Multimedia prefetching with optimal Markovian policies

Cezar Pleşca, Vincent Charvillat, Wei Tsang Ooi

Multimedia prefetching with optimal Markovian policies

Cezar Pleşca [a,*], Vincent Charvillat [b], Wei Tsang Ooi [c]

[a] Department of Computer Science, Military Technical Academy, 81–83 Bd. George Cosbuc, Bucharest, Romania
[b] Department of Computer Science and Applied Mathematics, IRIT-ENSEEIHT, 2 rue Camichel, 31071 Toulouse, France
[c] Department of Computer Science, School of Computing, National University of Singapore, Singapore 117417, Singapore

A B S T R A C T

Multimedia prefetching is able to reduce the end-to-end latency and improve the video quality perceived by users. Previous work modeled prefetching as applying sequential decisional policies under uncertainty, using a Markov Decision Process model that integrated both user behavior and resources availability, to achieve optimal prefetching policies under realistic assumptions. In this paper, we extended the existing MDP model by considering more complex and aggressive policies, while preserving the optimality of the prefetching policy. We further enriched the extended model by considering user's profile to provide finer prefetching policies. The proposed extensions and the associated policies are validated through comparison against the existing model and against heuristics found in related work. We showed that our profiles-aware optimal policies can be achieved up to 28% latency reduction with respect to known heuristics.

Keywords:
Multimedia
Prefetching
Markov Decision Process Profiling
Uncertainty Optimization

1. Introduction

Cisco (2015)predicted that 80% of all consumer Internet traffic (slightly more than one zettabyte) will be video traffic by 2019, about 4 times the video traffic in 2014. This trend is driven by popular services and Web sites, such as Netflix and YouTube, offering video contents ranging from home videos, news clips, and product advertisement, to professionally produced TV and movies. Users of these websites typically are presented with a set of video clips to view. The user selects one that she or he would like to view, causing the video to be streamed to the web browser. Once a sufficient amount of video data has been buffered, the video playback starts, while the rest of the video clip is still being downloaded. After watching a video, the user is typically presented with a selection of video clips that the system thinks the user may like (i.e., related list or video recommendations). The user might select one video from this set of video clips to watch, and the process repeats.

Recently, Krishnappa et al. (2013b) showed that viewers frequently choose videos from the related list and that the order of recommended video clips does influence the users' navigation patterns. In their work, these authors used this property and proposed to reorder recommended videos (by YouTube), prioritizing the ones that are already cached and ready to be delivered to the users. This strategy improved the viewing experience while jumping from one video clip to the subsequent one with reduced (resp. increased) latencies (resp. cache hit rates). Moreover, for a group of related videos, it is likely that a viewer would watch next video from the related list after viewing the previous one. Therefore, once a video is played back by a user, subsequent videos could be prefetched. Combining caching and prefetching is useful for popular applications such as online TV and video streaming services (Krishnappa et al., 2011a). As an example, Liang and colleagues recently introduced an integrated prefetching and caching proxy for services like Netflix and YouTube (Liang et al., 2015). These authors designed prefetching strategies that reduce latency between content servers and proxies in HTTP-based adaptive video streaming. In general, caching and prefetching are the two principal system techniques used to reduce latency. Both techniques have been extensively studied to reduce web pages access latency. The techniques complement each other: caching keeps a copy of downloaded objects in anticipation of repeated access, while prefetching downloads objects using any spare bandwidth in anticipation of future access.

In this work, we aimed at theoretically characterizing optimal prefetching policies. We considered users' navigation patterns within a video collection. While a user jumps from one video to another one, we tried to minimize the cumulative latency due to buffering of each video within an access sequence.

Accurately predicting which video will be accessed in the near future is a key to achieving good prefetching performance. A wrong prediction would lead to downloading videos chunks that

* Corresponding author.

http://dx.doi.org/10.1016/j.jnca.2016.05.002

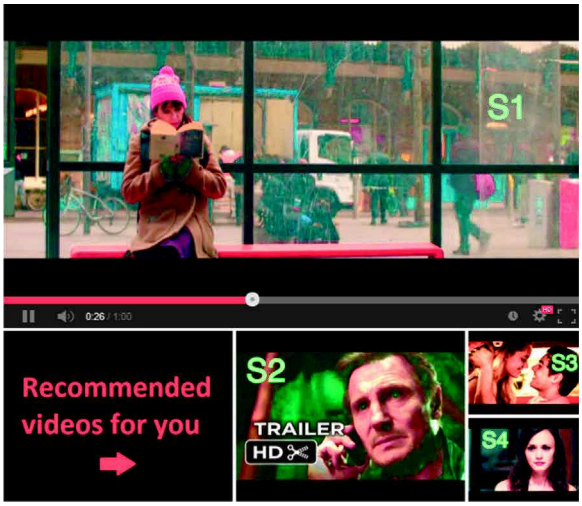


Fig. 1. Video content with related clips in the bottom-side of the video panel.

are not needed and does not reduce latency. Fortunately, predictions are often statistically possible due to the reproduction of users' navigation patterns. Frequent patterns might occur due to popular contents within specific communities of users. Frequent patterns might also be a direct consequence of an interface bias, such as the play lists on YouTube, for example. Fig. 1 illustrates another type of interface bias: even if the three related clips at the bottom are recommended, the left one is usually chosen more frequently because of its position and size.

