Optimizing HTTP-Based Adaptive Streaming in Vehicular Environment using Markov Decision Process

Ayub Bokani[1] Mahbub Hassan[2] Salil S. Kanhere[3]
Xiaoqing Zhu [4]

[1,2,3] University of New South Wales, Sydney, Australia
{abokani, mahbub, salilk}@cse.unsw.edu.au

[4] Chief Technology and Architecture Office,
Cisco Systems, San Jose, CA USA
xiaoqzhu@cisco.com

Technical Report
UNSW-CSE-TR-201512
July 2015

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Hypertext transfer protocol (HTTP) is the fundamental mechanics supporting web browsing on the Internet. An HTTP server stores large volumes of contents and delivers specific pieces to the clients when requested. There is a recent move to use HTTP for video streaming as well, which promises seamless integration of video delivery to existing HTTP-based server platforms. This is achieved by segmenting the video into many small chunks and storing these chunks as separate files on the server. For adaptive streaming, the server stores different quality versions of the same chunk in different files to allow real-time quality adaptation of the video due to network bandwidth variation experienced by a client. For each chunk of the video, which quality version to download, therefore, becomes a major decision-making challenge for the streaming client, especially in vehicular environment with significant uncertainty in mobile bandwidth. In this paper, we demonstrate that for such decision making, Markov decision process (MDP) is superior to previously proposed non-MDP solutions. Using publicly available video and bandwidth datasets, we show that MDP achieves up to 15x reduction in playback deadline miss compared to a well-known non-MDP solution when the MDP has the prior knowledge of the bandwidth model. We also consider a model-free MDP implementation that uses Q-learning to gradually learn the optimal decisions by continuously observing the outcome of its decision making. We find that MDP with Q-learning significantly outperforms MDP that uses bandwidth models.

1 Introduction

Due to immense scalability benefits, there is a strong push from the industry to adopt HTTP as a universal platform for delivering all types of contents, including video. Apple [1], Microsoft [2], and Adobe [3] have deployed their own proprietary HTTP-based video streaming platforms while a standard, called dynamic adaptive streaming over HTTP (DASH) [4], has been introduced by the world wide web consortium (W3C) to facilitate wide-spread deployment of this technology.

The key concept in DASH is to code the same video in multiple bitrates (qualities) and store each stream as a series of small video chunks of 2-10 sec duration. A client simply downloads and plays a chunk of a given quality using the standard HTTP GET command used for fetching any other objects on the Web. Since video has strict display deadlines for every frame, each chunk needs to be downloaded before its deadline to avoid the 'freezing' effect. It therefore becomes the responsibility of a DASH client to dynamically select the 'right' quality of the next chunk to ensure a smooth video at the receiver with the highest possible quality and minimum number of quality switches from one chunk to the next. The DASH standard specifies the format of metadata about the chunks, such as their URL and bitrates, which are sent to the clients in a manifest file. The actual *streaming strategy*, i.e., the client intelligence for selecting the right quality for each chunk in order to produce a high quality of experience (QoE) for the viewer is left to the developers.

Majority of the previous works [5,6] proposed simple heuristics that make decisions about the next chunk quality based only on the currently buffered video in the client and the recent observation of network bandwidth. These heuristics perform reasonably well, but do not *optimize* the tradeoff between individual QoE metrics such as picture quality vs deadline miss, especially in vehicular environments exhibiting significant uncertainty in network bandwidth. The key objective of this paper is to explore more advanced decision making tools that would enable an improved tradeoff between conflicting QoE metrics in vehicular environments. In particular, we study the effectiveness of Markov decision process (MDP), which is known for its ability to optimize decision making under uncertainty [7]. In this paper, we make two fundamental contributions:

- Using publicly available DASH datasets and real-world traces of mobile bandwidth and vehicular mobility, we show that MDP can reduce playback deadline miss up to 15 times compared to a well known non-MDP strategy when the bandwidth model is known *a priori*.
- We propose a Q-learning implementation of MDP that does not need any *a priori* knowledge of the bandwidth, but learns optimal decision making in a self-learning manner by simply observing the outcome of its decision making. We demonstrate that, in terms of deadline miss, the Q-learning-based MDP outperforms the model-based MDP by a factor of 3.

The rest of the paper is organised as follows. We discuss related work in Section 2. Section 3 shows how DASH can be formulated as an MDP problem. We discuss the three proposed MDP implementation algorithms in Section 4 followed by the non-MDP approach in Section 5. The simulation setup is ex-

plained in Section 6 and the results are presented in Section 7. We conclude the paper in Section 8.

2 Related work

Several research efforts have recently focused on improving the quality of service (QoS) of HTTP-based adaptive video streaming. One strategy that is frequently explored is the use of sender-driven rate adaptation. Lam et al. [8] for instance proposed a rate adaptation model in which the client-side buffer level is estimated in the server side and bitrate adaptation is done by keeping the buffer level above a certain threshold. Our study is related to the receiver-driven rate adaptation in which the client is responsible to choose the best available quality level based on the estimated network conditions.

The ultimate goal of any decision making about the chunk selection is to enhance the quality of experience (QoE) of the user. There are many factors that may effect the QoE, but not all of them will have the same effect. How to optimize QoE in DASH scenario remains an open problem. Using subjective quality assessments, McCarthy et al. [9] concluded that mobile users prefer high resolution more than high frame rate when streaming high motion videos such as sports materials. This finding suggests that picture quality is a very important factor for QoE. Other researchers have found [10] that the frequency of video freezing, i.e., playback deadline miss for the chunks, is the main responsible factor for the variations in QoE. Therefore, in this paper, we consider the frequency of video freezing and average picture quality to compare different streaming algorithms.

There have been several heuristic strategies proposed in literature [6, 11] that enable the client device to control the bitrate adaptation. With some minor variations, most of these approaches use the receiver buffer occupancy level as the primary effective parameter for the rate adaptation. As such, these strategies are often slow to react to sudden changes in network conditions. In this paper, we make use of a *revenue function* which allows us to assign tuneable weights to not only buffer occupancy but also other effective parameters such as deadline misses and number of quality changes. Li et al. in [12] presented a model called PANDA to better probe and adapt to the network conditions in order to reduce the instability of bitrate selection. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP has been reported recently by Akhshabi et al. [13]. They used the Netflix player, Adobe OSMF player and Microsoft smooth streaming to evaluate the rate adaptation algorithms in their experiments. However, none of these methods considered bandwidth statistics to predict the network conditions in different parts of client's route in a fast moving environment.

Sobhani et al. [14] designed an intelligent rate adaptation controller to provide smoother streaming by considering both throughput and buffer size. In their model, they used a scheduler unit for making the decision about downloading or postponing the download of next video chunk in order to prevent buffer overflow. However, their model is based on Fuzzy logic when we use different MDP algorithms to make the intelligent decision makings. They also used exponentially weighted moving average (EWMA) of currently observed bandwidth to estimate the available throughput while we use bandwidth statistics.

Yao et al. [15] and Deshpande et al. [16] have collected bandwidth traces from 3G networks while driving in a car and using an entropy-based method have shown that road-segment-based statistics contain more information about bandwidth compared to global statistics. Authors of [17, 18] have successfully implemented and tested a platform that allows streaming clients to access bandwidth statistics history of a given location in real-time, giving evidence that location-based streaming optimization is technically viable. Yao et al. [19], Halvorsen et al. [17], Curcio et al [20], and Singh et al. [18] have investigated the use of location-based bandwidth statistics to improve video streaming performance. However, none of these approaches have considered the MDP optimization framework that is used in our work.

Cuetos and Ross [21] used MDP to jointly optimize scheduling and error concealment in layered video, while Mastronarde et al. [22] have recently demonstrated that the power consumption problem of video decoding can be effectively modelled as an MDP. In [23], the authors used MDP to model the prefetching decision problem in adaptive multimedia. In [24, 25], the authors have proposed to use MDP to optimize rate adaption of streaming video where the uncertainty in network bandwidth is modelled as a Markov chain with its own bandwidth states. The MDP formulation we used in this paper is closest to the ones reported in [26, 27]. However, the authors of [26, 27] have neither considered comparing the MDP benefits against the non-MDP solution considered in this paper nor they have evaluated the value of using Q-learning in their MDP implementations.

The basic MDP formulation of DASH was also presented in our earlier publication [28]. However, the current work extends [28] in the following significant ways. First, we have included a comparison of MDP against a well-known non-MDP solution. Second, we have proposed and evaluated a Q-learning-based MDP implementation that does not require any *a priori* bandwidth knowledge. Third, to gain better insights to the working principles of Q-learning-based MDP, we compared its dynamics against a *completely random* as well as a *completely deterministic* decision maker. Finally, unlike [28], which used only the Big Buck Bunny clip, we have used a publicly available DASH dataset of 4 different video clips for evaluating the proposed MDP algorithms.

