HTTP-based Adaptive Streaming for Mobile Clients using Markov Decision Process

Ayub Bokani Mahbub Hassan Salil Kanhere

University of New South Wales, Australia
{abokani, mahbub, salilk}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-201316
Julay 2013

THE UNIVERSITY OF
NEW SOUTH WALES

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

Abstract

Due to its simplicity at the server side, HTTP-based adaptive streaming has become a popular choice for streaming on-line contents to a wide range of user devices. In HTTP-based streaming systems, the server simply stores the video segmented into a series of small chunks coded in many different qualities and sizes, and leaves the decision of which chunk to download next to achieve a high quality viewing experience to the client. This decision making is a challenging task, especially in mobile environment due to unexpected changes in network bandwidth as the user moves through different regions. In this paper, we consider Markov Decision Process (MDP) as an optimisation framework to optimise the three dimensions of streaming performance, picture quality, deadline miss, and the frequency of quality change. We highlight that MDP has a high overhead arising from frequent strategy updates as a moving client attempts to learn the statistical parameters of the underlying bandwidth. We propose and evaluate three approaches to reduce MDP overhead considering both on-line and offline optimisation. For online, we propose a k-chunk update approach (k-MDP) which recomputes the optimal strategy after downloading every k chunks. For offline, we propose two different approaches, single MDP strategy (s-MDP) and x-meter MDP strategy (x-MDP). s-MDP uses the global statistics of a given region to compute an optimal MDP strategy which is used throughout the video session, while x-MDP recomputes the optimal strategy for every x meters of the travel using offline statistics for each x-meter of the road. We have evaluated the performance of the proposed approaches using simulation driven by real-world 3G bandwidth and vehicular mobility traces. We find that k-MDP yields a linear trade off between performance and overhead. Interestingly, although offline approaches have zero online computation overhead, they both outperform the online approach. The best performance is achieved with x-MDP.

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 [2], Microsoft [9], and Adobe [1] are already trialling their own proprietary HTTP-based video streaming platforms while a standard, called dynamic adaptive streaming over HTTP (DASH) [13], has been recently drafted 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 into a series of small video chunks of 2-4 sec durations. 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 the 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 only specifies how the video chunks should be stored and what metadata about the chunks should be provided to a client. 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.

A major challenge in developing client intelligence for DASH-based streaming comes from the so called *personal taste* for the different metrics of QoE. For example, a user may prefer to maximise the average quality of the watched chunks as long as video does not freeze, but is not very sensitive to the frequency of quality change from chunk to chunk as the video progresses. Similarly, another user may be very annoyed each time there is a change in quality, but is rather tolerant of a slightly lower but stable quality. Existence of such personal tastes calls for an intelligence that is highly adjustable in each of the QoE dimensions. Conventional video quality adaptation algorithms react to current buffer level at the client to decide whether to increase, decrease, or maintain the quality of the next chunk [7]. These algorithms perform reasonably well in maximizing the quality of the video, while some adjustments can be made for the other dimensions by tuning the buffer thresholds used by the algorithm. However, these algorithms do not yield a finer control of all QoE dimensions.

Recently, Markov Decision Process (MDP) has been proposed [8] as an effective framework to achieve arbitrary control in any dimension. MDP is an optimization framework that maximizes a revenue function with arbitrary weights assigned to quality, deadline miss, as well as quality change frequency. By adjusting these weights, a client can achieve the desired QoE of a given user. The main challenge with MDP-based adaptation is the high computation cost of the optimized solution, which may have to be frequently recomputed in mobile environments due to changes in bandwidth statistics.

In this paper, we propose and evaluate three approaches to reduce MDP overhead. We consider both on-line and offline MDP optimization. For online, we propose a k-chunk update approach (k-MDP) which recomputes the optimal strategy after downloading every k chunks. The online approach uses the bandwidth statistics gathered only during a video session without referring to

any statistics collected offline. For offline, we propose two different approaches, single MDP strategy (s-MDP) and x-meter multiple MDP strategy (x-MDP). s-MDP uses the global statistics of a given region to compute an optimal MDP strategy which is used throughout the video session in any given trip in that region, while x-MDP recomputes the optimal strategy for every x meters of the travel using offline statistics for the last x-meter of the road.

