the (albeit minimal) information exchange.

6.5.4 Summary of Algorithms, and Non-stationary Dynamics

In Table 16, a summary comparing the different algorithms is provided. Note that for the local utility-based distributed algorithm, the information overhead is zero for single-path topologies. However, as we will show in the next section, a minimal amount of information exchange is required for multiple-path topologies. For the combined Safe Experimentation and local search algorithm, it is impossible to provide useful sufficient conditions to guarantee both *fast and sure* convergence in a dynamic environment to a global maxima for arbitrary utility functions. However, it is useful to have a very high exploration rate at the beginning of the algorithm, such that a good baseline configuration can be found as a starting point for local search. During later iterations, a very low exploration rate is preferable, such that the local search algorithm can perform "refined" exploration around a good baseline point until stream characteristics change again. In the following section, we will discuss a framework that takes advantage of the experimentation and local search algorithms based on stream dynamics.

Table 16 Summary comparing the different algorithms and the various criteria

Algorithm	*Information Overhead*	*Complexity*	*Convergence/Dynamics*
Local utility-based	None for single-path topologies, very low for multiple-path topologies	Low (requires solving a single-variable optimization problem)	Very fast
Discretized Safe Experimentation	Global exchange of local utilities	Very low (random configuration, constant memory usage)	Slow, suboptimal
Continuous Safe Experimentation with Randomized Local Search	Global exchange of local utilities	Very low (random configuration, constant memory usage)	Fast; optimality can vary depending on experimentation frequency and decay rate of local search

6.6 A Rules-based Framework for Choosing Algorithms

6.6.1 States, Algorithms, and Rules

Now that we have discussed the estimation portion of our framework (Figure 37) and introduced a few algorithms, we move to discuss the proposed decision-making process in dynamic environments. We introduce the rules-based framework for choosing algorithms as follows:

- A set of *states* $\mathcal{S} = \{S_1,...,S_M\}$ that capture information about the environment (e.g. APPs of input streams to each classifier) or the stream processing utility (local or global), and can be represented by quantized bins over these parameters.
- The expected *utility* derived in each state S_m, $Q(S_m)$.
- A set of *algorithms* $\mathcal{A} = \{A_1,...,A_K\}$ that can be used to reconfigure the system, where an algorithm determines the configuration at time t, $(\mathbf{p}_F)^t$, based on prior configurations, e.g. $(\mathbf{p}_F)^t = A_k\left((\mathbf{p}_F)^{t-1},...,(\mathbf{p}_F)^{t-\tau}\right)$. Note that an algorithm differs from an action in the MDP framework [157] in that an action simply corresponds to a (discrete) fixed configuration. In fact, algorithms are generalizations of actions, since an action can be interpreted as an algorithm that always returns the same configuration regardless of the prior configurations, i.e. $A_k\left((\mathbf{p}_F)^{t-1},...,(\mathbf{p}_F)^{t-\tau}\right) = \mathbf{c}_k$, where $\mathbf{c}_k$ is some constant configuration.
- A set of *pure rules* $\mathcal{R} = \{R_1,...,R_H\}$. Each rule $R_h : \mathcal{S} \to \mathcal{A}$ is a deterministic mapping from a state to an algorithm, where the expression $R_h(S) = A \in \mathcal{A}$ indicates that algorithm A should be used if the current system state is S. Additionally, we introduce the concept of a *mixed rule* R, which is a random rule with a probability distribution over the set of pure rules $\mathcal{R}$, given by a probability vector $\mathbf{r} = [p(R_1),...,p(R_H)]^T$. For convenience, we denote a mixed rule by the dot product between the probability vector and the (ordered) set of pure rules, $\mathbf{r} \cdot \mathcal{R} = \sum_{h=1}^{H} \mathbf{r}_h R_h$, where $\mathbf{r}_h$ is the hth element of $\mathbf{r}$. As will be shown later, mixed rules are powerful for both proving convergence results, and for designing solutions

to find the optimal rule for algorithm selection when stream characteristics are initially unknown.

6.6.2 State Spaces and Markov Modeling for Algorithms

Markov processes have been used extensively to model the behavior of dynamic streams (such as multimedia) due to their ability to capture temporal correlations of varying orders [118] [156]. In this section, we extend Markov modeling to the space of algorithms and rules. (Though a Markov model may not be entirely accurate for relating stream dynamics to algorithms, we provide evidence in our simulations that for temporally-correlated stream data, the Markov model approximates the real process closely.) Importantly, based on Markov assumptions about algorithms and states, we can apply results from the MDP framework to show that the optimal rule for selecting algorithms in steady state is always pure. While this result is a simple consequence of the MDP framework, we provide a short proof below to guide us (in the following section) on how to construct a solution for learning the optimal pure rule under unknown stream dynamics. Moreover, the details in the proof will also enable us to prove efficiency bounds when stream parameters cannot be perfectly estimated.

Definition 1: Define a *first-order algorithmic Markov process* (or *algorithmic Markov system*) for a set of algorithms $\mathcal{A}$ and discrete state space quantization $\mathcal{S}$ as follows: the state and algorithm used at time t, $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, is a sufficient statistic for s_{t+1}. Hence, s_{t+1} can be described by a probability transition function $p(s_{t+1} \mid s_t, a_t) = p\big(s_{t+1} \mid s_t, a_t\left(\mathbf{P}_{t-1}^F, ..., \mathbf{P}_{t-\tau}^F\right)\big)$ for any past configurations $\left(\mathbf{P}_{t-1}^F, ..., \mathbf{P}_{t-\tau}^F\right)$.

Note that Definition 1 implies that in the algorithmic Markov system model, the state transitions are not dependent on the precise configurations used in previous time intervals, but only on the algorithm and state visited during the last time interval.

Definition 2: The *transition matrix* for a *pure rule* R_h over the set of states $\mathcal{S}$ is defined as a matrix $\mathbf{P}(R_h)$ with entries $[\mathbf{P}(R_h)]_{ij} = p\big(s_{t+1} = S_i \mid s_t = S_j, a_t = R\,(s_t)\big)$. The transition matrix for a mixed rule $\mathbf{r} \cdot \mathcal{R}$ is given by a matrix $\mathbf{P}(\mathbf{r} \cdot \mathcal{R})$ with entries:

$[\mathbf{P}(\mathbf{r}\cdot\mathcal{R})]_{ij} = \sum_{h=1}^{H}\mathbf{r}_h p(s_{t+1} = S_i \mid s_t = S_j, a_t = R_h(s_t))$, where the subscript h indicates the $h\text{th}$ component of $\mathbf{r}$. Consequently, the transition matrix for a mixed rule can also be written as $\mathbf{P}(\mathbf{r}\cdot\mathcal{R}) = \sum_{h=1}^{\mathcal{H}}\mathbf{r}_h\mathbf{P}(R_h)$.

Definition 3: The *steady state distribution* for being in each state S_m, given a rule R_h, is given by $p(s_\infty = S_m \mid R_h) = \lim_{t\to\infty}\left[\mathbf{P}^t(R_h)\cdot\mathbf{e}\right]_m$, where $\mathbf{e} = [1,0,...,0]^T$ [9]. This can be conveniently expressed as a *steady state distribution vector* $\mathbf{p}_{ss}(R_h) = \lim_{t\to\infty}\mathbf{P}^t(R_h)\cdot\mathbf{e}$.

Likewise, denote the utility vector for each state by $\mathbf{q}(\mathcal{S}) = [Q(S_1),...,Q(S_M)]^T$. The *steady-state average utility* is given by:

$$Q(\mathbf{p}_{ss}(R_h)\cdot\mathcal{S}) \triangleq \mathbf{p}_{ss}(R_h)^T\mathbf{q}(\mathcal{S}). \tag{138}$$

Lemma 1: The *steady state distribution for a mixed rule* can be given as a linear function of the steady state distribution of pure rules, $\mathbf{p}_{ss}(r\cdot\mathcal{R}) = \sum_{h=1}^{H}\mathbf{r}_h\mathbf{p}_{ss}(R_h)$. Likewise, the steady state average utility for a mixed rule can be given by $Q(\mathbf{p}_{ss}(\mathbf{r}\cdot\mathcal{R})\cdot\mathcal{S}) = \sum_{h=1}^{H}\mathbf{r}_h\mathbf{p}_{\text{ss}}(R_h)^T\mathbf{q}(\mathcal{S})$.

Proof: The steady state distribution vector for being in each state can be derived by the following sequence of equations.

$$\begin{aligned}\mathbf{p}_{ss}(r\cdot\mathcal{R}) &= \lim_{t\to\infty}\mathbf{P}^t(\mathbf{r}\cdot\mathcal{R})\cdot\mathbf{e} = \lim_{t\to\infty}\sum_{h=1}^{H}\mathbf{r}_h\mathbf{P}^t(R_h)\cdot\mathbf{e}\\ &= \sum_{h=1}^{H}\mathbf{r}_h\lim_{t\to\infty}\left[\mathbf{P}^t(R_h)\cdot\mathbf{e}\right] = \sum_{h=1}^{H}\mathbf{r}_h\mathbf{p}_{ss}(R_h).\end{aligned} \tag{139}$$

Likewise, the steady state average utility for a mixed rule can be given by:

$$\begin{aligned}Q(\mathbf{p}_{ss}(\mathbf{r}\cdot\mathcal{R})\cdot\mathcal{S}) &= \sum_{m=1}^{M}\left[\sum_{h=1}^{H}\mathbf{r}_h p_{\text{ss}}(s\mid R_h)\right]Q(S_m)\\ &= \sum_{h=1}^{H}\mathbf{r}_h\sum_{m=1}^{M}p_{\text{ss}}(S_m\mid R_h)Q(S_m) = \sum_{h=1}^{H}\mathbf{r}_h\mathbf{p}_{\text{ss}}(R_h)^T\mathbf{q}(\mathcal{S}).\end{aligned} \tag{140}$$

■

Proposition 9: Given an algorithmic Markov system, a set of pure rules $\mathcal{R}$, and the option to play any mixed rule $\mathbf{r}\cdot\mathcal{R}$, the optimal rule in steady state is always pure.[10]

[9] Note that the steady state distribution can be efficiently calculated by finding the eigenvector corresponding to the largest eigenvalue (e.g. 1) of transition matrix $\mathbf{P}(R_h)$.

Proof: The optimal mixed rule $\mathbf{r} \cdot \mathcal{R}$ in steady state maximizes the expected utility, which is obtained by solving the following problem:

$$\begin{aligned} &\max_{\mathbf{r}} Q(\mathbf{p}_{ss}(\mathbf{r} \cdot \mathcal{R}) \cdot \mathcal{S}) \\ &\text{s.t. } \textstyle\sum_{h=1}^{H} \mathbf{r}_h = 1, \\ &\mathbf{r} \geq \mathbf{0}. \end{aligned} \tag{141}$$

From Lemma 1, $Q(\mathbf{p}_{ss}(\mathbf{r} \cdot \mathcal{R}) \cdot \mathcal{S}) = \sum_{h=1}^{H} \mathbf{r}_h \mathbf{p}_{\mathrm{ss}}(R_h)^T \mathbf{q}(\mathcal{S})$, which is a linear transformation on the pure rule steady state distributions. Hence, the problem in (141) can be reduced to the following linear programming problem:

$$\begin{aligned} &\max_{\mathbf{r}} \sum_{h=1}^{H} \mathbf{r}_h \mathbf{p}_{\mathrm{ss}}(R_h)^T \mathbf{q}(\mathcal{S}) \\ &\textstyle\sum_{h=1}^{H} \mathbf{r}_h = 1, \\ &\mathbf{r} \geq \mathbf{0}. \end{aligned} \tag{142}$$

Note that the extrema of the feasible set are given by points where only one component of $\mathbf{r}$ is 1, and all other components are 0, which correspond to pure rules. Since an optimal linear programming solution always exists at an extremum, there always exists an optimal pure rule in steady state. ■

6.6.3 An Adaptive Solution for Finding the Optimal Pure Rule

We have shown in the previous section that an optimal rule is always pure under the Markov assumption. However, a mixed rule is often useful for estimating stream dynamics when the distribution of stream data values are initially unknown. For example, when a new application is run on a distributed stream mining system, there may not be any prior transmitted information about its stream statistics (e.g. average data rate, APPs for each classifier). In this section, we propose a solution called *Simultaneous Parameter Estimation and Rule Optimization* (SPERO), depicted in Figure 39. SPERO attempts to accomplish two important objectives. First, SPERO accurately estimates the state utilities and state transition probabilities, such that it can determine the optimal steady state pure rule from (142). On the other hand, SPERO utilizes a mixed rule that approaches the optimal rule in the limit, but also provides high performance during any

[10] Note that this proposition is proven in [157] for MDPs.

finite time interval. This is obtained by initializing the mixed rule to play each pure rule with equal probability, and then slowly *reinforcing* the probability of playing the pure

Solution 1: Simultaneous Parameter Estimation and Rule Optimization (SPERO)