3 MDP Formulation of DASH

An MDP is a tuple $(S, A, P_{sa}, R_{sa}, \gamma)$, where S is a set of *states*, A is a set of *actions*, P_{sa} is a transition probability distribution over the state space when action a is taken in state s, R_{sa} is the immediate *revenue* for taking action a in state s, and $\gamma = [0, 1)$ is a discounting factor for the revenues collected from future actions and states. The solution of the MDP is a policy that tells us how to make a decision, i.e., choose an action when a particular state is observed. There are many possible policies, but the goal is to derive the *optimal policy* that maximises the expected revenue by considering the immediate as well as the future discounted revenues. In this work, we consider two different methods to derive the optimal policy, namely value iteration and Q-learning. They have different strengths and weaknesses and our goal is to compare their performances in the context of DASH quality decision making. Before we present these two methods, we explain how we obtain the MDP parameters for a DASH client.

System states and decision timings: We observe the system state when a video chunk is completely downloaded. The system state $s(\rho, a)$ is jointly represented by the quality level (q) of the downloaded chunk and the amount of time available (ρ) before its playback deadline. There is a deadline miss if the chunk download is not completed before its deadline ($\rho < 0$), in which case the video is frozen for a while until the chunk is downloaded, and it is played immediately at that time. Therefore, for a deadline miss, ρ is considered zero instead of negative.

If there is not enough space remaining in the buffer for another chunk after storing a downloaded chunk, the decision making and start of downloading the next chunk stall until there is enough room in the buffer. The value of ρ therefore assumes the value at the time of decision making (when there is space in the buffer) instead of when the last chunk was downloaded. This provides an upper bound for ρ, which is basically controlled by the buffer size. For example, if we have a buffer with a capacity to hold 7 chunks each 2 seconds long, then the upper bound for ρ is 14 seconds. Note that chunks have different sizes based on the quality while in most practical systems a buffer will have a maximum size in terms of bytes. One way to address this issue would be to configure the buffer size in bytes using the maximum chunk size, but measure buffer occupancy in units of chunks stored in the buffer. As most of today's mobile devices have enough storage capacity to download the entire video clips, the buffer length is no longer a concern. Although it is beneficial to increase the buffer length for streaming in higher quality levels [29], yet it is required to be strictly limited in order to save the network capacity. Therefore, we chose to consider the video length (i.e., number of chunks) rather than its bit size to limit the buffer length of video player [30].

Although ρ is a continuous number between zero and upper bound, we propose to use a discrete interval system to achieve a finite number of MDP states. We divide each second into n intervals and use an integer to represent the value of ρ. For example, for a 7-chunk buffer size holding only 2-sec chunks, $n = 2$ would give us $7 \times 2 \times 2 = 28$ different values for ρ. Clearly, the higher the value of n, the larger the state space becomes, requiring the longer for MDP to be solved. Fig. 3.1 shows the impact of n on the processing times (PT).

Actions: At each state, the decision taken is referred to as an action. For our adaptive HTTP streaming system, an action is basically a decision about the quality level for the next chunk. If we have N quality levels to choose from, then we have N possible actions. Each action will yield a different probability for completing the download of the next chunk at a specific time interval and hence specific value for ρ for the following state. Clearly, an action chosen (decision made) at the current state will influence the transition probability of reaching to a specific state at the next step.

Transition probabilities: Fig. 3.2 illustrates a fundamental aspect of transition probabilities in MDP using a buffer size of 7 chunks and the option of choosing from two different quality levels, i.e., two actions, a1 and a2. It shows that the probability of reaching to a particular buffer state depends on what action is taken. Given the action taken, some transition probabilities will be clearly zero. For example, if the decision is to download the next chunk in quality level 3, then in the next step, we are only concerned with calculating the transition probabilities for states with quality 3; the transition probability to reach any state with quality level other than 3 would be zero. Given the size of

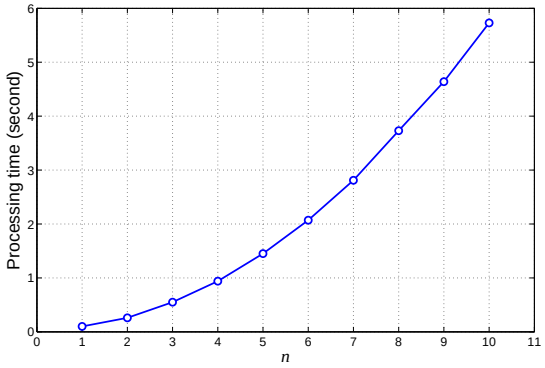


Figure 3.1: Computation complexity of MDP2: Processing time (PT) in second to solve MDP with different values of n. MDP computation was run on a laptop with an i5-3320M-2.60GHz CPU and 8GB RAM.

a chunk is known, the probability that in the next step the system will arrive at a state with a specific value for ρ can be calculated using the cumulative distribution function (CDF) of the underlying network bandwidth as follows. For T-sec video chunks in N quality levels, and assuming a buffer capacity of M chunks with n discrete intervals per second for measuring the value of ρ, the transition probability from state $s(i,x)$ to state $s'(j,y)$ can be obtained using the following equation, which for $1 \leq q \leq N$, yields a 3D matrix of size $\{(M \times T \times n + 1) \times N\} \times \{(M \times T \times n + 1) \times N\} \times N$:

$$P^q_{(i,x)(j,y)} =$$

$$\begin{cases} 0 & 1 \leq x \leq N, y \neq q, 0 \leq i \leq M \times T \times n, 0 \leq j \leq M \times T \times n \\ P^q_{ij} & 1 \leq x \leq N, y = q, 0 \leq i \leq M \times T \times n, 0 \leq j \leq M \times T \times n \end{cases} \tag{3.1}$$

where P^q_{ij} is calculated as:

$$P^q_{ij} = \begin{cases} 0 & \text{if } \begin{cases} 0 \leq i \leq (M-1) \times T \times n \\ T \times n + i \leq j \leq M \times T \times n \end{cases} \\ P^q(T \times n + i - j) & \text{if } \begin{cases} 0 \leq i \leq (M-1) \times T \times n \\ 1 \leq j \leq T \times n + i \end{cases} \\ 1 - \sum_{x=1}^{T \times n + i - 1} P^q(x) & \text{if } \begin{cases} 0 \leq i \leq (M-1) \times T \times n \\ j = 0 \end{cases} \\ P^q_{((M-1) \times T \times n)(j)} & \text{if } \begin{cases} (M-1) \times T \times n < i \leq M \times T \times n \\ 0 \leq j \leq M \times T \times n \end{cases} \end{cases} \tag{3.2}$$

and $P^q(x)$ is calculated as:

$$P^q(x) = \begin{cases} 1 - F(n \times S(q)) & \text{if } x = 1 \\ F\left(\frac{n \times S(q)}{(x-1)}\right) - F\left(\frac{n \times S(q)}{(x)}\right) & \text{if } x > 1 \end{cases} \tag{3.3}$$

where $S(q)$ is the size of a chunk in quality q and $F()$ is the CDF of the underlying network bandwidth. Note that downloading of next chunk starts

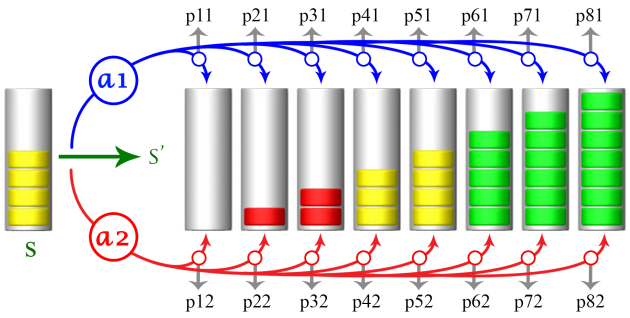


Figure 3.2: Transition probabilities in MDP for choosing different actions.

immediately if buffer is not full ($i \leq (M-1) \times T \times n$), but delayed when the buffer is full ($i > (M-1) \times T \times n$). The amount of delay will vary depending on the current state (relative progress or the value of i), but the downloading of next chunk will commence as soon as the buffer has a space ($i = (M-1) \times T \times n$), i.e., it changes its status from full to non-full, irrespective of the amount of delay. This means that for all $i > (M-1) \times T \times n$, the transition probabilities are identical and they are equal to the ones with $i = (M-1) \times T \times n$. Consequently, $T \times n + 1$ rows of the transition probability matrix will be identical.