We have evaluated the performance of the proposed approaches using simulation driven by real-world 3G bandwidth and vehicular mobility traces. We find that k-MDP is capable of reducing online computation overhead significantly without any noticeable degradation in video QoE. Interestingly, although offline approaches have zero online computation overhead, they both outperform the online approach. The best performance is achieved with x-MDP.

The rest of the paper is organised as follows. Section II shows how DASH can be formulated as an MDP problem. We present the proposed MDP overhead reduction approaches in Section III, followed by the simulation details in Section IV. Results are presented in Section V. We discuss related work in Section VI before concluding the paper in Section VII.

2 HTTP-based Adaptive Streaming using MDP

In this section, we elaborate on the Markov Decision Process (MDP) framework used to optimise HTTP-based adaptive streaming protocols under consideration in this paper. We follow the overall MDP framework proposed in [8] to define the systems states and actions, but derive our own formulations to compute the transition probabilities and parts of the revenue function. In the following, we explain the key parameters of this MDP framework central to the understanding of the proposed overhead reduction approaches.

System states and decision timings: We observe the system state when a video chunk is completely downloaded. The system state $S(\rho, q)$ 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.

Although ρ is a continuous number between zero and upper bound, we can 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 holding only 2-sec chunks, $n = 2$ would

give us $7 \times 2 \times 2 = 28$ different values for ρ.

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: 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 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 an upper bound of M with n discrete intervals per second for measuring the value of ρ, the transition probability from state (i, x) to state (j, y) can be obtained using the following equation, which for $1 \leq q \leq N$, yields a 3D matrix of size $\{(M \times T \times n + 1) \times N\} \times \{(M \times T \times n + 1) \times N\} \times N$:

$$\mathrm{P}^{q}_{(i,x)(j,y)} = \begin{cases} 0 & 1 \leq x \leq N, y \neq q, 0 \leq i \leq M \times T \times n, 0 \leq j \leq M \times T \times n \\ P_{ij}^{q} & 1 \leq x \leq N, y = q, 0 \leq i \leq M \times T \times n, 0 \leq j \leq M \times T \times n \end{cases}$$

where P_{ij}^{q} is calculated as:

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

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

$$P^q(x) = \begin{cases} 1 - F(n \times S(q)) & \text{if } x = 1 \\ F\left(\frac{n \times S(q)}{(x-1)}\right) - F\left(\frac{n \times S(q)}{(x)}\right) & \text{if } x > 1 \end{cases} \tag{2.1}$$

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 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 (i, x):

$$R^q(i, x) = u(q) - d(i, q) - c(x, q)$$

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. The deadline penalty can be derived as a function of the probability that the next chunk will miss its deadline:

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

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. Finally, $c(x, q)$ is a penalty for a specific change of quality level and we can assign different penalties for difference types of quality switches. For example, the penalty for switching to a lower quality can be larger than that for switching to a higher quality to encourage higher average quality for a video session. Similarly, penalty for jumping multiple quality levels can be harsher than a smoother change in quality.

Once the rewards and penalties are assigned, we basically need to solve the optimisation problem that maximises the revenue function. Value iteration [11] is a well known algorithm for solving such optimisations and we used the MATLAB implementation of this algorithm [10] to solve all our MDP formulations in this paper. Once solved, the outcome of the optimisation is an optimal action for each given state. This set of actions is called the optimal policy or strategy, which is essentially a 2-column table. Given an MDP strategy, an HTTP-streaming client can simply make the decision about the quality of the next chunk by first observing its current state, i.e., the quality of the last downloaded chunk and the value of ρ, and then looking up a strategy table.