1) Initialize state transition count, mixed rule count, and utilities for each state:

```
For all states and actions $s, s', a$ ,
   If there exists $R_h \in \mathcal{R}$ such that $R_h(s) = a$ ,
      Set state transition count $C(s', s, a) = 1$ .
   Else
      Set state transition count $C(s', s, a) = 0$ .
Set rule count $c_h := 1$ for all $R_h \in \mathcal{R}$ .
For all states $s \in \mathcal{S}$ , set state utilities $Q^{(0)}(s) := 0$ .
Set state visit counts $(v_1, ..., v_m) = (0, ..., 0)$ .
Set initial iteration $t := 0$ .
Determine initial state $s_0$ .
```

2) Choose a rule:

```
Select mixed rule $R^{(t)} = \mathbf{r} \cdot \mathcal{R}$ , where $\mathbf{r} = [\sqrt[M]{c_1}, ..., \sqrt[M]{c_H}]^T / \sum_{h=1}^{H} \sqrt[M]{c_h}$ .
Calculate $a_t = R^{(t)}(s)$ for current state $s$ .
```

3) Update state transition probability and utility based on observed new state:

```
Process stream for given interval, and update time $t := t + 1$ .
For new state $s_t = S_h$ , measure utility $\tilde{Q}$ .
   Set:    $Q^{(t)}(S_h) := v_h Q^{(t-1)}(S_h)/(v_h + 1) + \tilde{Q}/(v_h + 1)$ .
   Set:    $v_h = v_h + 1$ .
Update: $C(s_t, s_{t-1}, R^{(t-1)}(s_{t-1})) := C(s_t, s_{t-1}, R^{(t-1)}(s_{t-1})) + 1$ .
For all $s, s' \in \mathcal{S}$ , set: $p(s' \mid s, a) = C(s', s, a) / \sum_{s'' \in \mathcal{S}} C(s'', s, a)$ .
```

4) Calculate utilities that would be achieved by each rule, and choose best pure rule:

```
Calculate steady-state state probabilities $\mathbf{p}_{ss}(R_h)$ for pure
   rules.
Set $h^* := \arg\max_{h | R_h \in \mathcal{R}} \mathbf{q}^T \mathbf{p}_{ss}(R_h)$ , where $\mathbf{q} = [Q^{(t)}(S_1), ..., Q^{(t)}(S_M)]^T$ .
Update $c_{h^*} := c_{h^*} + 1$ .
```

rule that is expected to lead to the highest steady state utility, given the current estimation of state utilities and transition probabilities. As an example, the probability distribution of SPERO is shown in Figure 40 for a set of 8 rules used in our simulations (see Section 6.8, Approach B for more details). Note that the rule distribution is updated by reinforcing one rule at a time. Proof of steady state convergence to the optimal rule in SPERO is given in

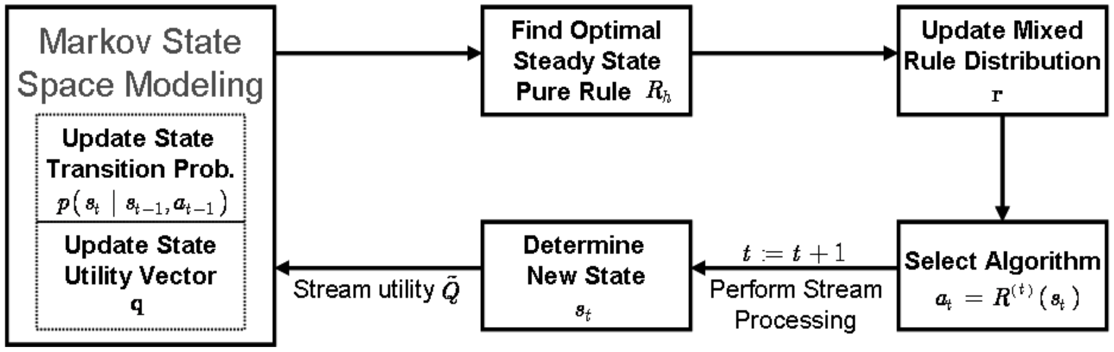

Markov State Space Modeling
Update State Transition Prob. $p(s_t \mid s_{t-1}, a_{t-1})$
Update State Utility Vector $\mathbf{q}$
Find Optimal Steady State Pure Rule R_h
Update Mixed Rule Distribution $\mathbf{r}$
Select Algorithm $a_t = R^{(t)}(s_t)$
$t := t+1$
Perform Stream Processing
Determine New State s_t
Stream utility $\tilde{Q}$

Figure 39 Flow diagram for updating parameters in Solution 1.

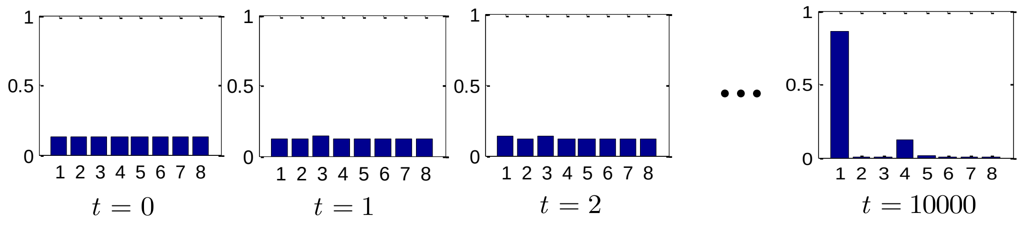


Figure 40 Rule distribution update in SPERO for 8 pure rules (see Section 6.8).

6.6.4 Tradeoff between Accuracy and Convergence Rate

In this section, we discuss the tradeoff between the estimation accuracy and the convergence rate of SPERO. In particular, SPERO uses a slow reinforcement rate to guarantee perfect estimation of parameters as $t \to \infty$. In practice however, it is often important to discover a good rule within a finite number of iterations, without continuing to sample rules that lead to states with poor performances. However, choosing a rule under finite observations can prevent the system from obtaining a perfect estimation of state utilities and transition probabilities, thereby converging to a suboptimal pure rule. In this section, we provide a probabilistic bound on the inefficiency of the convergent pure rule with respect to imperfect estimation caused by limited observations of each system state.

Consider when the real expected utility in a state is given by $Q(S_m)$, and the estimation based on time averaging of observations is given by $\hat{Q}(S_m)$. Depending on the variance of utility observations in that state σ_m^2, we can provide a probabilistic bound on achieving an estimation error of σ with probability at least $1-\frac{\sigma_m^2}{\sigma^2}$ using Chebyshev's inequality, i.e. $\Pr\left\{\left|Q(S_m)-\hat{Q}(S_m)\right| \geq \sigma\right\} \leq \frac{\sigma_m^2}{\sigma^2}$. Likewise, a similar probability estimation bound exists for the state transition probabilities, i.e. $\Pr\left\{\left|\mathbf{P}_{ij}(R_h)-\tilde{\mathbf{P}}_{ij}(R_h)\right| \geq \delta\right\} \leq \eta$. Both of these bounds enable us to estimate the number of visits required in each state to discover an efficient rule within high probability. We providing the following proposition and corollary to determine an upper bound on the expected number of iterations required by SPERO to discover a near optimal rule.

Proposition 10: Suppose that $\left|Q(S_m)-\hat{Q}(S_m)\right| \leq \sigma$ and $\left|\mathbf{P}_{ij}(R_h)-\tilde{\mathbf{P}}_{ij}(R_h)\right| \leq \delta$. Then the steady state utility of the convergent rule deviates from the utility of the optimal rule by no more than approximately $2M\delta(U_Q+2M\sigma)$, where U_Q is the average system utility of the highest utility state.

Proof: From [159], it is shown that if the entry-wise error of the probability transition matrices is δ, then the steady state probabilities for the estimated and real transition probabilities obey the following relation:

$$\frac{\left|p_{\text{ss}}(S_m \mid R_h)-\hat{p}_{\text{ss}}(S_m \mid R_h)\right|}{p_{\text{ss}}(S_m \mid R_h)} \leq \left(\frac{1+\delta}{1-\delta}\right)^M - 1 = 2M\delta + O\left(\delta^2\right). \tag{143}$$

Furthermore, since $p_{\text{ss}}(S_m \mid R_h) \leq 1$, a looser bound for the element-wise estimation error of $p_{\text{ss}}(S_m \mid R_h)$ can be given by $\left|p_{\text{ss}}(S_m \mid R_h)-\hat{p}_{\text{ss}}(S_m \mid R_h)\right| \leq \left(\frac{1+\delta}{1-\delta}\right)^M - 1 \approx 2M\delta$, where the $O\left(\delta^2\right)$ term can be dropped for small δ. Maximizing $\sum_{h=1}^{H} \mathbf{r}_h \hat{\mathbf{p}}_{\text{ss}}(R_h)^T \hat{\mathbf{q}}(\mathcal{S})$ in (142) based on estimation leads to a pure rule R_h (by Proposition 9) with estimated steady state utility that differs from the real steady state utility by no more than:

$$\begin{aligned}
&\left|\mathbf{p}_{\text{ss}}(R_h)^T \mathbf{q}(\mathcal{S}) - \hat{\mathbf{p}}_{\text{ss}}(R_h)^T \hat{\mathbf{q}}(\mathcal{S})\right| \leq \sum_{h=1}^{M}\left|p_{\text{ss}}(S_m \mid R_h) Q(S_m) - \hat{p}_{\text{ss}}(S_m \mid R_h)\hat{Q}(S_m)\right| \\
&\leq \sum_{h=1}^{M}\left|p_{\text{ss}}(S_m \mid R_h)-\hat{p}_{\text{ss}}(S_m \mid R_h)\right| \max\left(Q(S_m),\hat{Q}(S_m)\right) + p_{\text{ss}}(S_m \mid R_h)\left|Q(S_m)-\hat{Q}(S_m)\right| \\
&\leq MU_Q\delta + 2M^2\delta\sigma = M\delta(U_Q+2M\sigma)
\end{aligned} \tag{144}$$

Hence, the true optimal rule R^* will have estimated average steady state utility with an error of $M\delta(U_Q + 2M\sigma)$. The *estimated* rule $\hat{R}^*$, will have at least the same *estimated* average utility of the true optimal rule, and a true average utility within $M\delta(U_Q + 2M\sigma)$ of that value. Hence, combining the two maximum errors, we have the bound $2M\delta(U_Q + 2M\sigma)$ for differences between the performances of the convergent rule and the optimal rule. ■

Corollary 1: In the worst case, the expected number of iterations required for SPERO to determine a pure rule that has average utility within $M\delta(U_Q + 2M\sigma)$ of the optimal pure rule with probability at least $(1-\varepsilon)(1-\eta)$ is $O\left(\max_{m=1,\ldots,M}\left(1/\left(4n\delta^2\right), v_m^2/\left(\varepsilon\sigma^2\right)\right)\right)$.

Proof: $\max_{m=1,\ldots,M}\left(1/\left(4n\delta^2\right), v_m^2/\left(\varepsilon\sigma^2\right)\right)$ is the greater value between the number of visits to each state required for $\Pr\left\{\left|Q(S_m) - \hat{Q}(S_m)\right| \geq \sigma\right\} \leq \varepsilon$, and the number of state transition occurrences required for $\Pr\left\{\left|\mathbf{P}_{ij}(R_h) - \tilde{\mathbf{P}}_{ij}(R_h)\right| \geq \delta\right\} \leq \eta$. The number of iterations required to visit each state once is bounded below by the sojourn time of each state, which is, for recurrent states, a positive number τ. Multiplying τ by the number of state visits required to meet the two Chebyshev bounds gives us the expected number of iterations required by SPERO. ■

Note that we use big-O notation, since the sojourn time τ for each recurrent state is finite, but this can also vary depending on the system dynamics and the convergent rule.

6.7 Extensions of the Rules-based Framework

6.7.1 Evolving a New Rule from Existing Rules

Recall that SPERO determines the optimal rule out of a predefined set of rules. However, suppose that we lack the intuition to prescribe rules that perform well under *any* system state due to unknown stream dynamics. In this subsection, we propose a solution that evolves a new rule out of a set of existing rules.

Consider for each state S_m a set of *preferred algorithms* $\mathcal{A}_{S_m}$, given by the algorithms that can be played in the state by the set of existing rules $\mathcal{R}$. Instead of changing the probability density of mixed rule $\mathbf{r} \cdot \mathcal{R}$ through reinforcing each existing rule, we propose a solution called *Evolution From Existing Rules* (EFER), which reinforces the probability of playing each preferred algorithm in each state based on its expected performance (utility) in the next time interval. Since EFER determines an algorithm for each state that may be prescribed by several different rules, the resulting scheme is not simply a mixed rule over the original set of pure rules $\mathcal{R}$, but rather an *evolved rule* over a larger set of pure rules $\mathcal{R}'$.