Existing techniques for prefetching are often heuristically designed due to the complexity of the problem as discussed by Morad and Jean-Marie (2014a). Beyond the access pattern prediction, both the available resources (e.g., spare network bandwidth) and video clips sizes matter. Variability among users (e.g., profiles from different geographic regions, Krishnappa et al., 2013a) should also be understood by a smart prefetching agent. When several user profiles coexist, a simple prefetching policy can only fit with one mixed (e.g., averaged) profile, leading to sub-optimal results. We tackled this issue in this work. The core of this paper significantly extended our initial model by Charvillat and Grigoras (2007), already published in this journal. Our contributions in this paper are the following:

1. We proposed a general model that encompasses many existing techniques for prefetching. Our model allows computation of optimal Markovian prefetching policies based on complex access patterns and any network condition.
2. Our model inherently/automatically classifies the user access patterns into different profiles, allowing more accurate predictions and thus better prefetching performance.

We structured the rest of the paper into seven sections. Section 2 describes a general framework for prefetching, and presents several prefetching policies. Section 3 recapitulates our initial model for deciding the optimal prefetching policy using an approach based on Markov Decision Process. Section 4 extends our model by considering multiple stream prefetching policies while Section 5 considers multiple user profiles. We evaluate our proposed prefetching method in Section 6. Section 7 contrasts our work with existing literature about prefetching. Finally, Section 8 concludes the paper by summarizing the key messages of this paper and reflecting on possible future directions of this research.

2. Prefetching model

In this section, we describe, informally, our general model for prefetching video clips. In our model, the user is presented with a set of recommended video clips to watch, either while watching a video (see the screen shot in Fig. 1), or after watching a video. Prefetching decision is made after a prefetching operation completes, or when the user selects a new video to watch.

For each clip prefetched, we always prefetch only the initial segment of the video, corresponding to the buffering time required to playback the video. In other words, a video whose initial segment has been prefetched completely can start playing immediately if the user selects this video to watch later. For brevity, when we say that a *video has been prefetched*, we mean that the *initial segment* of the video has been prefetched.

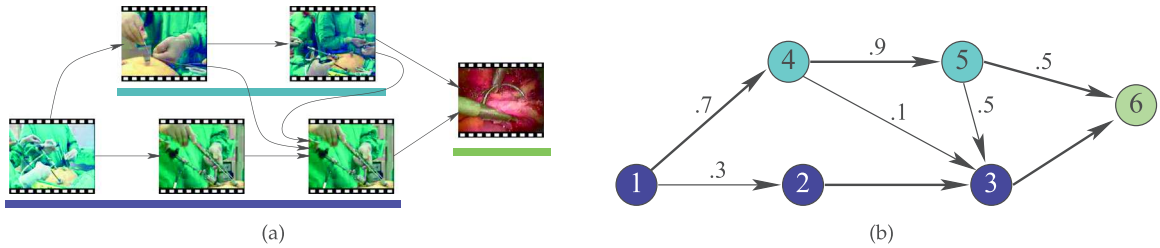
It may happen that the initial segment of a video is only partially prefetched. This situation occurs when the user selects a new video to watch, while the other videos are still being prefetched. In this case, a latency is incurred if the user decides to watch the partially prefetched video later. The user would have to wait for the initial segment of the video to finish buffering before playback starts.

A good prefetching policy should reduce the expected latency experienced by users while they watch the video clips from a given video collection. We use the expected latency as the performance objective. Another commonly used objective, the hit ratio (how many objects prefetched has been accessed) is not really suitable, as it does not consider partially prefetched objects.

A prefetching action should also consider how much of the spare bandwidth to allocate to prefetch each video. The prefetching decision depends on several factors. The first factor is the user behavior. The prefetching agent should prefetch videos that are likely to be accessed by the users after watching the current video. For instance, on YouTube, a long video is commonly split into shorter video clips. In this case, users are likely to watch the video in their original sequence. The second factor is the amount of video that we can prefetch within a given time. This factor in turn depends on the total spare bandwidth, the download rate, and the playback rate of the video. The third factor is how much of the initial segment of a video has been partially prefetched before.

The situation illustrated by Fig. 1 naturally leads to several intuitive prefetching heuristics. If the video stream S_2 is most likely to be viewed by the user after S_1, the associated transition probability $p(S_2|S_1)$ should be higher than $p(S_3|S_1)$ or $p(S_4|S_1)$. An intuitive idea is then to prefetch either S_2 only, or S_2, S_3 and S_4 proportionally according to their transition probabilities. Beyond this, several other policies can be devised. To allow the explanation of these heuristic policies, we first model the user behavior as a weighted directed graph called the *navigation graph* (or execution digraph as named by Fomin et al., 2014), where each vertex is a video and an edge going from video S_i to video S_j with weight $p(S_j|S_i)$. Fig. 2 shows an example of the navigation graph.

Let S_{cur} be the video currently being watched by the user. Every time the user selects a video S_{cur} to watch, the entire available bandwidth is allocated to download the initial segment of S_{cur}, to reduce the buffering time. Any on-going prefetching will stop. When the initial segment of S_{cur} is downloaded and the playback of S_{cur} starts, S_{cur} will be downloaded with data rate equivalent to the video playback rate. If there are spare bandwidth on the network, then prefetching of other video streams starts in the background, according to some policy. We will further introduce some well known heuristics policies.



(a) (b)

Fig. 2. (a) Navigating a video collection. (b) A video navigation graph.

The BEST-FIRST prefetching policy always allocates all the spare bandwidth to prefetching one video stream with the highest probability of being accessed by the user from S_{cur}, i.e., the neighbor of S_{cur} with the highest weight in the navigation graph. When this video has been prefetched, the policy does not prefetch another video, even if the current video is still playing and spare bandwidth is available.