Revenue function: The *Revenue function* $R^q(i,x)$ uses some rewards and penalties to evaluate the outcome when action q is chosen at state $s(i,x)$:

$$R^q(i,x) = u(q) - d(i,q) - c(x,q) \tag{3.4}$$

where $u(q)$ is a reward to watch a chunk in quality q, $d(i,q)$ is a penalty if a deadline is missed, and $c(x,q)$ is a penalty for changing a quality level from the last chunk to the next. Note that $u(q)$ and $c(x,q)$ can be derived immediately from predefined tables without observing or knowing the next state. The deadline miss penalty $d(i,q)$, however, depends on the next state and can be derived as a function of the probability that the next chunk will miss its deadline:

$$d(i,q) = \left\{1 - \sum_{x=1}^{T \times n+i} P^q(x)\right\} \times D \tag{3.5}$$

where $\left\{1 - \sum_{x=1}^{T \times n+i} P^q(x)\right\}$ is the probability of missing the deadline (the probability that the next chunk does not arrive in any of the intervals before the deadline) and D a constant that we can use to tune the MDP model. We can reduce the number of deadline misses by selecting a large value for D, and vice versa. If the solution method does not have access to the transition probabilities, which is the case with Q-learning as will be explained later, we can wait until the next state is observed and derive the deadline penalty as follows:

$$d(i,q) = \begin{cases} 0 & if \quad no\ deadline\ miss \\ D_Q & \quad\ otherwise \end{cases} \tag{3.6}$$

where D_Q is a constant that can be used to tune the MDP model.

It should be clear that the parameters of the revenue function allow us to tune the MDP algorithm according to the importance of certain outcomes. For

example, if deadline miss is to be absolutely avoided, then a large value must be selected for D or D_Q. Similarly, if watching the video in the highest quality is of prime interest, then large reward values should be chosen in $u(q)$ for the largest q.

4 MDP Solution Methods and Algorithms

As mentioned in the previous section, there are two well known methods for solving an MDP, value iteration and Q-learning. Value iteration assumes *a priori* knowledge of state transition probabilities and using it to derive the optimal policy before the system starts its operation. On the other hand, Q-learning assumes no *a priori* knowledge of transition probabilities, but *learns* the optimal policy on-line as actions are taken and states are observed. In essence, these two methods allow trading *learning time* with *a priori* knowledge. We briefly introduce these two methods followed by a set of proposed algorithms for a DASH client to make decisions using MDP.

4.1 Value Iteration

The core idea is to use the transition probabilities to work out the future states and then calculate the expected total revenue or *value* $V(s, a)$, i.e., for a given action a taken in a given state s [31]. The action that provides the maximum revenue for a given state s is selected as the optimal action for that state. Optimal actions for all states then constitute the optimal policy. The values can be computed using the following algorithm that repeats the process until it converges, i.e., no value change by more than a small number δ:

1. For each state-action pair, initialise $V(s, a) = 0$.
2. Repeat until convergence
   $V(s, a) = R(s, a) + \max\gamma \sum_s P(s, a, s')V(s', a')$.

As shown in the previous section, the transition probabilities for DASH are basically obtained from the bandwidth CDF. Value iteration, therefore, requires *a priori* knowledge of bandwidth CDF, which may or may not be available to a DASH client. Next, we show how Q-learning can make optimal decisions without having to know the transition probabilities of bandwidth CDF.

4.2 Q-learning

In this technique [32], a Q matrix which defines the values of every state s if action a is taken is initialised with zero. Whenever the DASH client starts out in state s, takes action a, and ends up in state s', it updates $Q(s, a)$ as [1]:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[R(s, a, s') + \gamma max_a Q(s', a')]$$

where α is the learning rate. When state s is observed, the Q matrix, or the particular row in the matrix for state s, is used to make the decisions in the

[1] Note that we use the expanded notation $R(s, a, s')$ because the next state has to be observed in Q-learning before the revenue can be calculated.

following way: choose the action that provides the maximum value based on the current estimates in Q for *most of the time*, but a *random action the rest of the time*. We use the *Boltzmann distribution* function for this probabilistic decision making purpose in the following way:

$$f(s,a) = \frac{e^{Q(s,a)/\theta}}{\sum_j e^{Q(s,a_j)/\theta}} \tag{4.1}$$

where $f(s,a)$ is the probability of selecting action a when in state s, and θ, often referred to as temperature, controls the degree of randomness in choosing an action. With large θ, actions will be fairly randomly selected irrespective of the values in Q, but for a small θ closer to zero, the best action based on Q values will be selected with high probabilities. For a DASH client with no *a priori* knowledge of the bandwidth, it is useful to begin with a large θ so the client makes mostly random decisions and learns from its observations. As it continues to learn, the Q values are used more often to make the decisions.

4.3 Quality Selection Algorithms for DASH Clients

In the previous subsections, we reviewed two well known methods to derive the optimal decision making in an MDP. In this section, we propose three practical algorithms for a DASH client based on these methods. The first two algorithms, MDP1 and MDP2, are based on value iteration, while MDP3 uses Q-learning.

MDP1: For this algorithm (Algorithm 1), we assume that the bandwidth model or bandwidth map [19], i.e., the distribution and its parameters, of every road segment is continuously measured, studied, and made available as a service by a third party. The DASH client simply uses the bandwidth CDF available from such bandwidth maps to solve the MDP once at the beginning of a trip, or until there is an update of the bandwidth map, using the value iteration. It then uses the optimal policy, which is a deterministic mapping between states and actions (quality levels), throughout the trip to select video quality level.

Algorithm 1

1: Load the bandwidth map of the route
2: Calculate P_{sa} and derive optimal policy π using *value iteration*
3: Download the first chunk using the lowest quality
4: REPEAT
5: Calculate state s using remaining time to the playback deadline
6: Select quality level for next chunk as $a = \pi(s)$
7: Download the next chunk using quality a
8: CONTINUE

MDP2: While bandwidth maps provide the *a priori* knowledge necessary for value iteration, such services may not be always available. However, a DASH client could use its own experiences to build a personal bandwidth map [33] incrementally as the user travels the same route over and over again. We propose MDP2 (Algorithm 2) to take advantage of such personal bandwidth maps to solve value iteration each time the map is updated significantly, which could occur even inside a given trip.

MDP3: Finally, we propose MDP3 (Algorithm 3) which uses *Q-learning* to make optimal decisions without having to know the bandwidth models.

Algorithm 2

Load the current version of the personal bandwidth map
Calculate P_{sa} and derive optimal policy π using *value iteration*
Download the first chunk using the lowest quality
REPEAT
  Update personal bandwidth map
  IF bandwidth map changes significantly
    Recompute optimal policy using *value iteration*
  ENDIF
  Calculate state s using remaining time to the playback deadline
  Select quality level for next chunk as $a = \pi(s)$
  Download the next chunk using quality a
CONTINUE

Algorithm 3

Initialize Q to zero
Download the first chunk using the lowest quality and observe next state
REPEAT
  Update Q
  Calculate Boltzmann probability and use it to select quality q of next chunk
  Download the next chunk using quality q and observe next state
CONTINUE

5 Non-MDP Adaptation

While there exists many streaming algorithms, the rate adaptation algorithm [6] has been considered as a representative of non-MDP solutions for DASH in this paper due to its recent references in the literature (e.g., [34], [35], and [36]). In the following, we provide a brief overview and discuss some modifications that were made while implementing this scheme.

This algorithm generally uses two different parameters to switch the quality *up* or *down*. However, as a result, this only achieves one fixed trade-off point among streaming quality dimensions (i.e., average quality, quality change and deadline miss). In our MDP approach, by varying the weights of quality change and deadline miss penalties, we get a wide range of streaming options with various combinations of the quality dimensions. The single result of *rate adaptation* algorithm can be one of the MDP results which in first place shows the advantage of MDP. We chose to vary the switch *up* and *down* thresholds to achieve more streaming performance options and have a more fair comparison with the MDP strategy as following:

First, we define μ as the ratio of media segment duration to segment fetch time which detects congestion and probes the spare network capacity as:

$$\mu = \frac{MSD}{SFT} \tag{5.1}$$

where MSD denotes the media segment duration and SFT is the current segment's average fetching time. Then, we use the *switching up* and *down* for bitrate adaptation as:

$$switch\ up \qquad if \qquad \mu > (1+\epsilon) \times \alpha \tag{5.2}$$

where α is a tuning parameter and used to vary the *switch up* and achieve various combination of the three streaming quality dimensions. In our simulations we varied α from 0 to 1 with the incremental step of 0.1 ($\alpha = 1$ in the original paper). ϵ is the switch up factor which can be determined as:

$$\epsilon = max \left\{ \frac{Br(i+1) - Br(i)}{Br(i)}, \forall i = [0, 1, ..., N-1] \right\} \tag{5.3}$$

where Br(i) denotes the encoded media bitrate of quality level (i) and N is the highest available quality level.

$$switch\ down \qquad if \qquad \mu < \lambda \tag{5.4}$$

where λ is the switch down threshold that effects the streaming performance. In simulations we vary λ from 0 to 1 with 0.1 incremental step ($\lambda = 0.67$ in the original paper). We also need to limit the maximum amount of buffered media data in order to save network bandwidth consumption as well as memory resources of the receiver. Therefore, we use the idle time calculation to prevent the client buffer overflow as prescribed in the original paper [6].