Next, we present an interpretation on the evolved rule space. The rule space $\mathcal{R}'$ can be interpreted by labeling each mixed rule R over the original rule space $\mathcal{R}$ as a $M \times K$ matrix $\mathbf{R}$, with entries $\mathbf{R}(m,k) = p(A_k \mid S_m) = \sum_{h=1}^{H} \mathbf{r}_h \cdot \mathrm{I}(R_h(S_m) = A_k)$, and $\mathrm{I}(\ )$ is the indicator function. Note that for pure rules R_h, exactly 1 entry in each row m is 1, and all other entries are 0, and any mixed rule $\mathbf{r} \cdot \mathcal{R}$ lies in the convex hull of all pure rule matrices $\mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_H$ (See Figure 47 for a simple graphical representation.). An *evolved rule* R', on the other hand, is a mixed rule over a larger set $\mathcal{R}' \supset \mathcal{R}$, which has the following necessary and sufficient condition: each *row* of rule $\mathbf{R}'$ is in the convex hull of each *row* of pure rule matrices $\mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_H$.

Solution 2: Evolution From Existing Rules (EFER)

1) Initialize state transition count, prescribed algorithm probabilities, and utilities for each state:

```
For all states and actions $s, s', a$, set state transition count
    $C(s', s, a) = 1$.
Initialize algorithm probability count
    $c(S_m, A_k) := \sum_{h=1}^{H} \mathrm{I}(R_h(S_m) = A)$ for each $S_m$ and $A_k$.
For all states $s \in \mathcal{S}$, set state utilities $Q^{(0)}(s) := 0$.
Set state visit counts $(v_1, ..., v_m) = (0, ..., 0)$.
Set initial iteration $t := 0$.
Determine initial state $s_0$.
```

2) Choose an algorithm:

```
Select algorithm $A_k$ with probability $p(A_k) = c(s_t, A_k) / \sum_{\kappa=1}^{K} c(s_t, A_\kappa)$.
```

3) Update state transition probability and utility based on observed new state:

```
Process stream for given interval, and update time $t := t + 1$.
For new state $s_t = S_m$, measure utility $\tilde{Q}$.
    Set:   $Q^{(t)}(S_h) := v_h Q^{(t-1)}(S_h) / (v_h + 1) + \tilde{Q} / (v_h + 1)$.
    Set:   $v_h = v_h + 1$.
Update: $C(s_t, s_{t-1}, R^{(t-1)}(s_{t-1})) := C(s_t, s_{t-1}, R^{(t-1)}(s_{t-1})) + 1$.
For all $s, s' \in \mathcal{S}$, set: $p(s' \mid s, a) = C(s', s, a) / \sum_{s'' \in \mathcal{S}} C(s'', s, a)$.
```

4) Calculate the expected utility in the next time interval, and increment frequency of best algorithm in the last state:

```
If $Q^{(t)}(S_m) = \max\{Q^{(t)}(S_\eta) \mid \eta = 1, ..., H\}$,
Set $h^* := \underset{h \mid R_h \in \mathcal{R}}{\arg\max} \sum_{h=1}^{H} \sum_{s' \in \mathcal{S}} p(s' \mid s, R_h(s)) Q(s')$, where:
    $\mathbf{q} = [Q(S_1), ..., Q(S_M)]^T$.
Increment $c_{h^*} := c_{h^*} + 1$.
```

5) Return to step 2 and repeat.

An important feature to note about EFER is that the evolved rule is *not* designed to maximize the steady state expected utility. SPERO can determine the steady state utility for each rule based on its estimated transition matrix. However, no such transition matrix exists for EFER, since in the evolution of a new rule, there is no predefined rule to map each state to an algorithm, i.e. no transition matrix for an evolving rule (until it converges). Hence, EFER focuses instead on finding the algorithm that gives the best expected utility during the *next* time interval (similar to best response play [139]). In the simulations section, we will discuss the performance tradeoffs between SPERO and

EFER, where steady state optimization and best response optimization lead to different performance guarantees for stream processing.

6.7.2 A Decomposition Approach for Complex Sets of Rules

While using a larger state and rule space can improve the performance of the system, the complexity of finding the optimal rule in Solution 1 increases significantly with the size of the state space, as it requires calculating the eigenvalues of H different $M \times M$ matrices (one for each rule) during each time interval. Moreover, the convergence time to the optimal rule grows exponentially with the number of states M in the worst case! Hence, for a finite number of time intervals, a larger state space can even perform more poorly than a smaller state space (as we will show in our simulations).

To overcome the complexity issue, we propose a decomposition method that omits a subset of rules in order to reduce a large rule space into a collection of simple rules that can be decided autonomously by each classifier site. We define the decomposition methods below.

Definition 4: Consider a centralized state space model $\mathcal{S}$ for a system of n different sites. $\mathcal{S}$ is said to be *decomposable* if $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_n$, where $\mathcal{S}_i$ is a *local state space model* at site i. Likewise, $\mathcal{S}$ is *partially decomposable* if $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_n \times \mathcal{S}'$, where S' is a *shared state space model* that is contained in all local models. In other words, all local state space models are of the form $\mathcal{S}_i \times \mathcal{S}'$. Similarly, an algorithm space model is said to be decomposable if $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_n$, where the algorithm space $\mathcal{A}_i$ is the set of algorithms that can be used to reconfigure system parameters at site i.

Definition 5: A *decomposable rule space model* $\mathcal{R} = \mathcal{R}_1 \times \mathcal{R}_2 \times \ldots \times \mathcal{R}_n$ is given over a *decomposable* algorithm space model $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_n$ and a *partially decomposable* state space model $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_n \times \mathcal{S}'$, where each local rule in $\mathcal{R}_i$ maps a local state in $\mathcal{S}_i \times \mathcal{S}'$ to a local algorithm in $\mathcal{A}_i$.

Note that in a decomposed rule space model, each site has its own set of local rules and algorithms that it plays independently based on partial information (or a state space model using partial information) about the entire system. The notion of partial information has several strong implications. For example, a centralized rule space is not always decomposable, even when it is played over a decomposable algorithm and state space (See *Example 1* below.). Hence, there always exists centralized rules that can not be simulated by a decomposed approach. Furthermore, when the local state space models are not identical between each classifier, the classifiers converge to a Nash equilibrium [139] when running SPERO locally and independently, even when their payoffs are identical. While proof of convergence is a straightforward extension of *Proposition 11*, it is difficult to prove conditions under which the convergence point is optimal or suboptimal, since multiple Nash equilibria may exist [139]. In general, the convergent rule depends highly on the initial rules used in SPERO (see *Example 2* below). However, as we demonstrate in *Example 2*, the probability of converging to a suboptimal rule is also correlated with its efficiency, such that poor equilibria are reached with low probability.

Example 1 - When a rule cannot be decomposed: Consider a centralized state space given by 4 states consisting of quantized local utilities of a 2 classifier system. Each classifier has a "bad" state $S_{1,i}$ corresponding to $\tilde{Q}_i\left(P_i^F\right) < Q_{\text{thresh}}$, and a "good" state $S_{2,i}$ corresponding to $\tilde{Q}_i\left(P_i^F\right) \geq Q_{\text{thresh}}$. Each classifier can perform a local algorithm $A_{1,i}$ given by randomly choosing a new configuration (experimentation), or performing a local search $A_{2,i}$ around the last configuration, and to memorize the new configuration if it outperforms the old (See [153] for details.). A centralized rule space can consist of all rules $R : \mathcal{S}_1 \times \mathcal{S}_2 \to \mathcal{A}_1 \times \mathcal{A}_2$, while a localized rule space can only consist of rules of the form $R = \left(R^{(1)}, R^{(2)}\right)$, where $R^{(1)} : \mathcal{S}_1 \to \mathcal{A}_1$, and $R^{(2)} : \mathcal{S}_2 \to \mathcal{A}_2$. A decomposable rule is for each classifier to use experimentation in state $S_{1,i}$, and local search in state $S_{2,i}$. A non-decomposable rule is for each classifier to use experimentation in all states, unless both classifiers are in state $S_{2,i}$. As can be seen, to use non-decomposable rules, each classifier needs information about the states of both classifiers.

Example 2 - Convergence to a suboptimal equilibrium: Consider a simple scenario involving two classifiers $i = 1,2$, and two algorithms for each classifier, $A_{1,i}$ and $A_{2,i}$. The centralized model contains four states given by the combinations of algorithms used in the previous time interval. Suppose that when both classifiers perform action $A_{1,i}$, the utility of the system in the following time interval is 2. When both classifiers perform action $A_{2,i}$, the utility of the system is 1. Otherwise, the utility is 0. In the local model, each classifier measures only two states, where each state is given by the algorithm that it performed during the last interval i.e. $S_{1,i} = A_{1,i}$, $S_{2,i} = A_{2,i}$. Suppose that during the first 100 iterations, the following actions happen to be played: $(A_{1,1}, A_{1,2})$ with probability 1/100, $(A_{2,1}, A_{1,2})$ with probability 9/100, $(A_{1,1}, A_{2,2})$ with probability 9/100, and $(A_{2,1}, A_{2,2})$ with probability 81/100. (Note that these classifiers are probabilistically choosing algorithms independently.) Then for each classifier, the estimated utility of using algorithm $A_{1,i}$ is $1/10 * 2 = 1/5$, while the utility of using algorithm $A_{2,i}$ is $9/10$. Each classifier will thus continue to reinforce its own algorithm $A_{2,i}$, leading to a convergent suboptimal rule of using $(A_{2,1}, A_{2,2})$ with probability 1 (unless the state/action $(A_{1,1}, A_{1,2})$ is played a significant fraction of time to update the local utilities). Note that $(A_{2,1}, A_{2,2})$ is a Nash equilibrium, as well as the optimal $(A_{1,1}, A_{1,2})$.

Note that while in *Example 2*, suboptimal convergence is possible, the likelihood of suboptimal convergence to $(A_{2,1}, A_{2,2})$ is dependent on the utilities achieved in the two Nash equilibria. The greater the difference between the utilities of the Nash equilibria, the more unlikely the distributed approach is to converge to a suboptimal rule. For example, suppose that $Q(A_{1,1}, A_{1,2}) = \alpha > 1$, and $Q(A_{2,1}, A_{2,2}) = 1$, and the utility is zero otherwise. Then algorithm $A_{2,i}$ must be played with probability of at least $1 - 1/(\alpha + 1)$ in order for both classifiers to reinforce the suboptimal combination of algorithms $(A_{2,1}, A_{2,2})$. Hence, for large α, suboptimal convergence is unlikely to occur unless initial conditions are heavily weighted towards $(A_{2,1}, A_{2,2})$.

6.8 Simulation Results

6.8.1 Application: Classification of Speech Signals

We evaluate our framework for classifier reconfiguration using a stream consisting of utterances of Japanese vowels ‘a’ and ‘e’, given by cepstrum data collected from 8 different speakers (See [124] for more details.). By sampling different portions of the cepstrum data, we were able to cluster the speakers hierarchically, such that successive filtering using a chain of classifiers could be performed, i.e. to gather speech data from speaker 1, the first classifier splits the data into speaker groups {1,2,7,8} and {3,4,5,6}, the second classifier splits the filtered data {1,2,7,8} into {1,2} and {7,8}, and the final classifier splits {1,2} into {1} and {2}. For simple demonstration purposes, each classifier uses thresholding based on a low-complexity Gaussian mixture model generated from prior training data (240 speech samples). The test data consists of a mixed stream comprised of a total of 370 distinct speech data objects from the 8 speakers.

To simulate stream dynamics, we define a *rate of change* metric as follows: The rate of change refers to the maximum number of test samples randomly added or removed from each speaker class during each time interval (or iteration). For example, if the rate of change is 4, then during each time interval, a maximum of 4 speech signals from each speaker can be removed or added to the test data, where the number of speech data added or removed is uniformly distributed over this range. Note that adding and removing speech data affects the APP and the underlying stream distribution for each speaker class. Furthermore, we define *fractional* rates of change, where a rate of change of ½ means that during each time interval, there is ½ probability that a speech sample will be added or removed from each speaker.

While we allow the APPs to vary for each speaker class, for simplicity, we considered a fixed input stream rate normalized to 1 data object per second, and the normalized classifier processing rates in our experiments are set to $\mu_1 = 1.2$, $\mu_2 = 0.82$, and $\mu_3 = 0.67$ data objects per second, such that the stream imposes a medium to high load on each site. The application delay sensitivity parameter is set to $\varphi = 1.5$ (highly sensitive), and false alarm to miss ratio to $\theta_i = 0.2$ for each classifier, as this corresponds to approximately the equal error rate region when the entire chain is used to filter speaker 1 data.