The proportional policy (denoted PROP) allocates the spare bandwidth, proportionally, to all video streams immediately accessible from S_{cur}, according to their access probability and independent of their bit rate. In this way, the amount downloaded for each video stream is proportional to its access probability. In other words, all neighbors of S_{cur} in the navigation graph will be prefetched in parallel. Upon completion of prefetching a video, its allocated bandwidth is proportionally redistributed to the other videos being prefetched. When all neighbors of S_{cur} have been prefetched, the policy does not prefetch other videos, even if the current video is still playing and spare bandwidth is available.

The two heuristics policies described before, BEST-FIRST and PROP, do not fully utilize the spare bandwidth, as they only make the prefetching decision once each time a video is playback. The next group of policy, collectively called the sequential policies, continually prefetch video while S_{cur} is being playback. The MAINLINE policy essentially searches for a path of limited length l on the navigation graph starting from S_{cur} with the highest probability of accessed, and prefetch the videos on that path one after another with full spare bandwidth allocated to each video prefetched. MAINLINE reduces to the case of BEST-FIRST if $l = 1$.

One can view BEST-FIRST and PROP as shortsighted policies, as they only prefetch immediately reachable videos (Tuah et al., 1998). On the contrary, MAINLINE prefetches objects possibly accessed further into the future (Tuah et al., 1998). PROP and MAINLINE are also more aggressive, because they attempt to prefetch more objects, at the price of greater bandwidth consumption.

The three policies described earlier rely solely on the user navigational behavior to make their prefetching decisions. Not all navigational behavior, however, are captured in the navigation graph. The navigation graph does not capture the notion of time. A user may select another video to watch before the current video completes its playback. In Fig. 1, the user might click and jump to S_2 from the middle of S_1. This action would interrupt the decided prefetching of S_2 and may be S_3, S_4 and thus some videos might not be completely prefetched depending on their bit rates and allocated spare bandwidths.

Such partially prefetched video affects the subsequent optimal decisions since some stalled prefetching actions should or should not be continued depending on user behavior. Partial prefetching can also occur due to variability in network bandwidth. The navigation graph also does not capture this source of uncertainty. In the next section, we present a much more general model for capturing these different factors and show how this general model can be used to compute an optimal prefetching policy that minimizes the expected latency for all users.

3. Prefetching as a Markov Decision Process

We now present a model that enriches the user navigation graph to include viewing length, network conditions, and past prefetching actions. The key to our model is the inclusion of buffer fill levels into the states of the navigation graph. The buffer fill level of each video succinctly summarizes how long a video has been watched (or equivalently, how long the prefetching operations are executed), the network bandwidth (how much data are prefetched) and what has been prefetched before.

Our previous work (Charvillat and Grigoras, 2007) called the new states that encapsulate both the buffer fill level of each video and the video currently watched as *buffer states*. For the sake of clarity, we briefly recapitulate this concept that has also recently been reused by Morad and Jean-Marie (2014a,b). A buffer state σ is denoted as $(s, b_1, b_2, \ldots, b_n)$, where s is the video stream currently being watched, and b_i is the current *buffer fill ratio* for video stream i. The buffer fill ratio $(0 \leq b_i \leq 1)$ is the percentage of the minimal amount of data needed to begin playing video stream i that has already been buffered.

With the introduction of buffer states, the access probability alone is no longer appropriate to determine the best video to prefetch. How much of each video has been prefetched before needs to be considered in the next prefetching decision. We combine both metrics by using the expected reduction in access latency as a metric to measure the reward if a given video or set of videos are prefetched.

We compute the reduction of latency at the entrance of the buffer state $\sigma = (s, b_1, \ldots, b_n)$. Upon entry into a buffer state, the video player first downloads the minimum quantity of data needed to begin playing s (a quantity denoted B_s). If bw is the current average bandwidth and b_s is the buffer fill ratio for video stream s, then prefetching have caused the latency to access s, to reduce by $B_s b_s$/bw. The video player, upon entering a state, also decides whether, while playing the current state, it will prefetch other video streams.

In our initial model by Charvillat and Grigoras (2007), we only consider *elementary* and disjoint prefetching actions: we can only prefetch (part of) the initial segment for a video stream i, as opposed to the heuristics policies seen earlier: sequential and proportional. In this work, we proposed and evaluated optimal prefetching policies for these two new classes that deal with multiple streams at once. Moreover, we also integrate a new hidden variable into our models, namely the user profile.

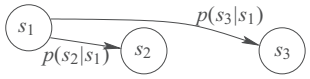
To summarize, prefetching can be seen as a *sequential decision problem* under *uncertainty*. Indeed, prefetching actions have to be taken sequentially, in each buffer state. The model's dynamics is expressed as transition probabilities between buffer states. Stochastic network conditions and random user navigation lead to variable efficiency (rewards). Therefore, we can model our prefetching problem as a Markov Decision Process (MDP).

An MDP is a stochastic controlled process that assigns rewards to transitions between states (Puterman, 1994). It is defined as a

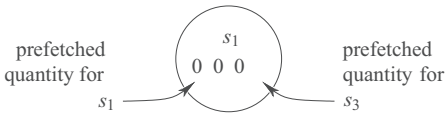
quintuple $(S; A; T; p; r)$ where S is the state space, A is the action space, T is the discrete temporal axis of instants when actions are taken, $p()$ are the probability distributions of the transitions between states[1] and $r()$ is a function of reward on the transitions. At each instant $t \in T$, the decisional agent observes its state $\sigma \in S$, applies on the system the action $a \in A$ that brings the system (randomly, according to $p(\sigma'|\sigma, a)$ to a new state σ'), and receives a reward $r(\sigma, a)$.