6 Simulations

In this section, we outline the simulation setup that we have used to evaluate the proposed video streaming approach. We first provide an overview of the vehicular mobile bandwidth dataset that are used in the evaluations. This is followed by the details of the video clips and video dataset used to drive the simulations for MDP1, MDP2 and MDP3 clients. Finally, we present the parameters used by the MDP to derive the optimal strategies.

In our MATLAB simulator, we use pre-collected bandwidth traces and start to fetch the first video chunk at time 0 and quality q. Then, the fetching time is calculated based on current available bandwidth in the testing trip and the system time gets updated. For each video chunk we choose one of the available bitrates based on streaming policy of the algorithm we use. Three video quality dimensions, i.e., bitrate, stability (quality changes) and freezing (deadline miss) are stored after receiving every video chunk. The average of these numbers over multiple testing trips will be used in our evaluations.

6.1 Vehicular bandwidth dataset

We use the real-world vehicular bandwidth traces collected empirically by Yao et al in our experiments. [15][1]. The dataset provides measurement of the downlink mobile bandwidth at approximately every 10s while driving along a route in the city of Sydney. The route is 24 Km long and typical drive time ranges from 22 to 30 minutes. The bandwidth measurements were tagged with the GPS coordinates and time. Measurements were conducted simultaneously for two 3G providers. In this paper, we use the traces from one provider. Table 6.1 illustrates an example of 6 measurements in a given trip. Column one represents the time when samples are recorded, column two and three are the geographical

[1]This dataset is available on-line at:
http://www.cse.unsw.edu.au/∼mahbub/PDF_Publications/Sydney_bandwidth_2008.zip.

Table 6.1: Illustrative Example of Bandwidth Traces

	time	latitude	longitude	bandwidth (Kbps)
1	1186549400	-33.919785	151.228913	1663.1440
2	1186549410	-33.919635	151.227787	1964.7330
3	1186549420	-33.91958	151.227322	2038.8659
4	1186549430	-33.91958	151.227322	2011.2631
5	1186549440	-33.91953	151.22692	1838.6578
6	1186549450	-33.91905	151.226322	1208.2767

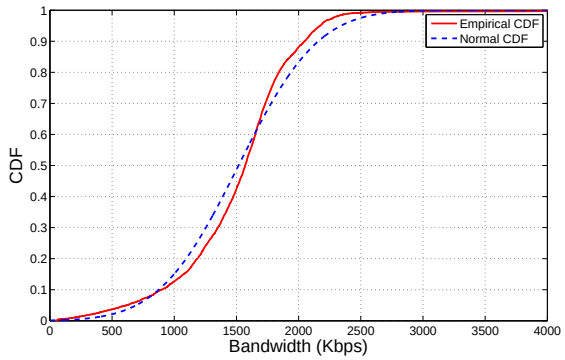


Figure 6.1: Normal Vs. Empirical CDFs of bandwidth samples.

coordinates and last column is the measured downlink bandwidth. 71 repeated trips were made along this route, resulting in a total of 71 bandwidth traces. This dataset therefore provides a rich set of historical bandwidth observations for a vehicular route. More detailed descriptions of the dataset including the hardware and the measurement methodology are available from [15, 19].

In our simulations, data from 65 traces, totalling 12649 samples, are used to build the bandwidth statistics. We first investigate if the bandwidth can be approximated by a known distribution. Assuming a normal distribution we compute the mean and standard deviation of the collective samples, $\mu = 1518.35$ Kbps and $\sigma = 503.10$ Kbps. In Fig. 6.1, we plot the empirical CDF of the bandwidth and compare it with the CDF assuming that the bandwidth is normally distributed (using the above computed μ and σ). One can readily observe that two CDFs are very similar thus indicating that the bandwidth can be assumed to be normally distributed. This is an important observation because it allows the MDP1 client to simply store the μ and σ of each given route and use them for calculating the normal CDF. This also simplifies the calculation of the transition probabilities (P) and revenues (R) as discussed earlier in Section 4. This procedure, which is adopted in our simulations, significantly reduces the implementation complexity of the MDP1 strategy.

6.2 Video Clips and DASH Dataset

We carried out two sets of simulations each with different set of video clips. In the first instance, we used the Big Buck Bunny movie [37]. The original clip was only 9:56 min long, but we repeated the movie until the end of a trip. We created five different quality versions of the movie. We used ffmpeg [38] to encode the original video file using bit-rates of 186 kbps (*quality 1*), 499 kbps

Figure 6.2: Five different quality levels for Big Buck Bunny.

Table 6.2: Mean, standard deviation, and coefficient of variance (Cv) of 2-sec chunk size (in Kb)

	$q1$	$q2$	$q3$	$q4$	$q5$
μ	375.29	938.77	2027.54	2360.88	3513.08
σ	10.91	122.22	255.82	351.84	874.84
Cv	0.03	0.13	0.13	0.15	0.25

(*quality 2*), 1101 kbps (*quality 3*), 1292 kbps (*quality 4*) and 1898 kbps (*quality 5*). Fig. 6.2 illustrates one particular frame of this video encoded at these 5 different quality levels. Each video stream was multiplexed with a common 128 kbps audio file to form a corresponding single MPEG-2 TS stream. The resulting stream is divided into 2 second chunks. We compute the average and variance of the chunk sizes for each quality of the movie. As observed in Table 6.3, the variance is small, so in our simulations we use the average chunk size instead of their exact sizes for all of video chunks (EQ. 3.3 shows how chunk size is used in calculating transition probability). This video clip is used to illustrate all the major results obtained in this paper.

For the second set of simulations we used a publicly available DASH dataset [39]. This dataset has video chunks and manifest files available for many different video clips. We used data for the following 4 different video clips: Elephant Dream, Of Forest and Men, The Swiss Account, and Valkaama. The details of these clips are shown in Table 6.3.

6.3 MDP Parameters

As explained in the previous sections, MDP is a flexible optimization framework, the outcome of which can be influenced to prioritise any of the three dimensions

Table 6.3: Mean, standard deviation, and coefficient of variance (Cv) of 2-sec chunk size (in Kb) for 5 different video clips. All video chunks and manifest files are freely available at [39])

Clips		$q1$	$q2$	$q3$	$q4$	$q5$
Elephant Dream	*Rate (kbps)*	200	500	1200	1500	2000
	μ	350.90	746.02	2016.13	2403.12	2918.83
	σ	64.53	234.43	488.93	672.67	1019.38
	Cv	0.18	0.31	0.24	0.28	0.35
Of Forest and Men	*Rate (kbps)*	200	500	1100	1400	2000
	μ	362.03	923.58	1960.10	2471.02	3439.16
	σ	63.32	438.16	402.21	544.97	902.57
	Cv	0.17	0.47	0.21	0.22	0.26
The Swiss Account	*Rate (kbps)*	200	500	1200	1500	2000
	μ	340.40	845.07	1890.00	2575.12	3376.19
	σ	45.66	196.09	387.60	399.81	572.13
	Cv	0.13	0.23	0.21	0.16	0.17
Valkaama	*Rate (kbps)*	200	500	1100	1400	2000
	μ	342.67	861.24	1933.92	2418.73	3646.90
	σ	48.49	183.47	261.34	381.74	444.59
	Cv	0.14	0.21	0.14	0.16	0.12

of streaming performance (i.e., picture quality, quality change and deadline miss) by adjusting its reward and penalty parameters. For MDP1 and MDP2 strategies, we keep the reward parameters fixed as shown in Table 6.4, but vary the deadline (D) and quality change (C) penalties. The value of D is varied between 2 and 5000 and C is varied as a factor of the base values shown in Table 6.5 (we considered 10 different factors between 0.1 to 1.9). The values of other MDP parameters are: N= 5, M= 7 and n= 2 andγ (discount factor)= 0.9. For MDP3, we change the reward to $u(q) = u(q) \times 10$, to encourage the Q-learning selecting higher quality levels. $\alpha = 0.9, \gamma = 0.9$, θ is initiated with 15 and $\epsilon = 0.005$. All of these values are found to be the most appropriate tuning options among many trials.

Table 6.4: Reward function

quality level (q)	**1**	**2**	**3**	**4**	**5**
$u(q)$	1	2	4	7	10

Table 6.5: Base penalty values for changing quality level from i to j

$i \backslash j$	**1**	**2**	**3**	**4**	**5**
1	0	1	5	10	25
2	10	0	1	5	10
3	50	10	0	1	5
4	250	50	10	0	1
5	500	250	50	10	0

7 Results

In this section, we present the simulation results to demonstrate the efficacy of MDP-based adaptive streaming. First, we explain the inherent flexibility of MDP that allows the client to control the trade-off between deadline miss and the picture quality. Second, we provide a quantitative comparison between

Table 7.1: MDP streaming performance as a function of penalty values. The columns show different values of quality change penalty and the rows show the deadline miss penalties.