6.8.2 Performance of Safe experimentation in Static Environments

We configured the operating points (thresholds) of the chain of 3 speech classifiers based on different classifier service rates and different application delay sensitivities. Given a source stream arrival rate close to 1, the highly loaded scenario involves classifier service rates of $\mu_1 = 1$, $\mu_2 = 0.45$, and $\mu_3 = 0.2$, while the low load scenario involves service rates $\mu_1 = 2$, $\mu_2 = 1.4$, and $\mu_3 = 1$. High and low delay sensitivities are set at $\varphi = 5$ and $\varphi = 2$ respectively, and false alarm to miss ratio for each classifier is set to $\theta_i = 0.5$. The results are shown in Table 20 for synthetic data and Table 18 for the real speech test data. In our safe experimentation solution, we used a local search Gaussian random variable with variance that decayed on the order of $O(1/t^2)$, where t is the iteration number. An approximate global maximum is also obtained by brute force sampling over 64000 joint configurations of the 3 classifiers. The local utility-based distributed algorithm is also compared. Finally, to show the gain achieved by the proposed algorithms, we considered an algorithm that simply maximizes the quality of classification for each SDO, and a probabilistic load shedding scheme [112] [113] is discard data if the average load exceeds moderately high levels (e.g. 0.6).

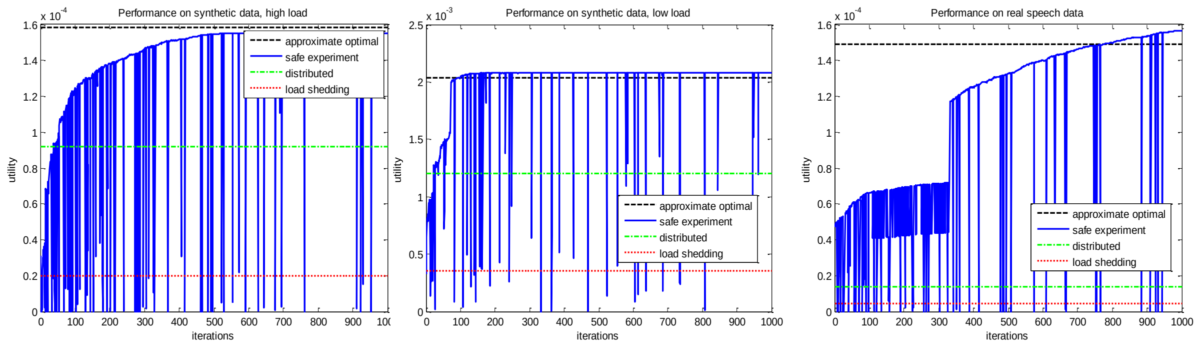
As can be seen from the results for both synthetic data and real speech data, the safe experimentation with random local search algorithm greatly outperforms the local utility-based distributed algorithm, which generally outperforms the probabilistic load shedding scheme. Note that the safe experimentation with random local search actually converges to a higher utility value than the approximate global maximum for synthetic data, since the global maximum approximation is discretized (albeit finely). In particular, it can be seen from Figure 41 that our algorithm uses less than 1000 iterations to find solutions that are almost as good if not *better* than the best possible configuration after uniformly quantizing the space with 64000 discrete points! On the other hand, safe experimentation alone would have required on average at least 64000 iterations to find the best possible *discrete* configuration (See Section 6.5.1.).

Table 17 Performance comparison for synthetic data, normalized to the global maximum.

	shedding	local utility	safe exp. (1000 iterations)	approx. global max
low load, low delay-sensitivity	0.1259	0.5806	0.9789	1.0000
high load, high delay-sensitivity	0.2701	0.5776	0.9999	0.9816

Table 18 Performance comparison for speech data test set.

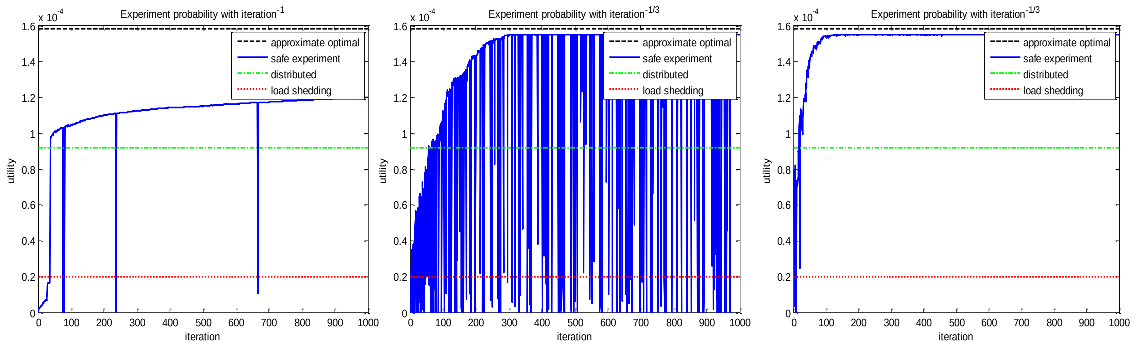
	shedding	local utility	safe exp. (1000 iterations)	approx. global max
low load, low delay-sensitivity	0.0900	0.0814	1.0000	0.9715
high load, high delay-sensitivity	0.0283	0.0893	1.0000	0.9556



(a) (b) (c)

Figure 41 Comparison of utility versus iteration for load-shedding, distributed, and safe experimentation algorithms under: (a) synthetic data, low load and low delay-sensitivity, (b) synthetic data, high load and high delay-sensitivity, (c) the real speech data stream. The exploration rates in these experiments were set to $1/\sqrt{t}$.

Secondly, we analyzed the adaptation time for our proposed experimentation and local search algorithm using the speech test data. In particular, we considered the cases where each classifier experiments with probability $1/t$, $1/\sqrt[3]{t}$, and $t^{-t/r}$, where r indicates the approximate number of explorations performed at the beginning. Some sample curves are shown in Figure 42 to highlight the rate of adaptation. In particular, note that the exploration rate is very low with $1/t$ and is insufficient for finding the global optimal utility. On the other hand, using $1/\sqrt[3]{t}$, which satisfies the minimal exploration rate condition in Theorem 2, is sufficient for finding the global optimal point, but the exploration rate decays very slowly, and even up to the 1000th iteration. Finally, for $t^{-t/50}$ (as shown in Figure 42c), frequent exploration is performed during the first few iterations, while local search dominates the later iterations. Our experiments verify our intuition that, during the first few iterations, it is important to explore frequently to find a good baseline configuration, while for later iterations, “playing it safe” by using local search performs better.



(a) (b) (c)

Figure 42 Comparison of adaptation times for exploration rates (a) $1/t$, (b) $1/\sqrt[3]{t}$, and (c) $t^{-t/50}$.

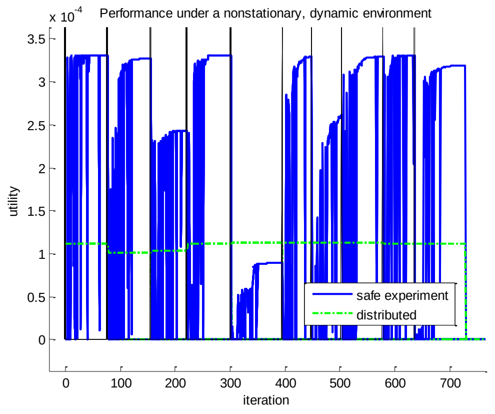


Figure 43 Dynamic adaptation results of Safe-Experimentation with exploration rate $t^{-t/50}$ for non-stationary streams. Vertical lines indicate the arrival of new stream characteristics.

In Figure 43, we show that a quickly decaying exploration rate (i.e. $t^{-t/50}$) is in fact a good heuristic for non-stationary streams. The vertical lines in the figure indicate when the stream is replaced by new source characteristics. To simulate a non-dynamic environment, the new source stream characteristics are chosen randomly by varying the number of samples selected from each speaker in the test data uniformly between 5 and 30. The interarrival times for each new stream is also uniformly distributed between 50 and 100 iterations. Note that in most cases, a high utility point can be found in very few iterations due to early exploration.

However, note that even in this simulation, the dynamics vary quite slowly. The safe experimentation algorithm would not be able to cope with stream characteristics that change significant during every time interval, as we will demonstrate in the following section.

6.8.3 State Space, Algorithms, and Rules used in Simulations under Dynamics

In our experiments, we use the following state space quantizations and algorithms listed below:

- *State space:* In our experiments, we associate four states S_1, S_2, S_3, S_4 with different levels of "minimum" utility given by 0, 6×10^{-4}, 1×10^{-3}, and 1.4×10^{-3} respectively. Note that the utilities are small due to the delay penalty factor, as well as the low a priori probability of speaker 1 stream data. The "minimum" utility levels merely determines bounds for being in each state and are not regarded as the average utilities estimated in each state. Furthermore, the state space can be divided into local states for each classifier that capture different ranges of local utilities. We used a low state 0 and a high state 0.1 for the local utilities of each classifier.
- *Algorithms:* The algorithm space consists of 4 algorithms modified from the solution proposed in [153]. Algorithm A_1 randomly chooses a new configuration for the classifier. A_2 samples a random configuration near its current best (or *baseline*) configuration, and if the utility increases with the new configuration, sets the new baseline configuration to the new configuration. Additionally, we use two algorithms A_3, A_4 to perform random experimentation in low P^F (below the equal error rate configuration) and high P^F (above the equal error rate configuration) regions of each classifier.

We will compare 3 different types of rules-based approaches.

- The first approach (*Experimentation*) involves a single fixed (but fairly efficient) rule, which performs algorithm A_1 when the system utility is below a threshold (7×10^{-4}), and algorithm A_2 otherwise. This approach is very similar to the one introduced for static environments, and has the lowest complexity of all the approaches.

- The second approach (*Small Rule Space*) uses a state space consisting of the 4 different levels of minimum utility, and a centralized algorithm that allocates identically to each and every classifier, one of the 4 algorithms. To map each state to an algorithm, 8 heuristic rules are used. SPERO is used to determine the optimal steady state rule.
- The third approach (*Distributed/Large Rule Space*) uses a large state space with 4 levels of utility, as well as 2 levels of local utilities for each classifier, totaling 32 states. Due to the high complexity and long convergence time of the centralized approach, we use decomposition by configuring each classifier independently using the 4 algorithms, leading to a total of $4^3 = 64$ possible algorithms. Finally, we consider 512 decomposable pure rules, where the rule space is a cross product between 8 local rules at each classifier. Note that the actual rule space at each classifier is similar to the second approach (8 states, 4 algorithms, 8 rules), although the combined centralized rule space is huge. SPERO is used at each classifier independently to learn the optimal local rule.

6.8.4 Comparison of Algorithms under Different Levels of Dynamics

In Figure 44, we display the average utilities achieved over the first 10000 time intervals of SPERO under different rates of change (given in Subsection 6.8.1). We discovered that the average performance of the first approach (experimentation) decreases as the rate of change increases, since changing stream characteristics requires the experimentation approach to randomly sample different points frequently when the utility level drops below the fixed threshold. In a highly dynamic case (e.g. rate of change equal to 12), the experimentation approach obtains an average utility of 8.88×10^{-4}. On the other hand, the second approach (small rule space) had a poorer average utility of only 7.42×10^{-4}. The poor performance can be attributed to the poor choice of rules, where out of the 8 rules, the rule that corresponds to the first approach actually outperforms all other rules. However, because SPERO performs random selection out of all 8 rules, and requires many iterations to converge, the average performance is poorer during the first 10000 iterations. Finally, we discovered that in the third approach (large rules-space), which we implemented in a distributed fashion across classifiers due to its high complexity, the rule space contained a convergent rule that significantly outperformed the optimal rule in the first two approaches. The average utility for the first 10000 iterations was 1.13×10^{-3}, about 27% higher than the experimentation approach. This is because the decomposed rule enables each classifier to model better the dynamics in its own local environment, which has a greater effect on its individual performance and delay.

On the other hand, for static or near static environments, we discovered that approaches 2 and 3 usually performed worse than experimentation. This is because, in slowly time-varying environments, the optimal rule in both the small and large rule spaces is in fact the experimentation rule. However, because of their slower learning rates, approaches 2 and 3 tend to perform more poorly during the first 10000 iterations while trying to discover (and reinforce) the experimentation rule.

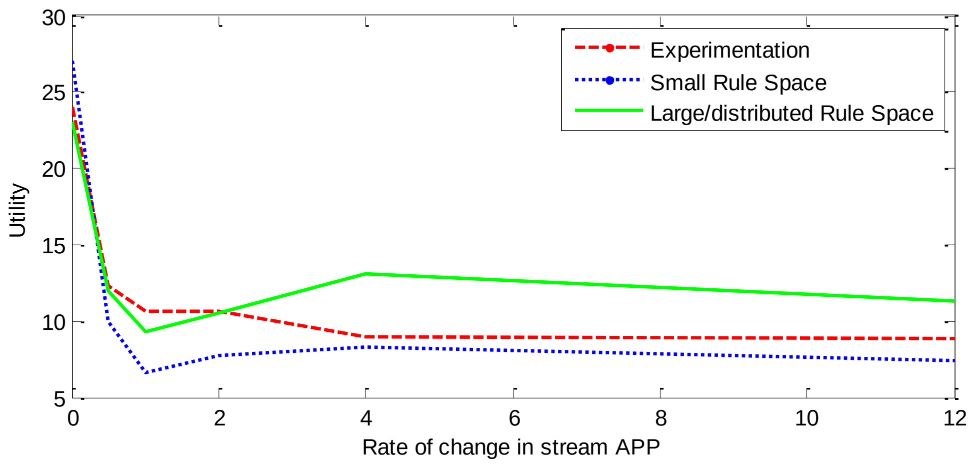


Figure 44 Comparison of utilities achieved by different rule spaces under different levels of dynamics.