Our MDP model is as follows. S is the set of *buffer states*, where each buffer state encapsulates the current video stream and the fill rate of each video buffer. This fill rate depends on the previous loading decisions. We note a buffer state as $\sigma = (s, b_1, \ldots, b_n)$, where n is the number of streams and b_i the fill ratio of video i's buffer. A is the set of prefetching actions depending on the policies: each action represents a decision such as prefetch a video out of n videos or prefetch several videos simultaneously or sequentially.

T is $\mathbb{N}$ and represents the time instants (a.k.a., decision epochs) at which decisions are made. In order to introduce the dynamics $p()$, the rewards $r()$, and the relation between the decision epochs and the video duration, we start with an introductory example and a basic navigation graph. Let us consider a video stream s_1 of duration d_1 and bit-rate br_1, followed by s_2 and s_3 as follows:



Our first buffer state (denoted σ_1) includes s_1 as the video currently being watched and three empty buffers showing that no previous prefetching has occurred:

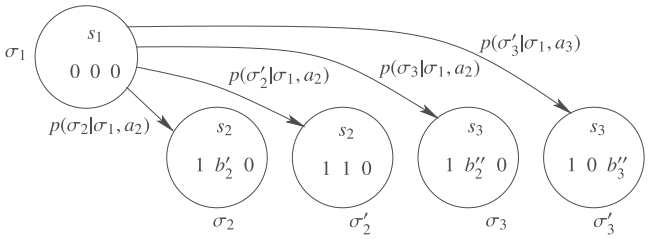


This first buffer state is visited at time $t = 0$ (first decision epoch). At the entrance of the buffer state, we observe the following events: (i) the system downloads the minimum quantity of data needed to begin playing s_1 (a quantity B_1). If bw is the average bandwidth, this causes a latency of B_1/bw. (ii) it decides whether, while playing the current state, it will also prefetch other video streams. We first consider *elementary* and disjoint prefetching actions: we can only prefetch (part of) B_i for one stream s_i. Let us assume that the model decides a_2 (resp. a_3) and s_2 (resp. s_3) is prefetched using the spare bandwidth while s_1 is viewed.

Playing s_1 lasts during a random duration $d \leq d_1$ depending on the user behavior (e.g., interest about s_1), and we consider here that she or he jumps to s_2 or s_3. This choice (e.g., a click on a recommended video) brings our model into a new buffer state associated with a new decision epoch $t = 1$, totally independent from the physical duration d. Possible transitions between buffer states encapsulate the effect of previous actions and transitions and two random sources: varying transition moments and varying bandwidth.

Below, four transitions among others are illustrated between σ_1 and other buffer states $(\sigma_2, \sigma'_2, \sigma_3, \sigma'_3)$. These target states differ: when a_2 has been decided, there is a probability $p(\sigma_2|\sigma_1, a_2)$ to reach σ_2, which represents a state where s_2 has been selected by the user, $B1$ has been downloaded ($b_1 = 1$ since s_1 has been played), and both the duration $d \leq d_1$ and the spare bandwidth led to prefetch a portion $b'_2 \times B_2$ of the initial segment for s_2. With another probability, $p(\sigma'_2|\sigma_1, a_2)$, the reached state σ'_2 shows a complete prefetching for s_2 due to user's navigation from s_1 to s_2 and better network conditions. We can compute the decrease in latency using the buffer level of b'_2 in σ_2 (resp. 1 in σ'_2) based on previous prefetching actions taken in σ_1. This latency reduction rewards good prefetching decisions previously taken.

The state σ_3 is similar to σ_2 except that the user made another choice and decided to watch s_3 after s_1 instead of s_2: this leads to an increased latency when the systems enter into σ_3 and download the minimum quantity B_3 of data needed to begin playing s_3 ($b_3 = 0$ in σ_3). The state σ'_3 represents a target state among many others at $t = 1$ when s_3 is accessed and have been partially prefetched:



This example illustrated the MDP dynamics. The probability of distribution of transitions between states is denoted by $p(\sigma'|\sigma, a)$, the probability of going to σ' when we are in σ and we chose action a at the start of σ. Let $\sigma' = (s', b'_1, \ldots, b'_n)$ and $\sigma = (s, b_1, \ldots, b_n)$, and a be the action of prefetching video i. The rewards $r_t(\sigma, \sigma')$ representing the *decrease in latency* as compared to a no-prefetching policy. The reward for transiting from a state σ to σ' at time t depends only of the target state and is calculated as follows:

$$r_t(\sigma, \sigma') = \begin{cases} 0 & \text{if } s = s' \\ b'_s & \text{if } s \neq s' \end{cases}$$

Solving such an MDP means finding an optimal prefetching policy that maximizes a criterion. Here, we optimize the expected discounted total reward given a starting buffer-state such as σ_1,

$$E\left[\sum_{t=1}^{\infty} \gamma^t r_t | \sigma_1\right]$$

where γ is a discount parameter to limit the influence of distant rewards. This objective amounts to maximizing the expected discounted total decrease in latency due to a prefetching policy.