D\C	0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
	195.4	202.8	207.0	230.4	268.8	277.4	280.8	282.6	282.8	286.4
10	4.74	4.74	4.75	4.77	4.80	4.81	4.81	4.82	4.81	4.81
	47.60	34.00	26.40	20.80	14.60	13.80	13.00	12.60	12.00	10.80
	66.80	62.60	60.00	59.80	55.20	52.40	54.20	51.40	48.40	43.20
15	4.58	4.57	4.57	4.56	4.55	4.54	4.54	4.52	4.50	4.44
	61.40	48.80	39.80	34.20	32.00	31.20	30.40	29.20	27.40	24.60
	48.00	46.80	47.80	44.20	39.60	41.60	36.20	33.40	30.20	27.60
20	4.55	4.55	4.54	4.54	4.53	4.52	4.47	4.43	4.37	4.32
	64.00	46.20	39.60	34.60	32.40	32.00	28.40	26.00	23.00	20.60
	40.80	36.60	37.60	37.40	34.40	30.80	31.20	26.60	26.80	25.60
24	4.53	4.53	4.53	4.52	4.50	4.45	4.41	4.34	4.30	4.25
	65.60	47.20	40.20	37.20	33.20	28.80	27.20	21.60	20.20	17.00
	28.00	32.40	32.60	32.20	33.40	30.20	27.60	24.60	24.00	20.00
27	4.51	4.52	4.51	4.50	4.47	4.41	4.33	4.29	4.27	4.21
	73.40	47.20	41.80	37.20	33.60	26.80	23.60	20.00	20.20	16.20
	22.40	27.60	29.80	29.60	26.60	25.60	23.60	24.00	23.20	21.80
30	4.50	4.50	4.50	4.47	4.42	4.35	4.30	4.28	4.23	4.20
	78.80	51.20	44.80	36.40	30.40	26.00	21.60	20.80	17.80	16.20
	10.60	13.20	17.60	16.80	15.00	15.20	16.40	17.40	16.60	16.60
50	4.44	4.42	4.35	4.27	4.22	4.20	4.18	4.17	4.16	4.15
	87.20	72.60	54.80	42.60	28.00	26.40	21.60	18.40	17.80	16.20
	7.40	8.40	12.20	11.80	12.00	12.00	11.60	12.60	12.00	12.40
70	4.41	4.36	4.26	4.19	4.18	4.15	4.14	4.14	4.12	4.11
	98.20	73.80	52.40	42.80	33.60	29.20	26.20	24.60	23.20	22.40
	5.20	6.20	6.20	5.80	6.80	6.60	6.80	9.20	8.40	8.00
100	4.37	4.26	4.20	4.14	4.12	4.11	4.10	4.09	4.06	4.05
	107.2	65.40	47.00	39.60	35.60	33.60	28.80	26.80	26.00	23.40
	4.40	5.20	5.80	5.00	5.20	5.00	4.60	5.20	4.60	6.60
130	4.31	4.22	4.14	4.12	4.10	4.09	4.08	4.04	4.02	4.03
	107.2	62.80	41.20	36.80	36.00	32.20	27.60	25.20	23.40	22.40
	4.00	5.80	5.60	6.00	5.00	4.20	4.20	4.80	4.20	4.60
150	4.28	4.18	4.12	4.11	4.09	4.08	4.05	4.01	4.01	4.02
	107.0	57.20	40.60	38.20	36.20	33.00	29.20	24.40	23.00	23.80
...	...	...	...	...	...	...	...	...	...	...

MDP-based and the state-of-the-art non-MDP-based adaptation strategies. Finally, we compare the performance of model-based vs model-free MDP strategies.

7.1 Deadline Miss Vs. Quality Trade-off with MDP

Table 7.1 shows the average streaming performance of MDP2 client for various combinations of deadline miss penalty D (rows) and quality change penalty C (columns). For each combination of D and C, the three real numbers represent the performance in three dimensions. The top number represents the number of deadline miss (DM) for the video session, the middle represents the average chunk quality (AQ), and the bottom represents the number of times quality was changed (QC) in the session. All these numbers are averaged over six test trips, from trip 66 to trip 71, streaming the Big Buck Bunny video clip. Note that each deadline miss will cause the video to freeze, which is considered one of the most important quality of experience metrics for video streaming.

Table 7.1 clearly demonstrates the extensive opportunity for tuning the performance of MDP. For instance, by setting a high value for D, say D=150 (last row), one can keep the deadline misses to a minimum (only 4 deadline misses for the entire trip). Moreover, for a selected value of D (say D = 150), one can either set a small value for C (say 0.1) which affords a high quality viewing experience (average video quality level of 4.28) but with a large number of quality changes (107), or chose a large value for C (1.9) which significantly reduces the number of quality changes (23.8) albeit with a slightly lower AQ of 4.02.

Fig. 7.1 further illustrates the flexibility afforded by MDP using a scatter plot of the DM and AQ values obtained from many different combinations of D and C settings. In particular, we observe that for a given AQ it is possible to trade-off between DM and QC (the size of circles indicates the QC). See for

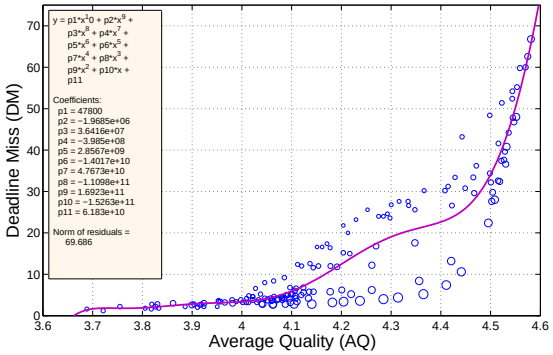


Figure 7.1: Scatter plot of AQ and DM data. The size of each circle corresponds to the number of QC with larger circles denoting more QC, and vice versa.

example AQ values between 4 and 4.5. In this interval, we have different DM for the same AQ, but smaller DM is achieved with larger QC, as the system has to switch to a lower quality more often in order to avoid a potential deadline miss. We further observe that MDP yields a non-linear trade-off between video quality and deadline misses with the number of deadline misses increasing rapidly if we try to watch the video in very high quality. It is however difficult to find a good fit to analytically capture this tradeoff (the purple curve shows the best fit, which is a ninth degree polynomial).

For the remainder of this section, we will use this trade-off to compare various approaches to streaming adaptation. One approach would be considered better than the other if it achieves a better trade-off, i.e., less DM for similar AQ or higher AQ for similar DM.

7.2 MDP vs. non-MDP

In this section, we will consider MDP1 as a representative implementation of MDP as this is the most basic implementation options among the three. Now, to compare the performance of MDP and non-MDP algorithms, we must first find a way to derive the deadline misses vs. quality trade-off curves for both algorithms. As explained in Section 5, for the non-MDP algorithm, we can study the trade-off curve by considering a range of values for the two parameters, α and λ (default values of 1 and 0.67 respectively). The lower the values of these parameters, the higher the picture quality and conversely, greater the number of deadline misses encountered. For the trips 66-71, Table 7.2 shows the multi-dimensional performance of the non-MDP algorithm (in a similar manner as Table 7.1 for the MDP algorithm) when streaming Big Buck Bunny video clip. These values are then used in Fig. 7.2a to compare the trade-off curves of the non-MDP algorithm against that of the MDP algorithm. Each resulting point in this graph is the average of 12 results in each column. The values of α and λ are varied between 0 and 1 with granularity of 0.1 first, then 18 more values between 0.5 and 0.7 with incremental step of 0.01 are added to the α values to have more result points within the quality range of 3.8 to 4.5.

We observe that the MDP algorithm provides a much better trade-off between deadline miss and quality compared to the non-MDP strategy, especially

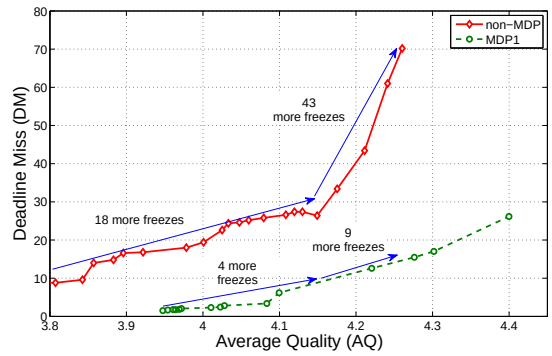
Table 7.2: Performance of the Rate Adaptation Algorithm (i.e. non-MDP) . Different results achieved by varying α and γ which are used to make the decision for switching the streaming bit-rate *up* and *down* respectively. Switch up when $\mu < (1+\epsilon) \times \alpha$ and switch down when $\mu < \gamma$. Based on Liu et all. [6] $\alpha = 1$ and $\gamma = 0.67$ are the default settings. (Video Clip: The Swiss Account)