While MDP provides an effective framework to tune and balance different dimensions of adaptive HTTP streaming performance by adjusting the rewards and penalty parameters, it requires the client to solve the MDP problem first before the strategy table can be used to make the decisions. However, as we have seen in the preceding equations, the optimisation depends on the CDF of the underlying bandwidth. Depending on how the client obtains this CDF, the MDP may lead to high computation overhead for a mobile client. For example, if the client only uses its on-line observations to learn the CDF, then it may want to compute the MDP every time there is a new bandwidth observation to ensure that most accurate CDF is used in the optimisation. In the case of HTTP streaming, this means one MDP optimisation for every chunk download, as we obtain a new bandwidth observation when a new chunk is downloaded. However, this would lead to very high overhead. It is therefore important to consider implementation options that would reduce MDP overhead without significantly deteriorating its performance, which is the topic of the next section.

3 MDP Strategy Update Approaches

In this section, we propose three MDP strategy update approaches that reduce MDP computation overhead in different ways. In all of these approaches, we assume that bandwidth has a normal distribution, hence we attempt to estimate the mean (μ) and standard deviation (σ) of the bandwidth, which can be readily used to obtain the normal CDF. Three approaches differ in ways they attempt to estimate μ and σ. The first of these is called k-MDP, which is designed to reduce MDP overhead when the client uses only the online observations to estimate bandwidth CDF. The client uses each bandwidth observation sample obtained after downloading every chunk to update the mean (μ) and standard deviation (σ) of the underlying bandwidth, but recomputes MDP strategy only after downloading k chunks. This approach reduces MDP overhead by a factor of k compared to the basic approach where MDP is computed each time a chunk is downloaded.

Next, we consider two approaches that completely eliminate the online MDP optimisation overhead by solving the MDP offline using bandwidth samples that are collected in previous trips over the same roads. We propose two different offline approaches, the single MDP(s-MDP) and x-meter MDP (x-MDP). s-MDP uses the global bandwidth statistics for a given region to generate an MDP policy which is used throughout a given trip taken in that region. With this approach, the client incurs no online overhead and there is only one offline MDP calculation.

Like s-MDP, x-MDP also incurs zero online overhead, but it recomputes a different MDP strategy each time it travels x meters. To compute a MDP strategy, x-MDP uses the bandwidth CDF specific to a given road segment of x meter long. This approach is motivated by the previous findings that distribution of mobile network bandwidth may change significantly from one road segment to another even within the same region [5, 14]. We compare the performance of these three approaches using simulation experiments driven by real bandwidth and mobility traces.

4 Simulations

In this section, we outline the simulation setup that we have used to evaluate the three approaches outlined in Section III. We first provide an overview of the empirical bandwidth traces that we have used in the evaluations. Next, we briefly discuss the parameters of the video and the MDP model.

4.1 Empirical bandwidth traces

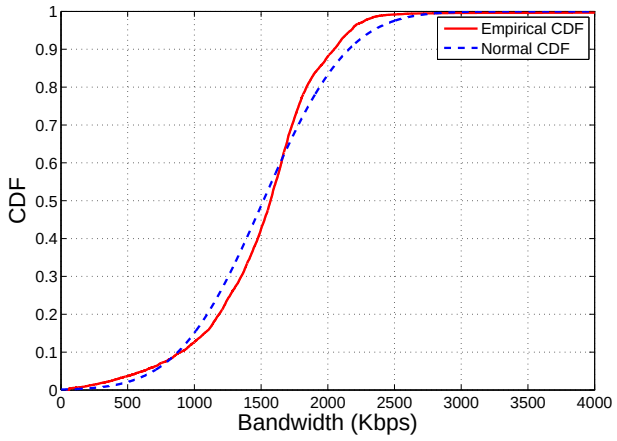
In our evaluations, we use real-world mobile bandwidth traces collected empirically by Yao et al. [14] . The researchers measured the downlink mobile bandwidth at approximately every 10s while driving along a route in the city of Sydney. The route is 24 Km long and typical drive time ranges from 22 to 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 4.1 illustrates an example of 6 data points. Column one represents the time when samples are recorded, column two and three are the geographical coordinates and last column is the measured downlink bandwidth. 70 repeated trips were made along this route, resulting in a total of 70 bandwidth traces. We use the first 64 trips to generate the bandwidth statistics for the offline MDP approaches while the final 6 trips (65-70) are used for our evaluations.