In practice, we use the *Q-learning* (Watkins and Dayan, 1992) algorithm to determine, for each buffer-state σ and each prefetching action a, a *Q-value*, $Q(\sigma, a)$, that estimates, by simulation, the expected discounted total reward from this state. Since we have a finite number of states and actions, *Q-values* are proven to converge to their true values and the number of states and actions directly influences the convergence speed (Puterman, 1994). Given a buffer-state σ, it is then simple to select the best action a^* that maximizes the *Q-value*. This is how optimal elementary prefetching policies are characterized in Charvillat and Grigoras (2007). These policies can be considered elementary since only one object is prefetched at each decision epoch. The prefetching policies introduced in this paper are enriched while preserving the same optimality.

Recently our model from Charvillat and Grigoras (2007) has also been reused by Morad and Jean-Marie (2014a,b), where they formally consider non-elementary MDP policies and buffer states (i.e., proportional actions as defined in Section 2) but adopt less realistic assumptions: no randomness in the prefetching process, deterministic bandwidth, deterministic bandwidth allocations, users committed to view a selected video entirely, etc. In the related work section, we compare their work and alternative techniques including the new prefetching policies brought by this paper (Table 11).

[1] In practice, p can be obtained through long term observation of user access patterns at the server as well as buffer states reported by the video player.

In the following of the paper, we bring more sophisticated strategies that remain optimal under realistic conditions. The next section extends our model by considering optimal multiple stream prefetching policies while Section 5 considers multiple user profiles with optimality in mind.

4. Multiple stream prefetching policies

This section presents the first contribution of this paper. We build on the previous model and derive optimal policies for two new classes of prefetching strategies: proportional and sequential. The interest of these new policies is justified by a more efficient use of the available bandwidth. Moreover the playing time of a video is better exploited by prefetching one or more video buffers in a sequential or parallel manner.

4.1. Proportional policies

A proportional policy aims to share the spare bandwidth between one or more prefetching streams in order to download several buffers in parallel. More precisely, two questions need to be answered: Which video streams should be prefetched? How to allocate the spare bandwidth between the prefetching streams that have been chosen?

K-order proportional policy: Given an integer k, a k-order proportional policy is a strategy that associates, with every buffer state, an action composed of a set $\{s_1, s_2, \ldots, s_k\}$ of videos to be prefetched together with their respective percentages of the spare bandwidth $\{p_1, p_2, \ldots, p_k\}$. Naturally, we have $\sum_{i=1}^{k} p_i = 1$. For example, an action containing the streams $\{3, 4\}$ with their respective percentages $\{70\%, 30\%\}$ means that, while the current video is played, the agent will prefetch the streams 3 and 4 in parallel, allocating 70% and 30% respectively from the available spare bandwidth.

One can notice that a k-order proportional policy can be also seen as a $k+1$-order strategy. Indeed, it is sufficient to add a zero percentage component to a k-order policy to obtain the same decision strategy of $k+1$ order. Therefore, the set of high order policies includes the set of lower order strategies. Consequently, optimal policy inside high order strategies always performs better than an optimal policy selected from lower order strategies. As a result, even a 2-order optimal proportional policy should improve the performance of the simple optimal policies presented in Charvillat and Grigoras (2007).

Let us describe the formal MDP model associated with prefetching using k-order proportional policies. Among the elements (i.e., $\{S, A, T, p(), r()\}$) of the previous MDP model, only the action set A changes. In fact, an action a is no longer represented by a single video, but instead by a set of k video streams to be prefetched together with a spare bandwidth allocation. As for the buffer states, we quantify the percentages of the allocation with a granularity called AG to limit the number of possible actions. For a 2-order policy, by choosing $AG = 3$ and the set of objects $\{s_1, s_2\}$, there are four possible actions represented in Table 1.

One can notice that the first and the last actions illustrated in Table 1 represent obviously simple prefetching actions: prefetch s_2 and prefetch s_1. It should be noted that granularity value AG has a direct impact on the performances of the optimal proportional policy. High orders optimal policies provide better performances but their real implementation raises difficulties. Consequently, we used in our experiments only 2-order proportional policies.

Table 1
Prefetching actions for a 2-order proportional policy and a granularity $AG = 3$.

$\begin{pmatrix} s_1 & s_2 \\ \frac{0}{3} & \frac{3}{3} \end{pmatrix}$	$\begin{pmatrix} s_1 & s_2 \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix}$	$\begin{pmatrix} s_1 & s_2 \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix}$	$\begin{pmatrix} s_1 & s_2 \\ \frac{3}{3} & \frac{0}{3} \end{pmatrix}$

For a collection composed of n video streams, the theoretical number of actions is $n \times n \times (AG + 1)$. In practice, only half of this action is useful as each action appears twice as suggested in the example shown in Table 1. Moreover, all actions corresponding to a couple (s_i, s_i) are equivalent, representing the prefetch of a single object s_i.

4.2. Sequential policies

Compared to simple policies that interrupt the prefetching process as soon as the chosen initial video stream is completely loaded, sequential policies continue to prefetch other videos during the time spent by the user in the current video. This behavior utilizes bandwidth more efficiently and should offer higher latency reductions. The MDP model used for simple policies can be easily adapted to sequential policies. One way to do this is to consider subsequent videos to prefetch in the buffer state space. Thus, the buffer state BS can be enriched with addition information (called *start*) that codes the beginning of the current video as follows:

$$start = \begin{cases} 1 & \text{presentation start} \\ 0 & \text{presentation middle} \end{cases}$$

With this enhancement of the buffer state, the MDP model for sequential policies can be completely described. Thus, the set of decision epochs (T) is composed of video accesses (i.e., user clicks) as well as the endings of prefetching processes during the presentation of a current video. The set of actions (A) can be composed either of elementary decisions (simple policy actions) or of more complex decisions like those corresponding to proportional policies previously seen. In order to alleviate the presentation of the MDP model, the discussion is limited here to elementary prefetching decisions. Finally, the prefetching agent is rewarded only in video accesses (i.e., $start=1$) corresponding to the beginnings of video stream play-out.