λ, α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	**1**
0	535.6 4.99 4	535.6 4.99 4	535.6 4.99 4	535.6 4.99 4	535.8 4.98 4	530 4.97 4	428 4.76 4	370.8 4.58 3.8	353.4 4.40 3.8	351.6 4.25 3.6	350.4 4.25 3.6
0.1	535.4 4.98 6	535.4 4.98 6	535.4 4.98 6	535.4 4.98 6	533.2 4.98 6	523 4.97 6	377.2 4.67 5.8	334.4 4.45 5.6	316.6 4.26 5.6	314.6 4.17 5.6	313.4 4.17 5.6
0.2	531.6 4.98 11.6	531 4.98 12	531 4.98 12	530.6 4.98 12	528.4 4.96 11.6	489 4.90 11	321.8 4.57 10.4	200 4.16 9.4	149.4 3.85 9.4	141.8 3.69 9	140.6 3.69 9
0.3	532.2 4.97 19.2	532.6 4.97 19.6	529 4.97 18.8	529 4.96 18.8	519.6 4.94 18.2	432.2 4.81 18.6	258 4.45 16	122.8 4.02 14.6	95.2 3.62 13.2	88.4 3.50 12.8	88.4 3.50 12.8
0.4	526.4 4.97 32.8	526.8 4.96 33.2	516.2 4.95 35.2	509.6 4.94 33.2	504.4 4.89 32.6	370.8 4.72 29.8	177 4.27 25.2	81 3.82 20	37.8 3.47 17	32 3.40 15.8	32.2 3.39 15.8
0.5	513.6 4.94 62.4	513.8 4.94 63.2	502.2 4.92 69.6	492 4.89 52.8	480.2 4.84 51	318.4 4.63 42.8	134.6 4.18 33.6	61.2 3.72 27.2	29.4 3.34 23.2	27.4 3.27 21.6	27.2 3.26 21.6
0.6	502.8 4.91 115.6	502.6 4.91 116.4	485.8 4.87 128.4	473 4.82 103.6	440.4 4.73 80.2	232.8 4.48 64	67.6 4.00 49.6	18.6 3.52 37.6	9.6 3.24 31.2	10.8 3.15 28	9.4 3.13 27.6
0.67	502.2 4.87 180.4	500.4 4.87 180	477.2 4.83 192.6	458.6 4.76 176.6	407.4 4.63 112	157 4.37 88.8	25.6 3.90 62	15.8 3.48 44.4	9.4 3.21 38	8.8 3.12 35.6	**8.8** **3.10** **35.2**
0.7	493.4 4.85 203	494.8 4.85 203.8	476.2 4.82 214.2	447 4.73 205.8	394.8 4.60 121.2	134.8 4.33 96	21 3.87 66.4	13 3.46 48	8.8 3.19 39.6	8.4 3.10 36.8	8.2 3.08 36
0.8	423.8 4.76 349.4	422.8 4.76 351.4	402 4.72 362.2	358.2 4.60 388	263.6 4.44 264.8	37.2 4.14 147	9.6 3.73 96.4	8.4 3.36 62	6 3.09 51.2	5.6 3.02 47.2	5.8 3.00 45.2
0.9	310 4.65 530.6	307.4 4.65 533.4	293.8 4.61 543.8	233.4 4.47 578	83.6 4.25 477.4	11.4 3.99 189.2	7.2 3.62 124.2	6.4 3.27 80.8	5.6 3.04 62	5.6 2.97 57.6	5.4 2.95 56
1	231.6 4.58 663.6	225 4.57 665	209.6 4.54 676.4	154.6 4.40 710.4	24.6 4.12 676.2	9.2 3.84 321.2	7.6 3.48 160	6.2 3.19 109.6	5.8 3.00 77.2	5.8 2.94 67.4	5.2 2.91 65.4

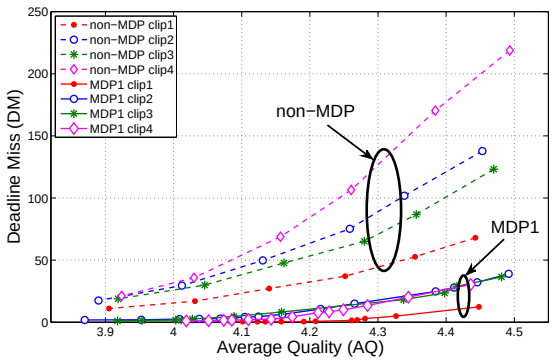
if the user wishes to watch the videos in higher qualities. For example, when watching the video at an average quality level of 4, the MDP algorithm misses only 2 playback deadlines during the entire trip on average compared to 20 for the non-MDP. Similarly, users experience 18 more DM if the AQ is increased from 3.8 to 4.15, and a further 43 if the AQ is lifted slightly from 4.15 to 4.25. In contrast, users can expect only 4 DM for an AQ increase from 3.8 to 4.15 and a further 9 for an AQ increase from 4.15 to 4.25. Overall, for the AQ interval between 3.8 and 4.3, which can be considered a high picture quality range for our experiments, the non-MDP would freeze the video 22.5 times on average in comparison to only 5.7 times caused by the MDP algorithm. This means that for high quality viewing, the MDP solution can reduce DM by a factor of 4 compared to non-MDP solutions. We have repeated the simulations with 4 other video clips of Table 6.3. As shown in Fig. 7.2b and Table 7.3, MDP significantly outperforms the non-MDP algorithm for all of these clips with the maximum and average DM reductions being 15x and 8x, respectively.

7.3 Model-based vs. Model-free MDP strategies

MDP1 requires *a priori* knowledge of the bandwidth model to arrive at the optimal decision making strategy. Because it may not be always possible to obtain this prior knowledge, we considered a model-free MDP implementation, i.e., MDP3, which is based on Q-learning. MDP3 learns the optimal decision over time as it observes the outcomes of its decisions, thus obviating the need for the *a priori* knowledge of the bandwidth model.

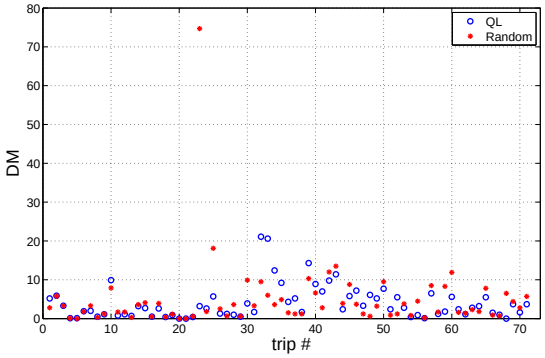


(a)

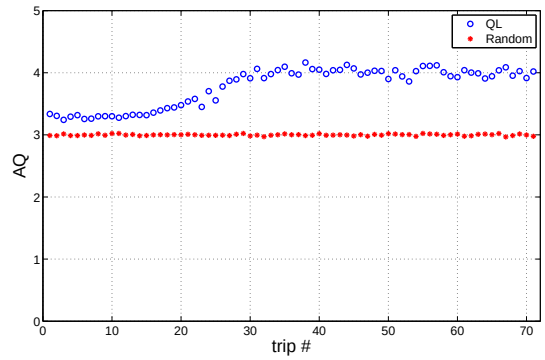


(b)

Figure 7.2: Comparison between MDP1 and non-MDP algorithms. MDP1 significantly outperforms non-MDP algorithm by achieving less DM for the same AQ. Testing trips: 66-71, (a) Big Buck Bunny, (b) Different video clips: 1- Elephant Dream, 2- Of Forest and Men, 2- The Swiss Account, 4- Valkaama



(a)



(b)

Figure 7.3: Comparing MDP3 (Q-learning) against a random decision maker (D_Q=15000, $\theta = 15$, $\epsilon = 0.0005$)

Table 7.3: Comparison between MDP1 and non-MDP algorithms: Average number of DM within AQ range of 3.9 - 4.5

		Avg DM/trip		
Video Clip		MDP	non-MDP	DM reduction
1	Elephant Dream	2.24	35.46	15.83x
2	Of Forest and Men	11.98	68.58	5.72x
3	The Swiss Account	11.69	61.91	5.30x
4	Valkaama	8.03	103.55	12.89x
μ		8.48	67.37	7.94x

In this section, we first study the performance dynamics of MDP3 and then compare it with the model-based MDP implementation that incrementally learns the model, i.e., MDP2. We choose to compare MDP3 with MDP2 instead of MDP1 because both MDP3 and MDP2 attempt to *learn* the optimal decision as more trips are being made. The difference between them is that MDP2 learns the model and uses the model to derive the optimal strategy using *value iteration*, whereas MDP3 uses Q-learning to learn optical decision making. Similar to previous results, we vary the penalty values to achieve different performance results.

Observing the dynamics of Q-learning

As mentioned in Section 4, Q-learning selects its decisions *probabilistically* from the set of available actions with the weights for different actions influenced by the current state of the Q matrix. We also explained that the probability of selecting a particular quality level (action) is further modified by the Boltzmann distribution to ensure that the MDP client makes more *random* decisions in the beginning to allow adequate learning and slowly handing it over to the Q matrix to make more *intelligent* decisions as time goes on. It would be therefore interesting to see how MDP3 compares with a decision maker that *always* make random decisions, i.e., selects any one of the 5 video quality levels with a probability of 0.2.