Table 19 Average confusion matrices per time interval for a dynamic stream.

Approach	***Experimentation***		***Small Rule Space***		***Large Rule Space***	
	Lbled Spk 1	*Lbled Spk 2-8*	*Lbled Spk 1*	*Lbled Spk 2-8*	*Lbled Spk 1*	*Lbled Spk 2-8*
True Spk 1	4.21	13.54	10.15	7.93	11.95	6.12
True Spk 2-8	11.06	153.66	29.97	126.21	9.58	146.61
Average Delay	3.96 secs.		6.51 secs.		3.42 secs.	

Table 20 Probabilities of using local rules by each classifier for the distributed large rule space.

Classifier	*Rule 1*	*Rule 2*	*Rule 3*	*Rule 4*	*Rule 5*	*Rule 6*	*Rule 7*	*Rule 8*
1	0.05%	**88.6%**	0.01%	0.49%	8.71%	0.96%	0.02%	1.11%
2	0.64%	0.06%	4.29%	3.80%	0.07%	**91.0%**	0.01%	0.18%
3	**85.6%**	0.01%	0.32%	12.0%	1.69%	0.12%	0.06%	0.28%

To provide better intuition about the utilities achieved by each approach, we constructed a table of the confusion matrices and delays (see Table 19 and Figure 45) under a very dynamic environment (rate of change = 12). Note that the labeled speaker 2-8 simply refers to data that has been dropped (i.e. misses and true negatives). The misses can be attributed to both classifier inaccuracy, as well as discarding of low confidence data to ensure that correctly classified data is received with low delay. From Table 19, it can be seen that the experimentation approach performs very poorly, since whenever it obtains a configuration with high utility, the stream dynamics change within the next few time intervals, forcing the solution to perform random experimentation again. On the other hand, the small rule space had a better confusion matrix, but the utility suffered from the long end-to-end processing delay and high delay variance. This is due to periodically choosing suboptimal rules that operate at high false alarm regions even when the APP is high. Hence, the experimentation approach achieved a higher delay-sensitive utility than the small rule space. Finally, the large/distributed rule space provided the best performance as well as the lowest average delay and delay variance. As indicated by Table 20, each classifier converges toward a different local rule that is highly dependent on its accuracy and resource constraints. (In Table 20, rule 2 corresponds to approach 1, while other rules are mixtures of local search and random configurations in low and high false alarm regions.) Importantly, Table 20 shows that by decomposing 512 rules into 8 local rules at each classifier, SPERO converges quickly to a rule that performs well under dynamics.

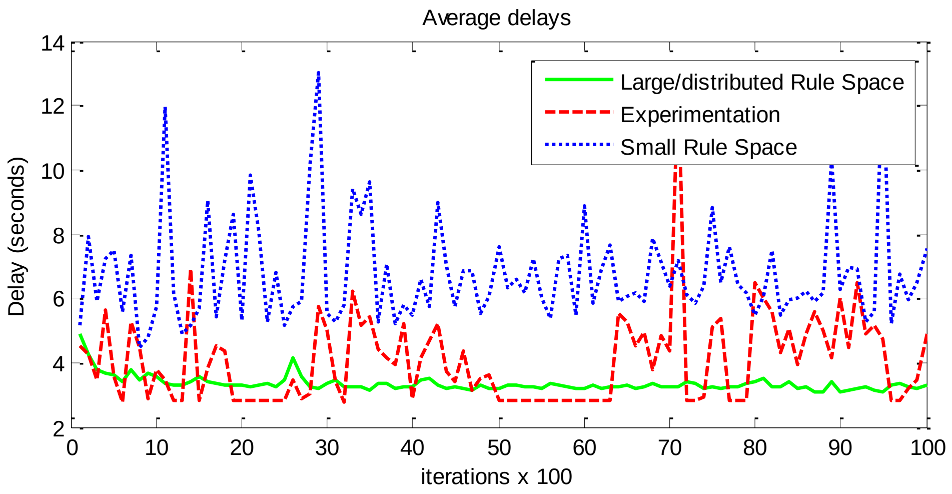


Figure 45 Comparison of delays for the 3 approaches. Each point is an average delay over 100 intervals.

6.8.5 Evaluation of the Markov Assumption for Algorithms

An important consideration is whether system dynamics are accurately modeled by the algorithmic Markov process given in Definition 1. To determine the sufficiency of information captured by the last state, we calculated the state transition probabilities for each algorithm conditioned on only the last state, versus the state transition probabilities conditioned on the last 2 states. The similarity between distributions obtained based on the last state, and the last two states, were evaluated for each algorithm using the average absolute difference between the estimated state transition probabilities. In other words, we evaluated a distance metric $\frac{1}{M}\sum_{s_t \in \mathcal{S}} |\Pr\{s_t \mid s_{t-1}, a_{t-1}\} - \Pr\{s_t \mid s_{t-1}, a_{t-1}, s_{t-2}\}|$ for each a_{t-1} and s_{t-2}. We discovered that for the centralized state space partitioned into 4 bins based on utilities, the (first-order) Markov model and the second order Markov model had transition probabilities that differed element-wise by no more than 0.04. This shows that the first-order Markov model is sufficient in capturing most of the information about the past two states, which provides higher confidence in the accuracy of Markov modeling for algorithms for the distributed classification system.

6.8.6 Evolution of a New Rule

In this section, we used EFER to evolve a new rule from the large/distributed rule space. We compared the performance of our evolved rule with the convergent distributed rule in the previous section and discovered that the average performance of the evolved rule was about 10% worse than that of the best prescribed rule in the large/distributed rule space. However, as shown in Figure 46 for a highly dynamic environment, EFER provides smaller utility fluctuations and guarantees a better minimum utility with high probability.

This phenomenon can be explained by how SPERO and EFER updates rules. Recall that in SPERO, the mixed rule was updated by reinforcing the pure rule with the highest *steady state* performance. Such a rule may perform well in certain states, but poorly in other states, since transients are ignored in this approach for the sake of maximizing the *average* performance. However, the evolved rule, which chooses an algorithm based on the expected performance in the *next time interval*, is more likely to discover a rule that performs well in each state, although not necessarily the rule that provides the optimal steady state performance.

Table 21 Average distance between a first order and second order Markov model.

State visited 2 intervals ago	*Avg. absolute distance compared to 1st Order Markov Model*
1	0.0032
2	0.0373
3	0.0353
4	0.0362

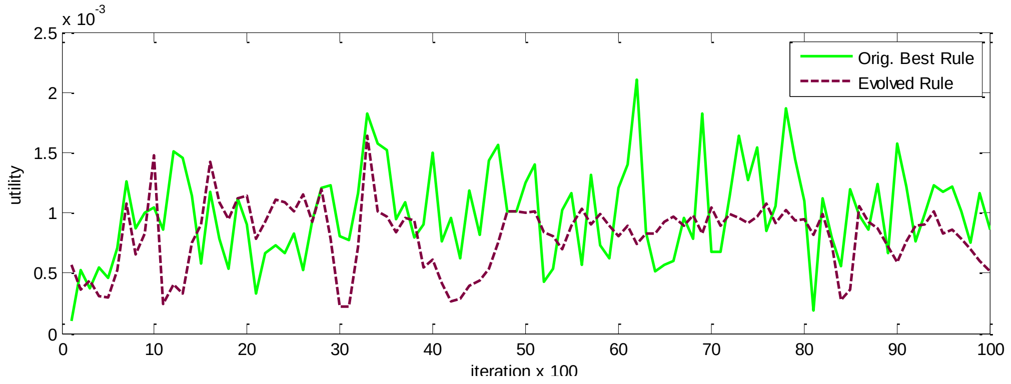


Figure 46 Comparison of utility achieved by the best rule in the original space, and the evolved rule.

Finally, note that the complexity of EFER is much less than SPERO, since EFER is not required to compute the eigenvalues for every pure rule matrix. Rather, it performs a single matrix-vector multiplication per algorithm, and chooses the best algorithm for the next time interval. From our experiments, the running time for determining the rule to reinforce during each iteration in SPERO was approximately 14.0ms, while the running time to determine the algorithm to reinforce in EFER was only 5.1ms. The savings become even more significant when the number of rules in SPERO becomes larger.

6.9 Conclusions

In this chapter, we proposed multi-agent learning solutions that fit within a rules-based framework for reconfiguring distributed classifiers for a delay-sensitive stream mining application with dynamic stream characteristics. By gathering information locally at each classifier and estimating local utility metrics, the framework employs rules based on models of the global system utility and transition probabilities between different states. We showed that the optimal rule can be chosen from a set of prescribed rules while accurately measuring parameters related to stream dynamics. Furthermore, we proposed a decomposition approach for reducing the complexity of the framework. Finally, we proposed a method to evolve a new rule based on prescribed rules. Using a chain of speech classifiers, we validated that large gains could be achieved by the proposed rules-based framework.

Note that while we used the classifier chain configuration problem as a key application to show the advantages of using rules to choose algorithms, the rules-based framework is not specific to the distributed stream mining problem and can be applied to various other dynamic and informationally-distributed systems. Importantly, when system dynamics are unknown, and intuition is insufficient for choosing the best algorithm to use for reconfiguration, the proposed methodology enables the system to adapt by learning the best algorithms to be played under different system conditions.

CHAPTER 7
Summary, Broader Impact, and Future Directions

To address the many challenges that arise from designing systems capable of efficiently supporting dynamic and/or distributed multimedia applications, this thesis proposes a systematic resource management framework that incorporates modeling, informationally-decentralized resource allocation mechanisms, and multi-agent learning solutions to optimally and fairly allocate and configure resources for multimedia applications. Here, we summarize several key insights obtained from this thesis.

First, modeling the dynamic behavior of applications (i.e. their time-varying utilities and workloads) is essential for improving the system performance, since this enables us to provide analytical estimates (bounds) for the resulting multimedia quality and the required computational costs for different algorithms configurations and implementations and, based on these estimates, predict and select the optimal application and system setting. For instance, in Chapter 2, we propose the first ever reported information-theoretic model relating rate, distortion, and complexity (R-D-C) for wavelet video compression. (Using the same methodology, such models can also be computed for other multimedia applications and algorithms.) Developing such R-D-C methods is important for existing and emerging multimedia streaming applications, where the transmitter or server needs to remotely determine the optimal configuration in terms of quality, rate and complexity (e.g. power utilization) that needs to be transmitted to the receiver. Since the video characteristics, channel conditions, and power constraints can change dynamically, the R-D-C model should be determined on-the-fly, even while the video is being encoded [11]. We show that our predictive approach to cross-layer optimization between applications and systems can significantly outperform existing solutions for joint application-system optimization, which are based on reactive (feedback) solutions.

Second, the information-theoretic modeling approach proposed in Chapter 2 can be used to *classify* sequences and frame types according to source characteristics (and hence, according to their complexity estimates). This differs from existing solutions for modeling complexity which rely on empirical models, which are constructed based on fitting experimental data, which assumes that the complexity requirements for video sequences do not change significantly over time or are known a priori. As discussed in Chapter 3, this classification scheme enables the encoder to transmit control messages that specify to the decoder which class of complexity distributions to use for predicting the application workload, thus enabling the decoder device to proactively adapt the system resource scheduling policies to optimize the performance of the application. The energy-saving benefit of using multiple classes of jobs to model and forecast the future complexity of various tasks, and based on the forecasted complexity, to determine the optimal voltage level, was highlighted in Chapter 3, where we proposed two online DVS algorithms that achieved near optimal performance.

Third, quality-complexity models enable us to develop Pareto efficient and fair resource allocation solutions for multiple delay-critical multimedia tasks running on the same system. Importantly, the principle of designing application-aware objective functions (e.g. social welfare) for resource allocation relies on the fact that the applications can accurately model their quality-complexity tradeoffs.

Fourth, we demonstrate that a novel, decentralized resource management scheme can be constructed to achieve intended centralized resource allocation solutions, even when the autonomous applications are unwilling to reveal their private internal parameters, algorithms, and quality-complexity functions. In chapter 4, the decentralized algorithms are implemented by exchanging “tax functions” and resource demands between the system resource manager and the applications. The tax functions are messages designed specifically by the resource manager to compel tasks to conform to a variety of intended resource allocation solutions, including (but not limited to) maximizing social welfare, minimizing system energy consumption, and performing workload balancing on multiple processors. An important feature of tax functions is that they do not require the resource manager to know each application's utility-resource function, but can nevertheless be adapted to produce the intended results.

Fifth, stochastic complexity models can also provide improved methods for estimating and optimizing the performance of informationally-distributed systems. For example, we used a stochastic model to construct delay-sensitive utility functions for stream mining across distributed, autonomous sites (Chapter 6). The modeled utility function could be decomposed into a product of local utility functions, which could then be locally estimated, calculated, and exchanged in order to obtain an estimate of the global utility function. Without this decomposition, a large amount of coordination and information sharing would be required between the various sites, which might not be feasible given the proprietary restrictions and communications/processing overhead.