The dynamics of the new MDP model is illustrated in Fig. 3. After the request (i.e., user click) of a video s, the prefetching agent enters in the enhanced buffer state BS ($start=1$) where it decides a prefetching action a. During the playback of the current video, the prefetching process continues until the amount Ba is completely loaded ($b_a = 1$). At this moment, the agent enters in the enhanced buffer state BS' ($start=0$) where it decides another prefetching action a'. This process goes on until the user clicks and jumps to another video stream s' or the expected sequence of all streams has been completely prefetched. When the user interrupts the prefetching process by a click, the value of *start* is reset to 1 (i.e., state BS'''). In practice, we considered only prefetching sequences whose depth are bound by a given length.

4.3. Experimental results

In our previous paper (Charvillat and Grigoras, 2007), we evaluated our simple optimal policies through several simulations. For the sake of comparison, we assess the quality of the newly introduced optimal policies in a similar way. We simply update the order of magnitude for both the video bit-rates and available bandwidth. We first used an example described in Fig. 4 and Table 2. This navigation path was used by Charvillat and Grigoras (2007). We include into our simulator several uncertainty sources with respect to access and resources models:

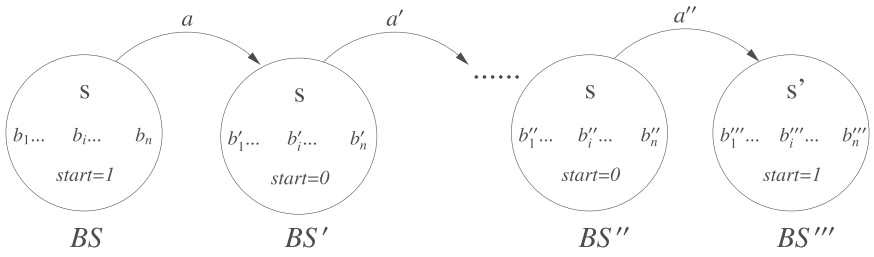


Fig. 3. MDP dynamics for sequential prefetching policies.

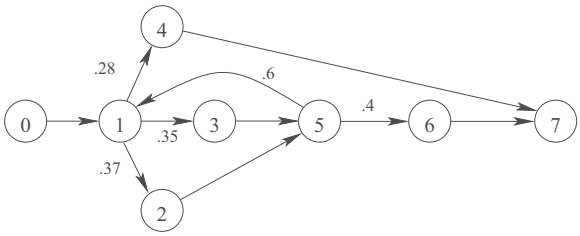


Fig. 4. Multimedia document navigation graph.

Table 3
Average latencies and gain percentages for 6 prefetching policies over 10,000 paths.

Policy type	Heuristics	Optimal policies	
Simple	*BEST-FIRST*	Latency	Gain
	28.994	18.262	**37**%
Proportional	*PROP*	Latency	Gain
	21.866	12.475	**43**%
Sequential	*MAINLINE*	Latency	Gain
	14.431	12.040	**17**%

Table 2
Characteristics of components of multimedia document.

Object (i)	0	1	2	3	4	5	6	7
Duration d_i (s)	60	240	60	60	240	360	60	60
Bitrate br_i (Kbps)	2400	1200	3600	3600	3600	3600	3600	3600
Buffer B_i (Mb)	12	8	20	20	80	80	20	20

- *Access model:* The path length varies uniformly between 4 and 10 clicks: the user is provided with multiple choices when visiting media objects 1 and 5. He or she may also come back to object 1 after having seen the media 5-cycles are therefore possible. The time spent by the user playing an object i follows a uniform distribution between 0 and d_i, where d_i stands for maximal duration of object i. We measure this duration from the moment the media object begins playing. This is because, while in the buffering stage, the user is not presented the content and thus a state change cannot occur.
- *Resources model:* In our experiments, the bandwidth dynamic respects a normal distribution $\mathcal{N}(m = 3500, \sigma = 250)$, therefore driving 95% of the values in 3–4 Mbps interval. Only the values between 3 and 4 Mbps were retained. Nevertheless, any other network model could be integrated.

The optimal prefetching policies inside each strategies class are obtained by *Q-learning* as done by Charvillat and Grigoras (2007). In other words, we solve the MDP prefetching problem with Q-learning by updating a value function $Q(\sigma, a)$ until convergence. These results are simply stored in a two-dimensional matrix of size $|S| \times |A|$ such as each couple $(\sigma \in S, a \in A)$ is mapped to an expected performance $Q(\sigma, a)$, obtained by applying action a in state σ and behaving optimally afterwards. This matrix is sparse, as the number of actually visited states is small and only meaningful actions are simulated for each buffer-state. We typically choose the total number of simulations as 1,000,000.

Validation: The new optimal policies are validated through comparison with the original model (i.e., simple policies) and some heuristic approaches. Recall that *MAINLINE* policy chooses objects that are placed on the most likely path starting from the current component. For example, while playing media object 1, *MAINLINE* tries to prefetch 2, 5, 6 and eventually 7. For *PROP* policy, the chosen objects are those immediately reachable (i.e., 2, 3 and 4) and those streams are prefetched in parallel, sharing the spare bandwidth according to their transition probabilities. Table 3 presents the results of 10,000 simulated random paths. For each of the six policies analyzed, we show the average latency; for the optimal policies we also present the gain percentage with respect to the corresponding heuristic.