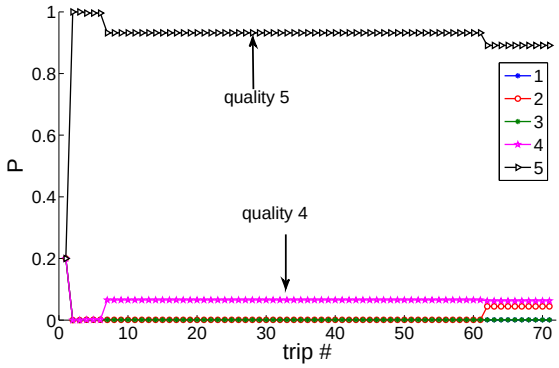
Fig. 7.3 shows that, in the beginning, MDP3 performs similar to the random decision maker. However, as more trips are made, MDP3 clearly outperforms the random decision maker in AQ (Fig. 7.3b). As it's hard to capture from Fig. 7.3a, we look at Table 7.5 where we can see that on average MDP3 misses less number of deadlines (i.e., 3.97 vs. 4.81). Table 7.4 shows that the Boltzmann probabilities are identical for all the 5 available quality levels in trip 1, but they change significantly in trip 33 as a result of the learning process. In particular, quality 5 has very high probability while the probabilities for all other quality levels are very small, implying that quality 5 will be selected most of the times. This observation clearly validates the fundamental principle of Q-learning.

Table 7.4: Boltzmann probability distribution (P)

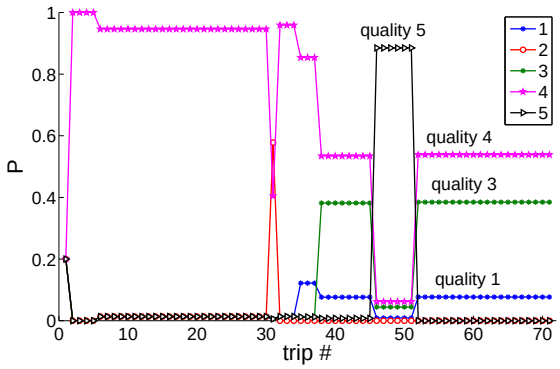
	a	1	2	3	4	**5**
seg:1, trip:1	P	0.2	0.2	0.2	0.2	0.2
seg:1, trip:33	P	0.0088	0.0088	0.0088	0.0885	**0.885**

Table 7.5: Comparing MDP3 outcomes against those of *random* and *deterministic* decision makers over 71 trips

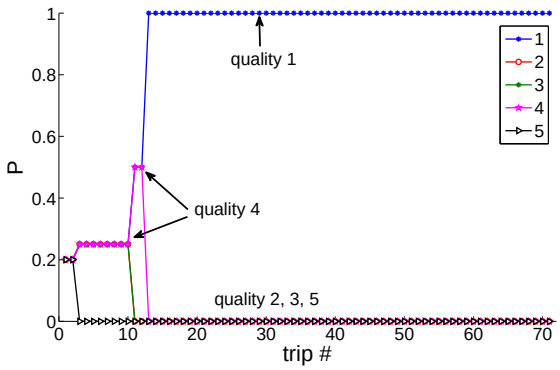
	MDP3	Random	Deterministic		
AQ	3.78	2.99	3	4	5
DM	3.97	4.81	4.35	8.38	309.1



(a) Low risk state

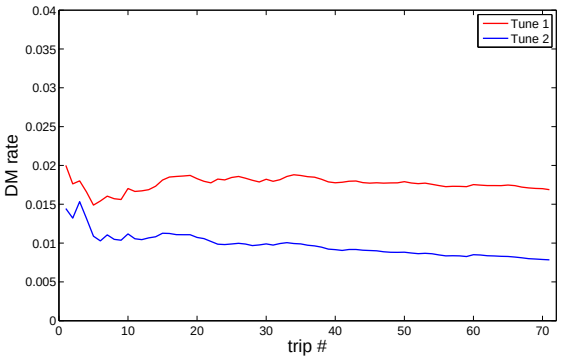


(b) Medium risk state

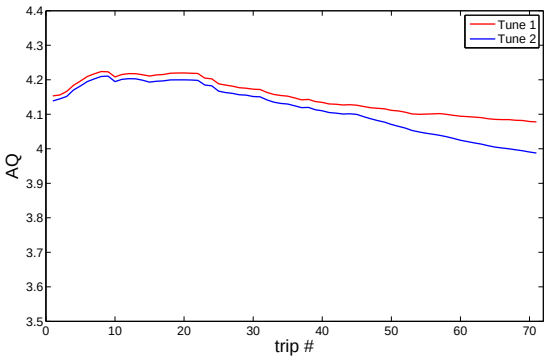


(c) High risk state

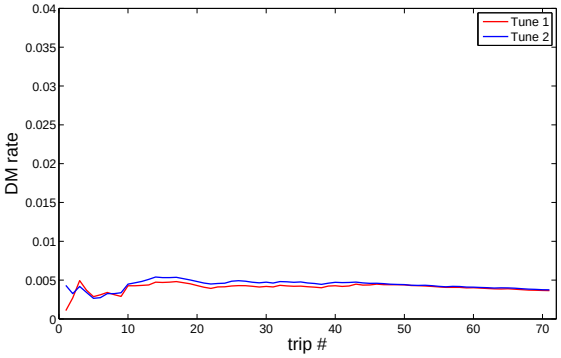
Figure 7.4: Convergence of Boltzmann probabilities in high, medium and low risk states



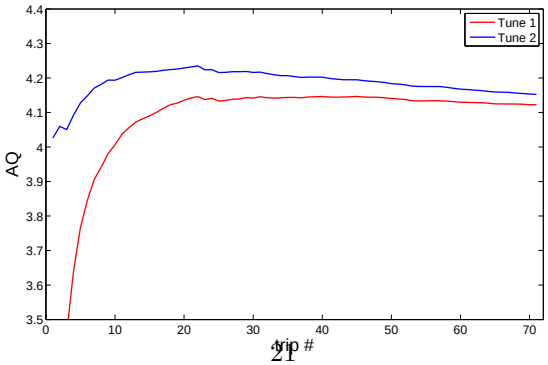
(a)



(b)



(c)



(d)

Figure 7.5: Learning with MDP2 (a, b) and MDP3 (c, d) for Big Buck Bunny

Next, we compare the Q-learning against a *deterministic* decision maker that *always* selects a particular action and never selects any other actions. Note that there is no learning involved in both random and deterministic decision makers. For the deterministic decision maker, we evaluate its performance for three different quality levels, 3, 4, and 5. Table 7.5 summarises the average results over 71 trips. As we can see MDP3 outperforms both the random as well as the deterministic decision makers, highlighting the value of Q-learning.

Finally, Fig. 7.4 shows how the Boltzmann probabilities converge over time giving higher weights to different quality levels for different states. A *low risk* state is the one when the playout buffer is full, i.e., the there is plenty of time until the playback deadline for the next chunk to be downloaded. Similarly, *high risk* means the playout buffer is empty and *medium risk* represents a half-full buffer. We see that for the low risk state (Fig. 7.4a), the probabilities converge in a way such that the probability of quality 5 is extremely high compared to all other probabilities, which allows MDP3 to select quality 5 most of the time as the risk of DM is very low. In contrast, for the high risk state (Fig. 7.4c), the situation is reversed with quality 1 having extremely high probability. For the medium risk state (Fig. 7.4b), both 3 and 4 have higher probability compared to the rest of quality levels, allowing MDP3 to achieve an AQ between 3 and 4.

Comparing MDP2 and MDP3

Next, we compare MDP3 against MDP2 to study whether Q-learning has any benefit over model learning, which could be achieved using only the *personal* bandwidth maps generated by the mobile device. Figure 7.5 plots the DM rates (average number of DM per chunk) and the AQ as a function of the number of trips made for different values of the deadline miss penalties. We make two important observations:

- MDP2 is more sensitive to parameter tuning than MDP3. This is clearly seen in the DM rates, where MDP2 achieves significantly different DM rates depending on the penalty values selected. In contrast, DM values are not sensitive to the penalty values for MDP3. While AQ values are sensitive to parameter tuning when learning occurs (at the beginning), over time they converge to similar values for MDP3. However, for MDP2, AQ values diverge for different parameter settings. This can be attributed to the fact that unlike MDP2, MDP3 makes its decisions *probabilistically*, which does not tie it *rigidly* to a particular parameter setting. This can be of advantage in vehicular context where bandwidth models cannot be known perfectly and some deviations from the model used in the derivation of the optimal strategy is more realistic.

- MDP3 reduces DM significantly compared to what can be achieved with MDP2 for the comparable AQ. Considering the case when the performance has converged, i,e, after 15 trips, we find that MDP3 achieves similar AQ, but about 3x less DM compared to MDP2. Again, this performance discrepancy can be attributed to the fact that models are only *estimates* of the bandwidth; as such, MDP2, which uses bandwidth models to derive the optimal strategy and then sticks to that strategy, is bound to suffer from any unexpected bandwidth experiences. On the other hand, Q-learning is model free, which frees MDP3 from any model related issues.