One final insight is that by extending the well-known Markov Decision Process (MDP), we can develop a rules-based framework that performs well in both static and dynamic environments. In particular, the rules-based approach differs from conventional MDP approaches in that the best performance that can be achieved by MDP in a static environment corresponds to a quantized action, which depending on the quantization method, can be significantly worse than the optimal convergent solution of an algorithm. The rules based approach, on the other hand, can choose the algorithm that optimizes performance in static environments (e.g. safe experimentation and local search in Chapter 6), while also choosing different algorithms to reconfigure classifiers in dynamic environments and guaranteeing high average performance like MDP.

Importantly, future multimedia systems will need to support an increasing number of concurrent multimedia tasks. Furthermore, new multimedia applications, such as multi-view video coding, require significantly more computational complexity than the previous generation of multimedia applications [80]. Likewise, as distributed multimedia applications such as real-time video streaming and online gaming become more popular, optimization techniques will be required to ensure that not only bandwidth, but also processing resources, are appropriately scheduled and distributed across sites in order to maximize user experience. The analytical methods proposed in this framework can have a broad impact in shaping the design of future multimedia systems by providing efficient real-time scheduling and resource allocation solutions for low-power and distributed systems.

The research described in this thesis can be extended in at least three directions. First, the proposed framework, mechanisms developed and statistics obtained from our system evaluations can impact both computer system developers and multimedia service providers. By having analytical models for application utility and complexity tradeoffs, these service providers can provide fair and efficient resource allocation solutions for multi-user environments.

Second, the proposed paradigm can catalyze a shift in multimedia software and systems research based on economics principles. For instance, powerful systems or devices can configure themselves as resource brokers that can make money or acquire other services in exchange for providing their system resources. In order for systems to determine the best prices for their services, they need to be able to accurately model the tradeoffs between resource costs and application utilities (i.e. willingness to pay), or at least, to be able to achieve their intended goals in a decentralized manner.

A third and more theoretical extension of this research involves determining how various parts of the framework should be jointly and automatically configured. For example, what is the minimum information exchange required from sites in order to optimize the performance of various distributed application? Implicit in such a question is knowing the underlying modeling approach, which can yield different modeled parameters that can be exchanged between applications or sites (e.g. utility metrics, processing delays, etc.). While the thesis discusses how to utilize each component for particular problems and solutions (e.g. stochastic modeling for DVS), a *complete* set of rules for determining how to jointly model, exchange information, and employ learning solutions for different types of multimedia applications and systems was not determined. Such rules can further guide the design of future dynamic, distributed, and highly complex multimedia systems.

Appendix A: Proof of Proposition 1

Proof: To show that the decentralized task objective functions are at equilibrium at the optimal point of the SWMEM objective function (95), and that the equilibrium point is unique, we use the Karush-Kuhn-Tucker (KKT) conditions for optimality. Since the SWMEM function is concave, the KKT conditions for optimality exist at a unique point, where for some $\vec{\mu} = (\mu_1, \mu_2, ..., \mu_I) \geq 0$, the following conditions hold:

$$\begin{aligned} &\nabla\left(\sum_{i=1}^{I} Q_i(x_i) - E_{\text{tot}}^{\text{act}}\left(\sum_{i=1}^{I} x_i\right)\right)\Bigg|_{\mathbf{x}=\mathbf{x}^*} - \vec{\mu} = 0 \\ &\mu_i x_i^* = 0,\ i = 1,...,I \end{aligned}. \quad (145)$$

Likewise, the KKT conditions for the task level optimization using the EEM tax function (96) are:

$$\begin{aligned} &\frac{\partial}{\partial x_i} Q_i(x_i) - \frac{\partial}{\partial x_i} E_{\text{tot}}^{\text{act}}\left(\sum_{i=1}^{I}(x_i + d_i)\right)\Bigg|_{x_i=x_i^*} - \mu_i = 0 \\ &\mu_i x_i^* = 0 \end{aligned}. \quad (146)$$

For all tasks to be at an equilibrium point for the decentralized algorithm, the perceived change in demand for each task i should be 0, and thus based on (97), the perceived demand from other tasks is $d_i = \sum_{l \neq i} x_l^*$. Hence at equilibrium, we have the following condition:

$$\begin{aligned} \frac{\partial}{\partial x_i} Q_i(x_i)\Big|_{x_i=x_i^*} - \frac{\partial}{\partial x_i} E_{\text{tot}}^{\text{act}}(x_i + d_i)\Big|_{x_i=x_i^*} - \mu_i &= \frac{\partial}{\partial x_i} Q_i(x_i)\Big|_{x_i=x_i^*} - \frac{\partial}{\partial x_i} E_{\text{tot}}^{\text{act}}\left(x_i + \sum_{l \neq i} x_l^*\right)\Bigg|_{x_i=x_i^*} - \mu_i \\ &= \nabla_i\left(\sum_{i=1}^{I} Q_i(x_i) - E_{\text{tot}}^{\text{act}}\left(\sum_{i=1}^{I} x_i\right)\right)\Bigg|_{\mathbf{x}=\mathbf{x}^*} - \mu_i \end{aligned}. \quad (147)$$

Since the intersection of all KKT conditions for the local objective functions at equilibrium (147) is the same optimality condition for the global system objective (145), convergence of the decentralized algorithm (EEM) guarantees an optimal solution to the global objective function (SWMEM). ■

Appendix B: Proof of convergence of SPERO

Proposition 11: For an algorithmic Markov system, SPERO converges to the optimal rule in steady state.

Proof: Note that the average utility in each state $Q(S_m)$, and the unknown state transition probabilities, must be perfectly estimated for each rule and state to determine the optimal rule. Since performing each state transition infinitely many times when $t \to \infty$ implies that each state is visited (and hence the utility is measured) infinitely many times, we need only prove perfect estimation for state transitions.

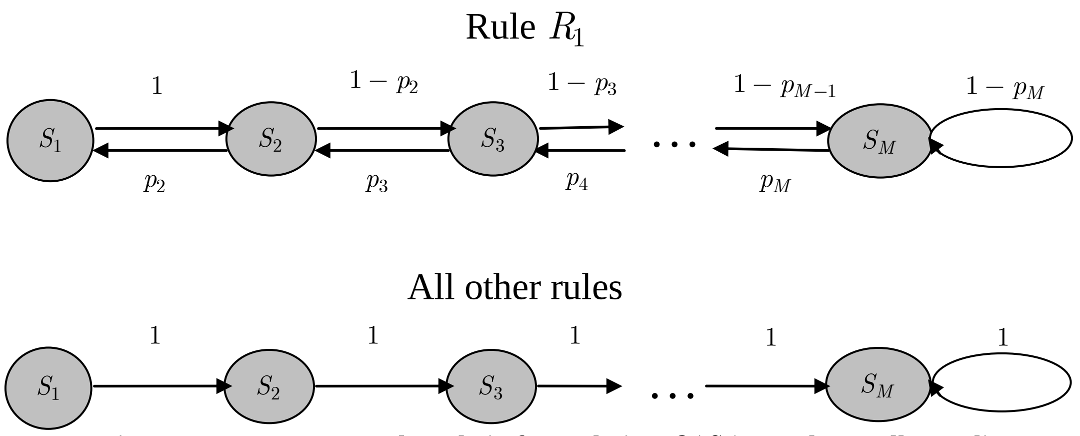


Figure 47 Worst case Markov chain for updating $Q(S_1)$, random walk on a line.

We use the worst-case lower bound for the number of times each rule is played in each state to prove that each feasible state transition occurs infinitely many times as $t \to \infty$. Consider the Markov chain given in Figure 47, where for a pure rule R_1, there exists a random walk from state S_m to S_{m+1} and S_{m-1} with non-zero probabilities (except for S_1 and S_M). For all other pure rules, algorithms are chosen such that state transitions always lead from S_m to S_{m+1}, until the state reaches S_M. Furthermore, we assume the relation $Q(S_1) < Q(S_2) < ... < Q(S_M)$, such that the solution reinforces rules leading away from S_1. This is the worst case scenario for updating the transition probabilities of S_1.

Let H be the total number of pure rules. Suppose that at time t, all other rules have been reinforced a total of t times, but R_1 has not been reinforced, i.e. $c_1 = 1$.

The worst case probability of playing R_1 at time t is if all other rules have been reinforced equally, i.e. $c_2 = c_3 = ... = c_H = t/(H-1)$. In this case, rule R_1 is played with probability $\mathbf{r}_1 = 1/\left(1+(H-1)\sqrt[M]{t/(H-1)}\right)$. The probability of transitioning from S_m to S_1 in $M-1$ steps, and playing rule R_1 in state S_1 (hence playing R_1 M consecutive times), can be bounded below by:

$$\begin{aligned} p(S_M \to S_1, R_1 \mid t) &= \mathbf{r}_1(t)\cdot p_M \cdot \mathbf{r}_1(t+1)\cdot p_{M-1}\cdot ... \cdot \mathbf{r}_1(t+M-2)\,p_2\cdot \mathbf{r}_1 \\ &> \prod_{m=2}^{M} p_m \prod_{m=0}^{M-1} \cdot \frac{1}{1+(H-1)\sqrt[M]{(t+m)/(H-1)}} \\ &> C\cdot\left(\frac{1}{1+(H-1)\sqrt[M]{(t+M-1)/(H-1)}}\right)^M \\ &> C\cdot\left(\frac{1}{2(H-1)\sqrt[M]{(t+M-1)/(H-1)}}\right)^M \\ &= C\cdot\left(\frac{1}{2(H-1)}\right)^M \frac{H-1}{t+M-1} \\ &> C'\frac{1}{t+M} \end{aligned} \quad . \qquad (148)$$

where C and C' are constants, and the inequality in the fourth line is due to the fact that $(H-1)\sqrt[M]{(t+m)/(H-1)} > 1$. Since from any other starting state transitioning to S_1 in less than M steps has higher probability than (148), we can also bound the average number of plays of rule R_1 in S_1 for every M time intervals by (148). Likewise, the average number of visits to any other state (e.g. S_2) starting from any initial state, times the probability of using any rule in the final state, is also bounded below by (148).

Thus, the total number of updates for transition probability pairs for any state S_m and rule R_h as $t \to \infty$ can be bounded below by:

$$\begin{aligned} \lim_{t\to\infty} E\left[N_{S_m}(t)\right] &> \sum_{\hat{t}=0}^{\infty} \frac{C'}{\hat{t}M+M} \\ &= C'\left(\frac{1}{M}+\frac{1}{2M}+\frac{1}{3M}+...\right) = \infty \end{aligned} , \qquad (149)$$

where the right hand side is given by recalculating the final inequality of (148) every M time intervals, starting with $t=0$. ■

Bibliography

[1] J. Markhoff, “Intel Prototype may herald a new age of processing”, New York Times, February 12, 2007, p. C9.

[2] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” *IEEE Trans. on Circ. Syst. for Video Technol.*, vol. 13, no. 7, July 2003.

[3] J.-R. Ohm, "Advances in scalable video coding," *Proc. of the IEEE*, vol. 93, pp. 42-56, Jan. 2005.

[4] D. G. Sachs, S. V. Adve, and D. L. Jones, “Cross-layer adaptive video coding to reduce energy on general purpose processors,” *Proc. IEEE Int. Conf. on Image Proc.*, ICIP 2003, vol. , pp. 25-28, Sept. 2003.

[5] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, “H.264/AVC baseline profile decoder complexity analysis,” *IEEE Trans. on Circ. and Syst. for Video Tech.*, vol. 13, no. 7, pp. 704-716, July 2003.

[6] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, “Video coding with H.264/AVC: Tools, performance, and complexity,” *IEEE Circ. and Syst. Mag.*, vol. 4, no. 1, pp. 7-28, Jan. 2004.

[7] J. Valentim, et al, “Evaluating MPEG-4 video decoding complexity for an alternative video verifier complexity model”, *IEEE Trans. on Circ. and Syst. for Video Tech.*, vol. 12, no. 11, pp. 1034-1044, Nov. 2002.

[8] D. S. Turaga, M. van der Schaar, B. Pesquet-Popescu, “Complexity-scalable motion compensated wavelet video encoding,” *IEEE Trans. on Circ. and Syst. for Video Tech.*, vol. 15, no. 8, pp. 982-993, Aug. 2005.

[9] W. Yuan and K. Nahrstedt, “Energy-efficient CPU scheduling for multimedia applications,” *ACM Trans. on Computer Syst.*, Vol. 24, Issue 3, Aug. 2006.

[10] M. Wang, M. van der Schaar, “Operational rate-distortion modeling for wavelet video coders,” *IEEE Trans. on Signal Proc.*, Vol. 54, Issue 9, Sept. 2006.

[11] M. van der Schaar and Y. Andreopoulos, “Rate-distortion-complexity modeling for network and receiver aware adaptation,” *IEEE Trans. on Multimedia*, vol. 7, no. 3, pp. 471-479, June 2005.