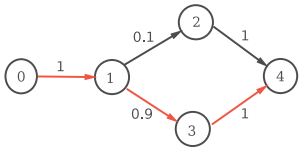
One can easily notice the incremental improvement from dummy policy (no prefetching) which takes in average 41.871 in, through heuristics approaches (i.e., *PROP* and *MAINLINE*) to the optimal sequential strategy, as computed by the Q-learning algorithm. Better performances can also be observed for proportional and sequential optimal policies by comparison with optimal simple policies.

Nevertheless, one should notice the performances of *MAINLINE* heuristic, tightly close to those of optimal sequential policy. In fact, its in-depth exploration principle and the available spare bandwidth allow it to prefetch heavy video streams (i.e., 4 and 5) earlier, especially during the presentation of second video (Video 1). Finally, another strong quality of the optimal policies is that, even if they did not anticipate the path that the user actually chose, they are able to provide the next optimum action at every step.

5. User profile-aware policies

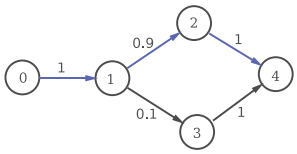
We have presented various models for obtaining optimal policies within some policy families (simple, proportional or sequential). These models are based on global access logs and therefore consider the user as an "average user". Indeed, transition probabilities in the navigation graphs as well as the playing time were used as average values with respect to all navigations.

Our models are able to learn how a community of users is browsing a predefined video collection. This learning process makes computation of different optimal policies possible. Although they aim at satisfying the average user, these policies are able to take into account the way the current user is using the content. In practice, however, users are non-homogeneous and form multiple sub-communities with different navigation patterns. The optimal policy computed after mixing navigations from



Object (i)	Avg. Time (s)	D_i (s)	Br$_i$ (Kbps)	Buffer B_i (Kb)
0	10	20	1200	3000
1	20	30	640	960
2	40	45	960	2400
3	40	45	960	2400
4	10	15	320	320

Fig. 5. Access characteristics for users from community P_1.



Object (i)	Avg. Time (s)	D_i (s)	Br$_i$ (Kbps)	Buffer B_i (Kb)
0	15	20	1200	3000
1	10	30	640	960
2	40	45	960	2400
3	40	45	960	2400
4	10	15	320	320

Fig. 6. Access characteristics for users from community P_2.

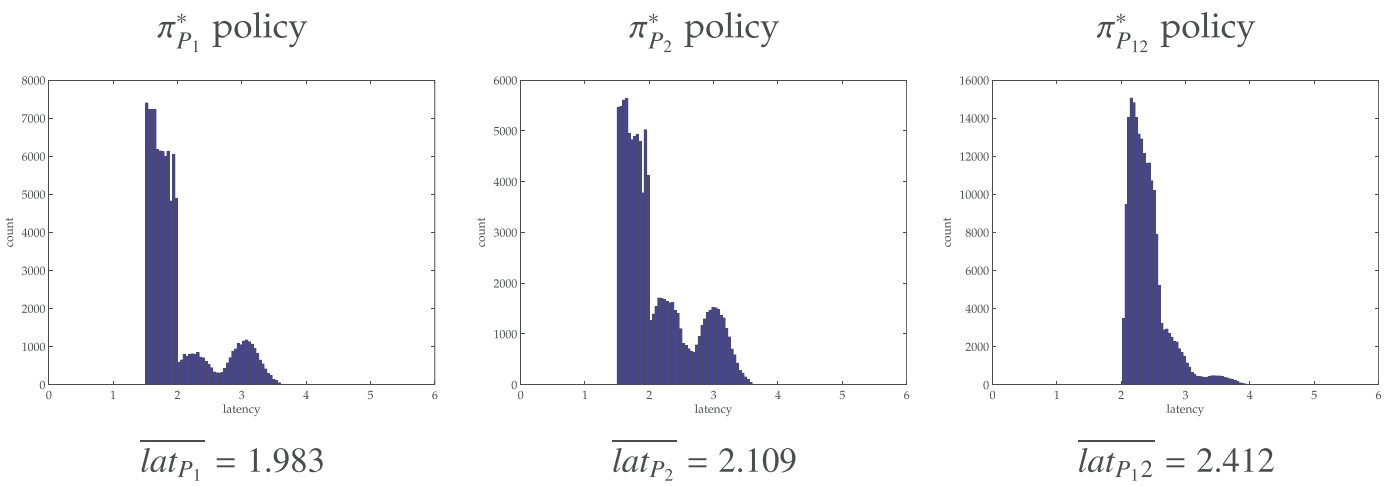


Fig. 7. Latencies' distributions for the three optimal policies applied to P_1, P_2 and P_1&P_2 communities.

two different sub-communities can perform less efficient than two optimal policies, adapted for each community individually.

5.1. Motivation

Let us consider for example two communities called P_1 and P_2 afterwards, whose access' characteristics are given in Figs. 5 and 6 respectively. One can notice that we have *the same pool of video streams* and only the *navigation patterns and average play times* for videos 0 and 1 *are different*. In these simulations, we consider a variable bandwidth from 1.5 to 2 Mbps (for both current and prefetching streams). Our *fair* prefetching agent only uses a maximum of 50% of the spare bandwidth as a resource for latency reduction.