Table 4.1: Illustrative Example of Bandwidth Traces

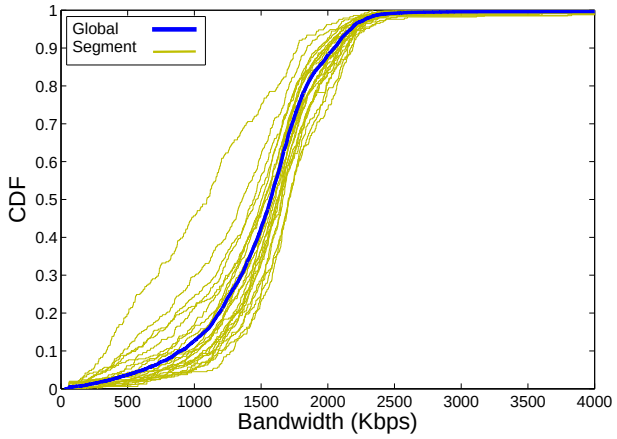
	time	latitude	longitude	bandwidth (Kbps)
1	1186549400	-33.919785	151.228913	1663.1440
2	1186549410	-33.919635	151.227787	1964.7330
3	1186549420	-33.91958	151.227322	2038.8659
4	1186549430	-33.91958	151.227322	2011.2631
5	1186549440	-33.91953	151.22692	1838.6578
6	1186549450	-33.91905	151.226322	1208.2767

Recall that the single MDP (s-MDP) scheme uses global bandwidth statistics from a given region. We assume that the entire trip is encompassed in a single region. Thus, all bandwidth samples from the first 64 trips are used to generate the statistics. We compute a single μ and a single σ across these samples leading to a single normal CDF and then use this CDF to compute the transition probabilities and generate a single MDP strategy for the entire route.

Unlike s-MDP, x-MDP recomputes the optimal strategy for every x meters of the road. We consider an $x = 1000$, which yields good number of samples for each road segments. Segment-wise statistics for the first 64 trips are shown in Table 4.2. As we can see, with 1000-meter segments, we have 24 different μ's and σ's, which provides 24 different normal CDF. Figure 4.1 (a) shows that normal CDF provides a good approximation for the empirical CDF of the bandwidth, while (b) shows that there are differences between CDFs of individual road segments.



(a)



(b)

Figure 4.1: CDF of bandwidth traces (a) empirical vs. normal CDF, and (b) individual (normal) CDFs of 24 road segments

4.2 Video Statistics

In our evaluations, we used the Big Buck Bunny movie [3]. The original clip is 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 [6] to encode the original video file using bit-rates of 186 kbps (*quality 1*), 499 kbps (*quality 2*), 1101 kbps (*quality 3*), 1292 kbps (*quality 4*) and 1898 kbps (*quality 5*). Figure 4.2 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 4.3, the variance is small, so we used the average chunk size in our simulations instead of exact sizes for each individual chunk (Equation 1 shows how chunk size is

Table 4.2: Bandwidth statistics for 1000m road segments

Road segment	# of samples	μ	σ
1	825	442.07	249.05
2	1022	478.57	368.20
3	1206	423.60	92.66
...	...	...	...
23	758	438.87	126.37
24	79	444.14	94.08
Whole-Route	12716	438.02	251.61

used in calculating transition probability).

Figure 4.2: Quality levels

4.3 MDP parameters

As explained in the previous sections, MDP is a flexible optimisation framework whose outcome can be influenced in any of the three dimensions of streaming performance by adjusting its reward and penalty parameters. We keep the reward parameters fixed as shown in Table 4.4, but vary the deadline (D) and

Table 4.3: 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 change (C) penalties. The value of D is varied between 2 and 350 and C is varied as a factor of the base values shown in Table 4.5 (we considered 10 different factors between 0.1 to 1.9). The values of other MDP parameters are as: N= 5, M= 7 and n= 2.