8 Conclusions and Future Works

Using publicly available DASH datasets, we have evaluated the benefits of using MDP for adaptive multimedia streaming. We have found that, comparing to a well-known non-MDP method, it reduces deadline miss rate by a factor of 4-15 when the bandwidth model is known *a priori*. We have also proposed and evaluated a model-free MDP implementation that uses Q-learning to gradually learn the optimal decisions over time. We find that MDP with Q-learning converges to optimal decision making after about 15 trips and outperforms the model-based MDP by a factor of 3 in terms of deadline miss rates. Our next target is to develop a practical implementation of a DASH player that incorporates the proposed MDP options and evaluate their performances in real road traffic using real video streams.

In this paper, we focussed on video freezing, which have been found as the most important factor affecting user QoE. However, long term stability problem [40] caused by frequent adaptation of the picture quality is another important metric that could be studied with MDP. The current MDP model can be extended to consider the instability by extending the state definition to include a stability parameter, which would define the current state of stability for the video stream. Then the penalty function will have to be extended to introduce penalties that are dependent on the current state of stability. These extensions will enable MDP to tune the stability performance of DASH streaming, albeit at higher complexity. How to best design the MDP for stability control would constitute a significant future work.

Acknowledgement

Ayub Bokani's PhD is supported by an Australian Postgraduate Award and a PhD enhance scholarship from National ICT Australia (NICTA).

Bibliography

[1] Apple, "HTTP Live Streaming Overview," [Online accessed 01-March-2013], URL: https://developer.apple.com/library/ios/#documentation/networkinginternet/conceptual/streamingmediaguide/Introduction/Introduction.html.

[2] A. Zambelli, "IIS smooth streaming technical overview," *Microsoft Corporation*, vol. 3, 2009.

[3] Adobe, "HTTP Dynamic Streaming on the Adobe Flash Platform," [Online accessed 04-March-2013], URL: http://www.adobe.com/au/products/hds-dynamic-streaming.html.

[4] T. Stockhammer, "Dynamic Adaptive Streaming Over HTTP: Standards and Design Principles," in *Proceedings of the second annual ACM conference on Multimedia systems (MMSys)*, San Jose, USA, 23-25 February 2011.

[5] T.-Y. Huang, R. Johari, and N. McKeown, "Downton Abbey Without the Hiccups: Buffer-Based Rate Adaptation for HTTP Video Streaming," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, Hong Kong, China, 16 August 2013.

[6] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the second annual ACM conference on Multimedia systems (MMSys' 11)*, New York, USA, 23 February 2011.

[7] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.com, 2009, vol. 414.

[8] L. Lam, J. Y. Lee, S. C. Liew, and W. Wang, "A Transparent Rate Adaptation Algorithm for Streaming Video Over the Internet," in *The 18th International Conference on Advanced Information Networking and Applications (AINA)*, Fukuoka, Japan, 29-31 MArch 2004.

[9] J. D. McCarthy, M. A. Sasse, and D. Miras, "Sharp or Smooth?: Comparing the Effects of Quantization Vs. Frame Rate for Streamed Video," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, Vienna, Austria, 2429 April 2004.

[10] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the Quality of Experience of HTTP Video Streaming," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, Dublin, Irland, 23-27 May 2011.

[11] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation Algorithm for Adaptive Streaming Over HTTP," in *The 19th International Packet Video Workshop (PV)*, Munich, Germany, 10-11 May 2012.

[12] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming at Scale," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 4, pp. 719–733, 2014.

[13] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming Over HTTP," in *Proceedings of the second annual ACM conference on Multimedia systems*, San Jose, USA, 23-25 February 2011.

[14] A. Sobhani, A. Yassine, and S. Shirmohammadi, "A Fuzzy-based Rate Adaptation Controller for DASH," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '15)*, Portland, USA, 20 March 2015.

[15] J. Yao, S. S. Kanhere, and M. Hassan, "An Empirical Study of Bandwidth Predictability in Mobile Computing," in *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization (MobiCom-WiNTECH)*, San Francisco, USA, 14-19 September 2008.

[16] P. Deshpande, X. Hou, and S. R. Das, "Performance Comparison of 3G and Metro-Scale WiFi for Vehicular Network Access," in *Proceedings of the 10th ACM conference on Internet measurement*, Melbourne, Australia, 13 November 2010.

[17] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 8, no. 3, p. 24, 2012.

[18] V. Singh, J. Ott, and I. D. Curcio, "Predictive Buffering for Streaming Video in 3G Networks," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, San Francisco, USA, 25-28 June 2012.

[19] J. Yao, S. S. Kanhere, and M. Hassan, "Improving QoS in High-Speed Mobility Using Bandwidth Maps," *IEEE Transactions on Mobile Computing*, vol. 11, no. 4, pp. 603–617, 2012.

[20] I. D. Curcio, V. K. M. Vadakital, and M. M. Hannuksela, "Geo-Predictive Real-Time Media Delivery in Mobile Environment," in *Proceedings of the 3rd workshop on Mobile video delivery (MoViD)*, Firenze, Italy, 25 October 2010.

[21] P. de Cuetos and K. W. Ross, "Optimal Streaming of Layered Video: Joint Scheduling and Error Concealment," in *Proceedings of the eleventh ACM international conference on Multimedia (MM '03)*, Berkeley, USA, 02-08 November 2003.

[22] N. Mastronarde, K. Kanoun, D. Atienza, P. Frossard, and M. van der Schaar, "Markov Decision Process Based Energy-Efficient On-Line Scheduling for Slice-Parallel Video Decoders on Multicore Systems," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 268–278, 2013.

[23] V. Charvillat and R. Grigoraş, "Reinforcement Learning for Dynamic Multimedia Adaptation," *Journal of Network and Computer Applications*, vol. 30, no. 3, pp. 1034–1058, 2007.

[24] M. Xing, S. Xiang, and L. Cai, "Rate Adaptation Strategy for Video Streaming over Multiple Wireless Access Networks," in *IEEE Global Communications Conference (GLOBECOM)*, Anaheim, 3-7 December 2012.

[25] S. Xiang, L. Cai, and J. Pan, "Adaptive Scalable Video Streaming in Wireless Networks," in *Proceedings of the 3rd Multimedia Systems Conference (MMSys)*, Chapel Hill, USA, 22-24 February 2012.

[26] C. C. Wüst and W. F. Verhaegh, "Quality Control for Scalable Media Processing Applications," *Journal of Scheduling*, vol. 7, no. 2, pp. 105–117, 2004.

[27] D. Jarnikov and T. Özçelebi, "Client Intelligence for Adaptive Streaming Solutions," *Signal Processing: Image Communication*, vol. 26, no. 7, pp. 378–389, 2011.

[28] A. Bokani, M. Hassan, and S. Kanhere, "HTTP-Based Adaptive Streaming for Mobile Clients using Markov Decision Process," in *Proceedings of the 20th Packet Video Workshop (PV)*, San Jose, USA, 12-13 December 2013.

[29] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "Using the Buffer to Avoid Rebuffers: Evidence from a Large Video Streaming Service," *arXiv preprint arXiv:1401.2209*, 2014.

[30] G. Zhong and A. Bokani, "A Geo-Adaptive JavaScript DASH Player," in *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming (ACM-CoNEXT '14)*, Sydney, Australia, 2-5 December 2014.

[31] J. Van Der Wal, *Stochastic Dynamic Programming.* Methematisch Centrum, Amsterdam, The Netherlands, 1980.

[32] C. J. C. H. Watkins, "Learning From Delayed Rewards." Ph.D. dissertation, University of Cambridge, 1989.

[33] G. Murtaza, A. Reinhardt, M. Hassan, and S. S. Kanhere, "Creating Personal Bandwidth Maps Using Opportunistic Throughput Measurements," in *IEEE International Conference on Communications (ICC)*, Sydney, Australia, 10-14 June 2014.

[34] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "QDASH: a QoE-aware DASH system," in *Proceedings of the 3rd Multimedia Systems Conference (MMSys '12)*, Portland, USA, 18-20 March 2015.

[35] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies (CoNEXT '12)*, Nice, France, 10-13 December 2012.

[36] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies (CoNEXT '12)*, Nice, France, 10-13 December 2012.

[37] BlenderFoundation, "Big Buck Bunny," [Online accessed 04-March-2013], URL: http://www.bigbuckbunny.org.

[38] FFmpegProject, "FFmpeg," [Online accessed 04-March-2013], URL: http://ffmpeg.org.

[39] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proceedings of the 3rd Multimedia Systems Conference, MMSys'12*, Chapel Hill, NC, USA, 22-24 February 2012.

[40] S. akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth?" in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '12)*, Toronto, Canada, 7-8 June 2012.