[12] Y. Andreopoulos and M. van der Schaar, “Complexity-constrained video bitstream shaping,” *IEEE Trans. on Signal Processing*, to appear, preprint available at: www.ee.ucla.edu/~yandreop .

[13] Z. He, and S. K. Mitra, “A unified rate-distortion analysis framework for

transform coding," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 11, no. 12, Dec. 2001.

[14] H.-M. Hang and J.-J. Chen, "Source model for transform video coder and its application – Part I: Fundamental theory," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 7,no. 2,pp. 287-298,Apr. 1997.

[15] Z. He, Y. Liang, L. Chen, I. Ahmad and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circuits and Syst. for Video Technol.*, vol. 15, no. 5, pp. 645-658, May 2005.

[16] M. Flierl and B. Girod, "Video coding with motion-compensated lifted wavelet transforms," Signal Processing: Image Communication, vol. 19, no. 7, pp. 561-575.

[17] B. Girod, "Efficiency analysis of multihypothesis motion-compensated prediction for video coding," *IEEE Trans. on Image Proc.*, vol. 9, no. 2, pp. 173-183, Feb. 2000.

[18] K. Stuhlmuller, N. Farber, M. Link and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Select. Areas in Comm.*, vol. 18, no. 6, pp. 1012-1032, June 2000.

[19] W. Ding and B. Liu, "Rate control of MPEG video coding and recording by rate-quantization modeling," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 6, pp. 12-20, Feb. 1996.

[20] D. Taubman, M. W. Marcellin, JPEG 2000-Image Compression Fundamentals, Standards and Practice, *Kluwer Academic Publishers*, 2002.

[21] P. Schelkens, A. Munteanu, J. Barbarien, M. Galca, X. Giro-Nieto, J. Cornelis, "Wavelet coding of volumetric medical datasets," *IEEE Transactions on Medical Imaging*,vol. 22,no. 3,pp. 441-458,Mar. 2003.

[22] R. Shukla, P. L. Dragotti, M. N. Do, and M. Vetterli, "Rate-distortion optimized tree-structured compression algorithms for piecewise polynomial images," *IEEE Trans. on Image Proc.*, vol. 14, no. 3, Mar. 2005.

[23] P. Chen and J. W. Woods, "Bidirectional MC-EZBC with lifting implementation," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 14, no. 10, pp. 1183-1194, Oct. 2004.

[24] A. Munteanu, J. Cornelis, G. Van der Auwera and P. Cristea, "Wavelet image compression – the quadtree coding approach," *IEEE Trans. on Information Technol. in Biomed.*, vol. 3, no. 3, pp. 176-185, Sept. 1999.

[25] C. Chrysafis, et al, "SBHP – a low complexity wavelet coder," Proc. IEEE Internat. Conf. on Acoust. Speech and Signal Process., ICASSP '00, vol. 6, pp. 2035-2038, June 2000.

[26] J. Xu, Z. Xiong, S. Li and Y. Zhang, "Three-dimensional embedded subband coding with optimized truncation (3-D ESCOT)," *Appl. Comput. Harmon. Anal.*, vol. 10, pp. 290-315, 2001.

[27] J. Liu, P. Moulin, “Information-theoretic analysis of interscale and intrascale dependencies between image wavelet coefficients,” *IEEE Trans. on Image Processing*, vol. 10, pp. 164701658, Nov., 2001.

[28] M. K. Mihçak, et al, “Low-complexity Image Denoising based on Statistical Modeling of Wavelet Coefficients,” *IEEE Signal Processing Letters*, vol. 6, no. 12, pp. 300-303, 1999.

[29] S. Mallat, F. Falzon, “Analysis of low bit-rate image transform coding,” *IEEE Trans. on Signal Processing*, vol. 46, pp. 1027-1042, Apr., 1998.

[30] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. van der Schaar, J. Cornelis and P. Schelkens, “In-band motion compensated temporal filtering,” *Signal Processing: Image Communication*, vol. 19, no. 7, pp. 653-673, Aug. 2004.

[31] T. Rusert, K. Hanke, and J. Ohm, “Transition filtering and optimization quantization in interframe wavelet video coding,” *VCIP, Proc. SPIE*, vol. 5150, pp. 682-693, 2003.

[32] P. Borwein, T. Erdelyi, Polynomials and polynomial inequalities, *Springer*, New York, 1995.

[33] C. Hummel, “Chapter IV: The Squeezing Theorem,” in: Gromov's Compactness Theorem for Pseudo-Holomorphic Curves, *Spinger*, Series: Progress in Mathematics, vol. 151, 1997.

[34] D. Marpe, et al, “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard,” *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 13, Issue 7, July 2003.

[35] C. Chrysafis, A. Ortega, “Efficient context-based entropy coding for lossy wavelet image compression,” in *Proc. IEEE Data Compression Conference*, Snowbird, UT, 1997, pp. 221-230.

[36] Y. Andreopoulos and M. van der Schaar, "Adaptive linear prediction for resource estimation of video decoding," *IEEE Trans. on Circuits and Systems for Video Technology*, to appear.

[37] E. Lam, J. Goodman, “A Mathematical Analysis of the DCT Coefficient Distributions for Images,” *IEEE Transactions on Image Processing*, Vol. 9, Issue 10, Oct. 2000.

[38] Intel Inc. “Intel XScale Technology,” Available: http://www.intel.com/design/intelxscale

[39] AMD Inc. ”AMD PowerNow!TM Technology Platform Design Guide for Embedded Processors,” Available: http://www.amd.com/epd/processors

[40] L. Benini, G. De Micheli. *Dynamic Power Management: Design Techniques and CAD Tools.* Kluwer Academic Publishers, Norwell, MA, 1997.

[41] P. Pillai, K. Shin. “Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems.” *Proceedings of the eighteenth ACM symposium on Operating Systems*, 2001.

[42] V. Raghunathan, C. Pereira, M. Srivastava, and R. Gupta. "Energy aware wireless systems with adaptive power-fidelity tradeoffs," IEEE Transactions on VLSI Systems , February 2005.

[43] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets. "GRACE: Cross-layer Adaptation for Multimedia Quality and Battery Energy," *IEEE Transactions on Mobile Computing*, 2006.

[44] Y. Zhu, F. Mueller. "Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling," Proceedings of the 11th international conference on Computer Architecture, 2004.

[45] D. H. Albonesi, R. Balasubramonian, S. Dropsho, S. Dwarkadas, E. G. Friedman, M. Huang, V. Kursun, G. Magklis, M. L. Scott, G. Semeraro, P. Bose, A. Buyuktosunoglu, P. W. Cook, S. E. Schuster. "Adaptive Processing: Dynamically Tuning Processor Resources for Energy Efficiency", *IEEE Computer*, December 2003

[46] S. Irani, G. Singh, S. K. Shukla, and R. K. Gupta. "An overview of the competitive and adversarial approaches to designing dynamic power management strategies," *IEEE Trans. on Very Large Scale Integ.*, vol. 13, no. 12, pp.1349-1362, Dec. 2005.

[47] Kihwan Choi, Karthik Dantu, Wei-Chung Cheng, Massoud Pedram. "Frame-based dynamic voltage and frequency scaling for a MPEG decoder." *ICCAD* 2002, pp. 732-737

[48] H. Aydin, R. Melhem, D. Mosse, P. Mejia-Alvarez. "Power-aware Scheduling for Periodic Real-time Tasks," *IEEE Trans. on Computers*, Vol. 53, Issue 5, May 2004.

[49] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *Proc. Int'l Symposium on Low Power Electronics and Design*, pp. 76-81, Aug. 1998.

[50] W. Yuan, K. Nahrstedt. "Practical voltage scaling for mobile multimedia devices," *Proceedings of the 12th annual ACM international conference on Multimedia*, October 10-16, 2004, New York, NY, USA.

[51] A. Maxiaguine, S. Chakraborty, L. Thiele. "DVS for buffer-constrained architectures with predictable QoS-energy tradeoffs," *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2005.

[52] Z. Ren, B. Krogh, R. Marculescu. "Hierarchical adaptive dynamic power management," *IEEE Trans. on Computers*, Vol. 54, Issue. 4, Apr. 2005.

[53] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, G. de Micheli. "Dynamic voltage scaling and power management for portable systems," *Proc. from Design Automation Conference*, 2001.

[54] J. Liu, W. Shih, K. Lin, R. Bettati, J. Chung. "Imprecise Computations," *Proceedings of the IEEE*, 1994.

[55] M. Mesarina, Y. Turner. "Reduced Energy Decoding of MPEG Streams," *Multimedia Systems*, 2003.

[56] J. Shapiro. "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. on Signal Processing*, 1993.

[57] O. Boxma, W. Groenendijk. "Waiting Times in Discrete-Time Cyclic-Service Systems," *IEEE Transactions on Communications*, Vol. 36, No. 2, February 1988.

[58] Teunis J. Ott. "Simple Inequalities for the D/G/1 Queue." *Operations Research*, 1987 INFORMS.

[59] L. D. Servi. "D/G/1 Queues with Vacations." *Operations Research*, 1986 INFORMS.

[60] H. Zhang, S. C. Graves. "Cyclic Scheduling in a Stochastic Environment." *Operations Research*, 1997 INFORMS.

[61] D. M. Sow, A Eleftheriadis. "Complexity Distortion Theory." *IEEE Trans. Inform. Theory*, vol. 49, no. 3, pp. 604-608, Mar. 2003.

[62] B. Foo, Y. Andreopoulos, M. van der Schaar. "Analytical Rate-Distortion-Complexity Modeling of Wavelet-based Video Coders." *IEEE Trans. on Signal Processing*, Feb. 2008.

[63] Y. Lim, J. Kobza. "Analysis of a delay-dependent priority discipline in a multi-class traffic packet switching node," *Proc. IEEE INFOCOM*, pp. 889-898, Apr. 1988.

[64] T. Ishihara, H. Yasuura. "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," in *Proc. ACM ISLPED*, 1998, pp. 197-202.

[65] S. Regunathan, P. A. Chou, and J. Ribas-Corbera, "A generalized video complexity verifier for flexible decoding," *Proc. IEEE International Conference on Image Processing*, vol. 3, pp. 289-292, Sept. 2003.

[66] A. Ortega, K. Ramchandran. "Rate-distortion Methods for Image and Video Compression," *IEEE Signal Processing Mag.*, vol. 15, issue 6, pp. 23-50, Nov, 1998

[67] J. Ohm, M. van der Schaar, J. Woods. "Interframe Wavelet Coding—Motion Picture Representation for Universal Scalability," *Signal Processing: Image Communication* 19 (2004) pp. 877-908.

[68] F. Fitzek, M. Reisslein. "MPEG-4 and H.263 Video Traces for Network Performance Evaluation," *IEEE Network*, Nov.-Dec. 2001.

[69] T. Jiang, C. K. Tham, and C. C. Ko, "An approximation for waiting time tail probabilities in multiclass systems", *IEEE Communications Letters*, vol. 5, no. 4, pp 175-177. April 2001.

[70] J. Abate, G. L. Choudhury, and W. Whitt. "Exponential approximations for tail probabilities in queues I: Waiting times", *Operations Research*, vol. 43, no. 5, pp

885-901, 1995.

[71] R.G. Gallager, *Discrete Stochastic Processes*, Kluwer, Dordrecht, 1996.

[72] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[73] D. Gross and C. Harris, *Fundamentals of Queueing Theory*, New York: Wiley-Interscience, 1997.

[74] Y. Yoo, A. Ortega, and B. Yu, "Image subband coding using progressive classification and adaptive quantization," *IEEE Trans. Image Processing*, pp. 1702-1715, Dec. 1999.

[75] M. Ravasi, M. Mattavelli, P. Schumacher, R. Turney, "High-Level Algorithmic Complexity Analysis for the Implementation of a Motion-JPEG2000 Encoder," *PATMOS* 2003: 440-450.

[76] M. Mattavelli and S. Brunetton. "Implementing real-time video decoding on multimedia processors by complexity prediction techniques," *IEEE Transactions on Consumer Electronics*, 44(3):760--767, August 1998.

[77] Q. Wu, P. Juang, M. Martonosi, D. Clark. "Formal online methods for voltage/frequency control in multiple clock domain microprocessors," *Proc. of 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS),* 2004.

[78] S. Zhang, and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. *Proceedings of ICCAD*, 2007.

[79] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," *Proceedings of Design and Automation Conference (DAC)*, 2004.

[80] Y. Zhang, G. Jiang, W. Yi, M. Yu, Z. Jiang, Y. Kim, "An Approach to Multi-Modal Multi-View Video Coding," *Proceedings of the 8th International Conference on Signal Processing*, 2006.

[81] Z. Cao, B. Foo, L. He, M. van der Schaar, "Optimality and Improvement of Dynamic Voltage Scaling Algorithms for Multimedia Applications," *Proceedings of DAC,* 2008. (Nominated for best paper award)

[82] B. Foo, M. van der Schaar. "A Queuing Theoretic Approach to Processor Power Adaptation for Video Decoding Systems." IEEE Trans. on Signal Processing, Vol. 56, No. 1, Jan. 2008.