Table 4.4: Reward function

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

Table 4.5: 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

5 Results

In this section, we analyse simulation results to evaluate and compare the performance of the three proposed MDP overhead reduction approaches, k-MDP, s-MDP, and x-MDP. Before the comparison, we first examine the personalisation feature of MDP in general.

5.1 Personalization with MDP

Table 5.1 shows the streaming performance for the online MDP with k=1 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 the five test trips.

The personalization opportunity with MDP is clearly demonstrated by Table 5.1. For example, by setting a high value for D, say D=150 (last row), one can keep the deadline miss to a minimum (only about 4 deadline miss for the entire

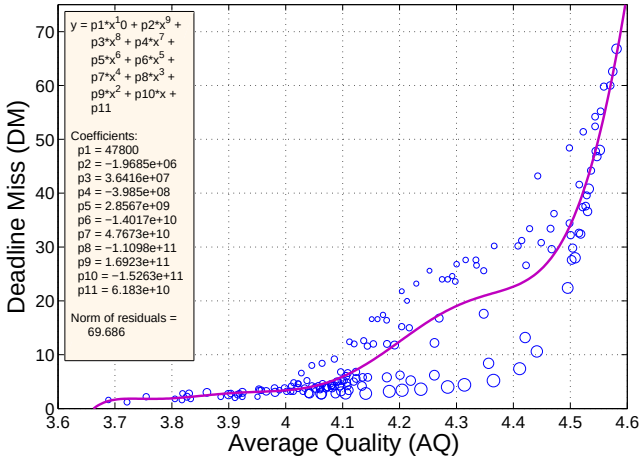


Figure 5.1: Scatter plot of AQ and DM data. The size of the circles is adjusted to reflect the different values of QC with larger circles denoting larger QC, and vice versa. For some AQ values, we have a wide range of DM, where larger DM correspond to smaller QC, as seen in the middle of the graph between 4 and 4.5 in the x-axis.

trip). For the minimal deadline miss with D=150, one can either set a small value for C (say 0.1) to watch a high quality video (average quality level of 4.28), but with large number of quality changes (107), or the number of quality changes could be reduced significantly (to 23.8) for a slightly lower AQ of 4.02.

Figure 5.1 shows the flexibility of MDP using a scatter plot of all data obtained from many different combinations of D and C values. We can see that not only we have a wide range of options between the AQ (x-axis) and the DM (y-axis), we have the opportunity to trade-off between DM and QC when considering a specific AQ. See for 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 to avoid a potential deadline miss. We further observe that MDP yields a non-linear trade-off between picture quality and deadline miss with number of deadline miss increases rapidly if we try to watch the video in very high quality. We were able to fit this non-linear trade-off to a tenth degree polynomial.

5.2 Online MDP

In this section, we evaluate the performance of k-MDP in terms of online computational overhead and streaming performance. Intuitively, with an increasing k, the proposed k-MDP would reduce online computation overhead linearly because the value of k directly controls the number of times the MDP optimization will have to be solved. To verify this, we have recorded the total simulation time for two different test trips for different values of k (see Table 5.2). We can see that the simulation time reduces as we increase k and the reduction is roughly proportional to k.

A key feature of k-MDP is the trade-off between computational overhead and streaming performance. To characterize this trade-off, we now turn to

Table 5.1: 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.

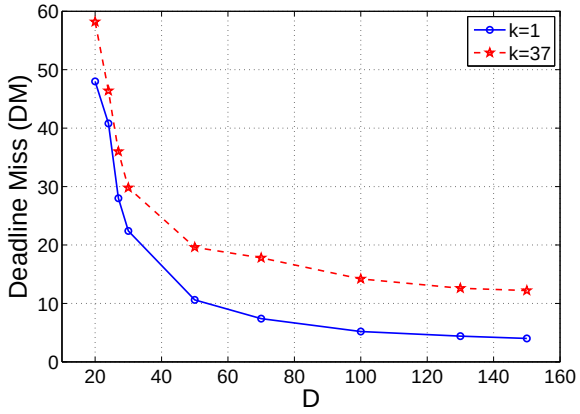
D\C	0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
	195.4	202.8	207.0	230.4	268.8	277.4	280.8	282.6	282.8	286.4
10	4.74	4.74	4.75	4.77	4.80	4.81	4.81	4.82	4.81	4.81
	47.60	34.00	26.40	20.80	14.60	13.80	13.00	12.60	12.00	10.80
	66.80	62.60	60.00	59.80	55.20	52.40	54.20	51.40	48.40	43.20
15	4.58	4.57	4.57	4.56	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
...	...	...	...	...	...	...	...	...	...	...

Table 5.2: Length of simulation (in seconds) of k-MDP for different values of k. Simulation was run on a laptop with an i5-3320M-2.60GHz CPU and 8GB RAM

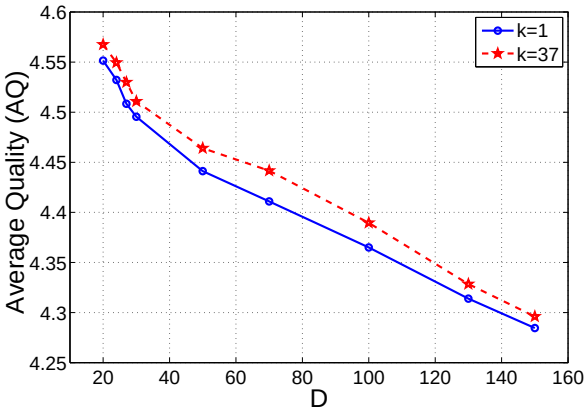
		k=1	k=10	k=20
Trip	65	502.79	53.74	26.80
	66	343.81	36.33	19.66

investigate the effect of an increased k on the streaming performance. Figure 5.2 plots the effect of k on all three dimensions by varying the deadline miss penalty. We find that the effect of k is most pronounced on deadline miss. In the subsequent analysis, we therefore focus on this dimension to find out how deadline miss is affected as a function of k.

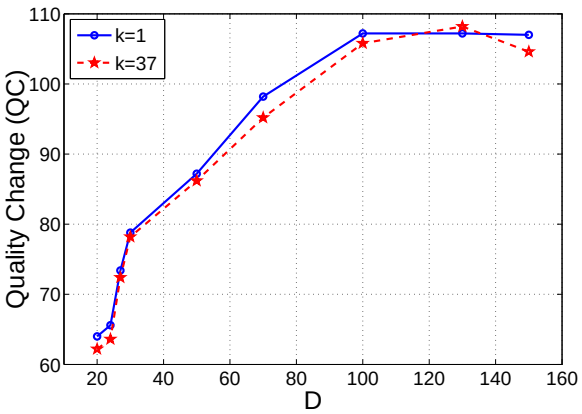
Table 5.3 shows the streaming performance for different values of k for D=130 averaged over 10 different values of C in the interval [0.1, 1.9] with increments of 0.2. As we can see, k has a negligible effect on AQ and QC, but number of deadline miss continues to increase as we increase k. Figure 5.3 shows that deadline miss increases linearly with k. We have about one extra deadline miss for every 10 jumps in k. Note that each deadline miss represents a video freezing event during the session. Given that k has linear effect on both computation overhead as well as streaming performance (deadline miss), we can conclude that k-MDP yields a linear trade-off between performance and overhead.



(a)



(b)



(c)

Figure 5.2: Impact of deadline miss penalty on three dimensions of quality for two different values of k. $k = 37$ causes noticeable increase in DM compared to $k = 1$, but no noticeable difference in AQ and QC.