[83] A. Adas. Traffic Models in Broadband Networks. IEEE Communications Magazine, Vol. 35, Issue 7, July 1997.

[84] Intel Inc. "Intel Multi-Core Processors: Leading the Next Digital Revolution" Available: http://www.intel.com/technology/magazine/computing/multi-core-0905.pdf

[85] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. "Project Aura:

Towards Distraction-Free Pervasive Computing," *IEEE Pervasive Computing*, Volume 21, Number 2, April-June, 2002.

[86] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lematre, N. Maudet, J. Padget, S. Phelps, J. A. Rodrguez-Aguilar, and P. Sousa. "Issues in multiagent resource allocation." http://www.doc.ic.ac.uk/˜ue/MARA/mara-may-2005.pdf.

[87] V. Poladian, J. Sousa, D. Garlan, M. Shaw. "Dynamic configuration of resource-aware services," *Proc. of the 26th ICSE*, 2004.

[88] V. Poladian, J. Sousa, F. Padberg, M. shaw. "Anticipatory configuration of resource-aware applications," *ACM SIGSOFT Software Engineering Notes*, Vol. 30, Issue 4, July 2005.

[89] R. Rajkumar, C. Lee, J. Lehoczky, D. Siewiorek, "A Resource Allocation Model for QoS Management," *IEEE Real-Time Systems Symposium*, 1997.

[90] Ghosh, S., Rajkumar, R. R., Hansen, J., and Lehoczky, J. (2003). "Scalable resource allocation for multi-processor QoS optimization," In *23rd IEEE International Conference on Distributed Computing Systems* (ICDCS 2003).

[91] L. Capra, W. Emmerich, C. Mascolo, "CARISMA: Context-aware reflective middleware system for mobile applications," *IEEE Trans. on Software Engineering*, Vol. 29, No. 10, Oct. 2003.

[92] List of Video Codecs at: http://www.fourcc.org/codecs.php

[93] T. Stoenescu, J. Ledyard. "A Pricing Mechanism which Implements a Network Rate Allocation Problem in Nash Equilibria," submitted to *IEEE/ACM Transactions on Networking*.

[94] X. Lin, N. Shroff, R. Srikant, "A Tutorial on Cross-Layer Optimization in Wireless Networks." *IEEE JSAC*, 2006.

[95] T. Wiegand, H. Schwarz, H. Joch, A. Kossentini, F. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. on Circuits and Systems for Video Technology (CSVT)*, Vol. 13, Issue 7, July 2003.

[96] P. Chowdhury, C. Chakrabarti. "Static Task-Scheduling Algorithms for Battery-Powered DVS Systems," *IEEE Trans. on VLSI Systems*, Vol. 13, No. 2, Feb. 2005.

[97] R.G. Gallagher, *Discrete Stochastic Processes*, Kluwer, Dordrecht, 1996.

[98] L. Rabiner, B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, Vol. 3, Issue 1, Jan. 1986.

[99] B. Foo, Y. Andreopoulos, M. van der Schaar. "Analytical Complexity modeling of Wavelet-based Video Coders," *ICASSP* 2007.

[100] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[101] N. Mastronarde and M. van der Schaar, "A Bargaining Theoretic Approach to Quality-Fair System Resource Allocation for Multiple Decoding Tasks," *IEEE*

Trans. Circuits and Systems for Video Technology, to appear.

[102] Z. Yu, J. Zhang, "Video deblocking with fine-grained scalable complexity for embedded mobile computing," *ICSP* 2004.

[103] N. A. Lynch, *Distributed Algorithms.* San Mateo, CA: Morgan Kaufmann, pp. 51-80, 1996.

[104] D. Bertsekas, *Nonlinear Programming,* Palgrave Macmillan, 1997.

[105] A. Sinha and A. P. Chandrakasan, "Jouletrack: A web based tool for software energy profiling," in *Proc. IEEE/ACM DAC*, pp. 220–225, 2001.

[106] M. Shah, J. Hellerstein, M. Franklin, "Flux: An adaptive partitioning operator for continuous query systems," *International Conference on Data Engineering (ICDE),* 2003.

[107] C. Olston, J. Jiang, J. Widom, "Adaptive filters for continuous queries over distributed data streams," *ACM SIGMOD, 2003 International Conference on Management of Data,* pp. 563-574, June 2003.

[108] L. Amini, H. Andrade, F. Eskesen, R. King, Y. Park, P. Selo, C. Venkatramani, "The stream processing core," *Technical Report RSC 23798*, Nov. 2005.

[109] D. Turaga, O. Verscheure, U. Chaudhari, L. Amini, "Resource Management for Chained Binary Classifiers," *Proceedings of the Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, 2006.

[110] Y. Schapire, "A brief introduction to boosting," *International Conference on Algorithmic Learning Theory*, 1999.

[111] A. Garg, V. Pavlovic, "Bayesian networks as ensemble of classifiers," *International Conference on Pattern Recognition (ICPR),* 2002.

[112] B. Babcock, S. Babu, M. Datar, R. Motwani, "Chain: Operator scheduling for memory minimization in data stream systems," *In ACM International Conference on Management of Data (SIGMOD),* 2003.

[113] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, M. Stonebraker, "Load shedding in a data stream manager," *Proc. of the 29th International Conference on Very Large Databases (VLDB),* Sept. 2003.

[114] B. Babcock, M. Datar, R. Motwani, "Cost-efficient mining techniques for data streams," *in Worshop on Management and Processing of Data Streams (MDPS),* 2003.

[115] N. Tatbul, S. Zdonik, "Dealing with overload in distributed stream processing systems," *in IEEE International Workshop on Networking Meets Databases (NetDB),* 2006.

[116] N. Tatbul, "QoS-driven load shedding on data streams," *in XML-Based Data Management and Multimedia Engineering,* 2002.

[117] V. Eide, F. Eliassen, O. Granmo, O. Lysne, "Supporting timeliness and accuracy in distributed real-time content-based video analysis," in ACM

International Conference on Multimedia, 2003.

[118] Y. Chi, P. Yu, H. Wang, R. Muntz, "Loadstar: A load shedding scheme for classifying data streams," *International Conference on Data Mining*, Oct. 2005.

[119] D. Turaga, O. Verscheure, U. Chaudhari, L. Amini, "Resource management for networked classifiers in distributed stream mining systems," *IEEE ICDM*, Dec. 2006.

[120] F. Fu, D. Turaga, O. Verscheure, M. van der Schaar, and L. Amini, "Configuring Competing Classifier Chains in Distributed Stream Mining Systems" *IEEE Journal of Selected Topics in Signal Process. (JSTSP)*, 2008.

[121] Y. Xing, S. Zdonik, J.-H. Hwang, "Dynamic Load Distribution in the Borealis Stream Processor." *In proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, Tokyo, Japan, April 2005.

[122] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, S. Zdonik. "Scalable Distributed Stream Processing," Proc. of CIDR, Asilomar, CA, Jan. 2003.

[123] M. Balazinska, H. Balakrishnan, S. Madden, M. Stonebraker, "Fault Tolerance in the Borealis Distributed Stream Processing System," SIGMOD International Conference on Management of Data, SIGMOD 2005.

[124] M. Kudo, J. Toyama and M. Shimbo, "Multidimensional Curve Classification Using Passing-Through Regions," *Pattern Recognition Letters*, Vol. 20, No. 11-13, pp. 1103-1111, 1999. (See: http://kdd.ics.uci.edu/databases/JapaneseVowels/JapaneseVowels.html for the data set.)

[125] A. Hero, J. Kim, "Simultaneous signal detection and classification under a false alarm constraint," *ICASSP*, 1990.

[126] D. Turaga and T. Chen, "I/P frame selection using classification based mode decisions," *ICIP* 2001.

[127] A. Pentland, B. Moghaddam, T. Starner, "View-Based and Modular Eigenspaces for Face Recognition," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1994.

[128] P. Burke, "The Output of a Queuing System," *Operations Research* 4, 699, 1956.

[129] A. Roth, I. Erev, "Learning in extensive-form games: experimental data and simple dynamic models in the intermediate term," *Games and Economic Behavior*, 8, pp. 164-212, 1995.

[130] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, 2nd edition, 1991.

[131] S. Viglas, J. Naughton, "Rate-based query optimization for streaming information sources," *SIGMOD*, 2002.

[132] M. Ciraco, M. Rogalewski, G. Weiss, "Improving classifier utility by altering

the misclassification cost ratio," 1^{st} *International Workshop on Utility-based Data Mining,* 2005.

[133] D. Abadi, D. carney, U. Centintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik, "Aurora: a new model and architecture for data stream management," *VLDB Journal,* 2003.

[134] V. Kumar, B. Cooper, K. Schwan, "Distributed Stream Management using Utility-Driven Self-Adaptive Middleware," *ICAC,* 2005.

[135] E. Horvitz, G. Rutledge, "Time-Dependent Utility and Action Under Uncertainty," *In Proc. of the Seventh Conference on Uncertainty in Artificial Intelligence,* pp. 151-158, July, 1991.

[136] P. Young, *Strategic Learning and Its Limits,* Oxford University Press, 2004.

[137] F. Douglis, M. Branson, K. Hildrum, B. Rong, F. Ye, "Multi-site Cooperative Data Stream Analysis," *ACM SIGOPS,* Vol. 40, Issue 3, July 2006.

[138] J. Marden, H. Young, G. Arslan, J. Shamma, "Payoff Based Dynamics for Multi-Player Weakly Acyclic Games," submitted to *SIAM Journal on Control and Optimization, special issue on Control and Optimization in Cooperative Networks*, 2007.

[139] D. Fudenberg, J. Tirole, *Game Theory,* The MIT Press, 1991.

[140] P. Pardalos, H. Romeijn, H. Tuy, "Recent developments and trends in global optimization," *Journal of Computational and Applied Mathematics,* Vol. 124, Issues 1-2, Dec. 2000.

[141] R. Horst and P.M. Pardalos, *Handbook of Global Optimization,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[142] K. Arrow, *Social Choice and Individual Values,* Yale University Press, 1970.

[143] J. Nash, "The Bargaining Problem," *Econometrica,* 1950.

[144] P. Boggs AND J. Tolle, "Sequential quadratic programming," *Acta Numerica*, 1995.

[145] R. Lienhart, L. Liang, A. Kuranov, "A detector tree for boosted classifiers for real-time object detection and tracking," *in Proceedings of the International Conference on Multimedia and Expo (ICME)*, 2003.

[146] H. Xiao, L. Zhu, "Boosting chain learning for object detection," *in Proceedings of the 9th IEEE International Conference on Computer Vision,* 2003.

[147] T. Senator, "Multi-stage classification," *in Proceedings of the 5th International Conference on Data Mining (ICDM),* 2005, pp. 386–393.

[148] J. Smith, "IBM multimedia analysis and retrieval system (MARVEL)," http://mp7.watson.ibm.com/marvel/.

[149] L. Carreras, "Boosting trees for anti-spam email filtering," *in Proceedings of the Recent Advances in Natural Language Processing,* 2001.

[150] M. Chen, T. Yen, B. Coonan, “Real-time fault detection and classification for manufacturing etch tools,” *in Proceedings of IEEE Semiconductor Manufacturing Technology Workshop,* Sept, 2004.

[151] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, M. Stonebraker, “Load shedding in a data stream manager,” *Proc. of the 29th International Conference on Very Large Databases (VLDB),* Sept. 2003.

[152] B. Babcock, M. Datar, R. Motwani, “Cost-efficient mining techniques for data streams,” *Workshop on Management and Processing of Data Streams (MDPS),* 2003.

[153] B. Foo, M. van der Schaar, "Distributed Classifier Chain Optimization for Real-time Multimedia Stream Mining Systems," *Proceedings of SPIE Multimedia Content Access, Algorithms and Systems II,* Jan. 2008.

[154] E. Crawford, M. Veloso, “Learning to Select Negotiation Strategies in Multi-Agent Meeting Scheduling,*” In the working notes of the Multiagent Learning Workshop (to appear), AAAI,* Pittsburgh, USA, 2005.

[155] S. Merugu, J. Ghosh, “Privacy-preserving distributed clustering using generative models,” *International Conference on Data Mining (ICDM),* 2003.

[156] L. Xie, S. Chang, A. Divakaran, and H. Sun, “Structure analysis of soccer video with hidden Markov models,” *In Proc. Interational Conference on Acoustic, Speech and Signal Processing (ICASSP),* Orlando, FL, 2002.

[157] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming,* John Wiley & Sons, 1994.

[158] R. Sutton, A. Barto, *Reinforcement Learning: An Introduction,* MIT Press, 1998.

[159] C. O’Cinneide, “Entrywise perturbation theory and error analysis for Markov chains,” *Numerische Mathematik,* Vol. 65, No. 1, Dec. 1993.