Table 5.3: Streaming performance of k-MDP for different values of k

	k=1	k=5	k=10	k=20	k=37	k=50
DM	15.54	16.54	17.18	19.4	20.9	22.84
AQ	4.256	4.263	4.262	4.266	4.269	4.268
QC	38.56	37.76	36.28	36.62	36.66	36.64

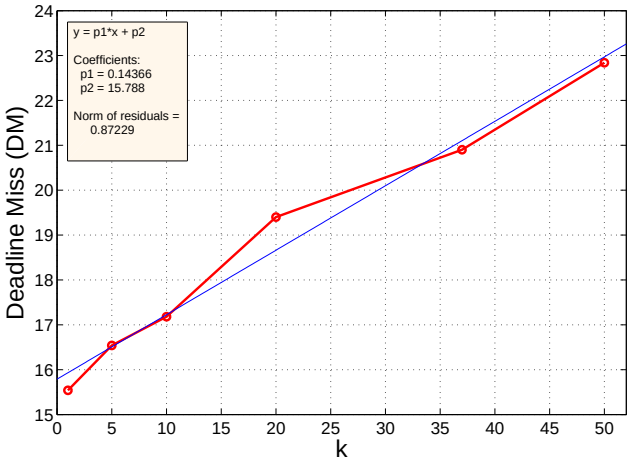


Figure 5.3: Deadline miss increases linearly with k. There is roughly one extra deadline miss for every 10 jumps in k.

5.3 Offline MDP

Note that off-line MDP approaches, s-MDP and x-MDP, have zero on-line overhead as all optimisations are computed off-line. We are interested in comparing their performances with those of k-MDP to see how the elimination of on-line overhead impacts performance. Since we found that MDP provides a non-linear trade-off between AQ and DM (Figure 5.1), it is important that we compare off-line approaches with on-line ones using this trade off curve.

For a given MDP approach, we derive the AQ-DM trade off curve by first generating a large number of performance data for many different combinations of deadline miss penalty and quality change penalty similar to the combinations shown in Table 5.1. We have run a total of 150 simulations representing 150 combinations, where C assumed 10 different values in the interval [0.1, 1.9] with increments of 0.2, and D assumed 15 different values in the interval [2,350]. For each value of D, we obtain a single performance value by averaging all performance data from 10 different values of C. This gives us 15 trade off data points for each approach as plotted in Figure 5.4. For k-MDP, we consider a value of k=37, as we find that on average about 37 chunks are downloaded in each of the 1000-meter road segment. This aligns k-MDP approach with x-MDP, which also switches to a new MDP strategy after every 1000 meters of travel.

We make several observations. First, we find that like on-line, off-line MDP also exhibits similar non-linear trade-off between AQ and DM. Second, we find that although on-line overhead is totally eliminated in the off-line approaches,

they actually perform better than on-line approach. x-MDP performs the best, which could be attributed to its ability to make use of more precise estimation of the CDF of mobile bandwidth.

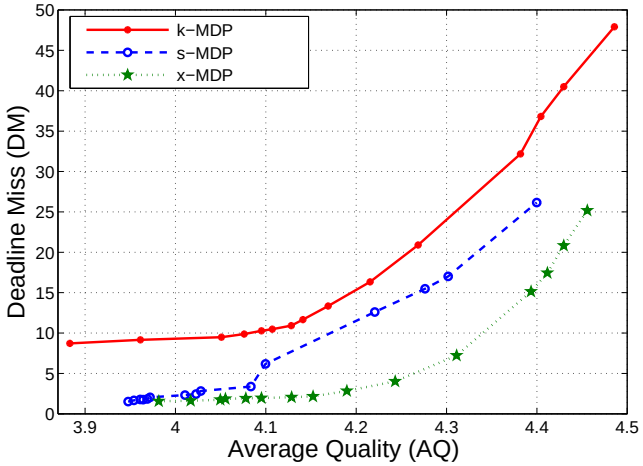


Figure 5.4: Comparison between online MDP (k-MDP) and offline MDP (s-MDP and x-MDP).

6 Related work

Jarnikov et al. [8] considered MDP to optimize DASH video streams, but they have not considered approaches to reduce MDP computation overheads for a mobile client. Halvorsen et al. [12], Curcio et al [4], and Yao et al. [15] have considered location-based bandwidth statistics to improve video streaming performance, but they have not considered the MDP optimization framework. Yao et al. [14] and Despandhe et al. [5] have collected bandwidth traces from 3G networks while driving in a car and using an entropy-based method have shown that location-based statistics contain more information about bandwidth compared to global statistics. Authors of [12] have implemented a platform that allows streaming clients to access bandwidth statistics history of a given location, giving evidence that location-based streaming optimization is technically viable.

7 Conclusions

We have proposed and evaluated three approaches to reduce optimisation overhead for HTTP-based adaptive streaming when MDP is used as the underlying optimisation framework. For on-line optimisation, we have shown that updating the MDP strategy after downloading every k chunks yields a linear trade off between performance and overhead. Conceptually, on-line overhead could be completely eliminated with off-line optimisation using only past observations of

mobile bandwidth for a given region or road segment. Our simulation experiments involving real bandwidth and mobility traces along with actual video contents have revealed that such pure off-line MDP optimisations outperform the on-line optimisation in terms of improved trade off for picture quality and deadline miss. The best performance is achieved when the MDP strategy is optimised using the bandwidth CDF of each specific road segment, which allows the system to realign with any statistical differences between different locations of a given region.

Acknowledgement

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

Bibliography

[1] 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.

[2] Apple. HTTP Live Streaming Overview. Also available as `https://developer.apple.com/library/ios/#documentation/networkinginternet/conceptual/streamingmediaguide/Introduction/Introduction.html`.

[3] BlenderFoundation. Big buck bunny. [Online accessed 04-March-2013], URL: http://www.bigbuckbunny.org.

[4] Igor D.D. Curcio, Vinod Kumar Malamal Vadakital, and Miska M. Hannuksela. Geo-predictive real-time media delivery in mobile environment. In *Proceedings of the 3rd workshop on Mobile video delivery*, MoViD '10, pages 3–8, New York, NY, USA, 2010. ACM.

[5] Pralhad Deshpande, Xiaoxiao Hou, and Samir R. Das. Performance comparison of 3g and metro-scale wifi for vehicular network access. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 301–307, New York, NY, USA, 2010. ACM.

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

[7] Youn-Sik Hong, Ji-Hong Kim, and Yong-Hyun Kim. A buffer-controlled adaptive video streaming for mobile devices. In *Convergence Information Technology, 2007. International Conference on*, pages 30–35, 2007.

[8] Dmitri Jarnikov and Tanr zelebi. Client intelligence for adaptive streaming solutions. *Signal Processing: Image Communication*, 26(7):378 – 389, 2011. ¡ce:title¿Advances in IPTV Technologies¡/ce:title¿.

[9] A. Majumda, D.G. Sachs, I.V. Kozintsev, K. Ramchandran, and M.M. Yeung. Multicast and unicast real-time video streaming over wireless lans. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(6):524–534, 2002.

[10] Mathworks. Marcove decision process toolbox. Available as `http://www.mathworks.com.au/matlabcentral/fileexchange/25786-markov-decision-processes-mdp-toolbox`.

[11] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*, volume 414. Wiley-Interscience, 2009.

[12] Haakon Riiser, Tore Endestad, Paul Vigmostad, Carsten Griwodz, and Pâl Halvorsen. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM Trans. Multimedia Comput. Commun. Appl.*, 8(3):24:1–24:19, August 2012.

[13] Thomas Stockhammer. Dynamic adaptive streaming over http –: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, MMSys '11, pages 133–144, New York, NY, USA, 2011. ACM.

[14] Jun Yao, Salil S. Kanhere, and Mahbub 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*, WiNTECH '08, pages 11–18, New York, NY, USA, 2008. ACM.

[15] Jun Yao, S.S. Kanhere, and M. Hassan. Improving qos in high-speed mobility using bandwidth maps. *Mobile Computing, IEEE Transactions on*, 11(4):603 –617, april 2012.