For each community (P_1 and P_2), we obtained an optimal prefetching policy by applying *Q-learning* algorithm on 10,000 simulations. These two policies are noted $\pi^*_{P_1}$ and $\pi^*_{P_2}$. Considering the mixed community P_1&P_1 (the mix of P_1 and P_2, equally represented) we obtained another policy $\pi^*_{P_{12}}$. Fig. 7 shows the latencies' distributions by applying the three policies, namely $\pi^*_{P_1}$, $\pi^*_{P_2}$ and $\pi^*_{P_{12}}$ on P_1, P_2 and the resulting P_1&P_2 community respectively.

As one can observe from Fig. 6, the two profiles are quite similar, except for the time the user spent in video object 1. Thus, the users belonging to the profile P_1 had in average more time to prefetch the playout buffer of next optimal object compared to users from P_2 profile. Therefore, the P_1 users perceive a smaller average latency (i.e., 1.983 in) with respect to P_2 users (i.e., 2.109 in).

This introductory example clearly illustrates the shortcoming of our strategies when computed on users having different

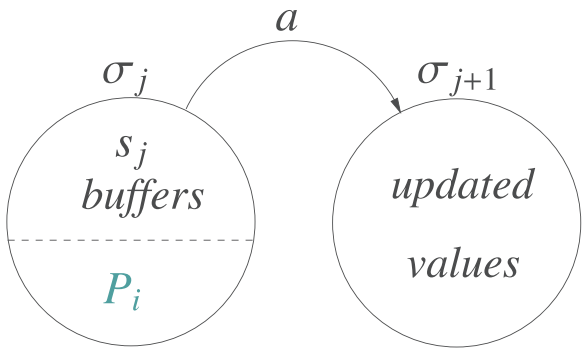


Fig. 8. Buffer-states model enriched with user profile.

navigation behaviors. The communities are in fact "averaged" but the obtained result is far from being optimal in this case. Some information giving hints on the community the current user belongs to (for example, the time spent viewing video 0) may help us to improve our prefetch strategy.

5.2. Handling composite user communities

Ideally, we only need to duplicate our optimization mechanisms for prefetching each profile. We suppose there is a known number of K profiles (denoted as $\{P_1 \ldots P_K\}$) but this assumption will be relaxed later. For every profile P_i, we can define an MDP similar to those presented in the previous sections.

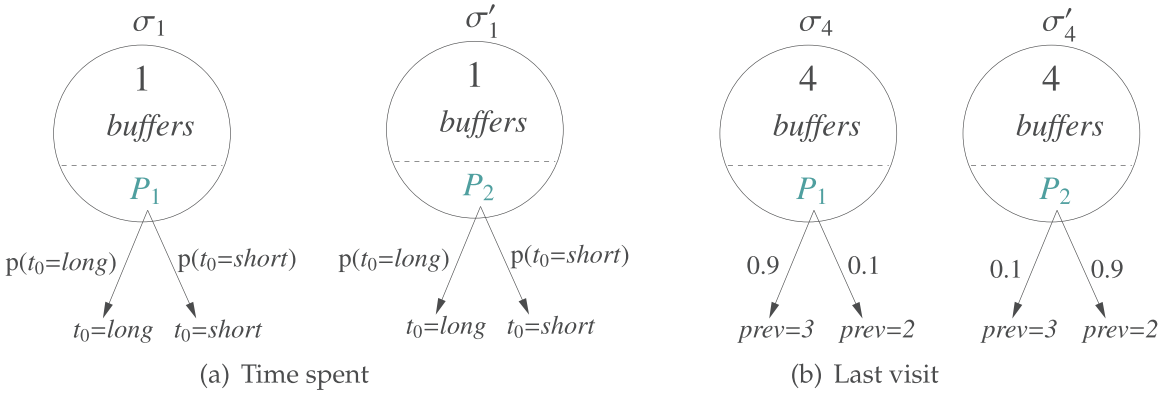


Fig. 9. Time spent and last visit observations for two profiles.

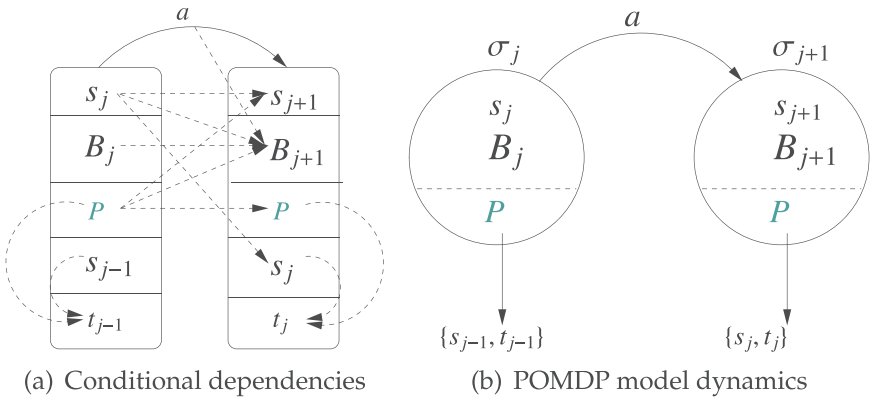


Fig. 10. POMDP integrating the user's profile.

In particular, a buffer state introduced in Section 3 can be easily extended by tagging it with a profile P_i (Fig. 8). If we suppose that we have a set of navigations of users belonging to profile P_i, then we can estimate an MDP model (transition probabilities between buffer states) specific to this profile. Therefore we can optimize various policies and find optimal policies for each community. These policies naturally perform better (for their community) than a single policy computed for the whole, mixed, community. There is still one important problem pending: how to automatically associate a user to a profile?

One way to model this problem is to say that the profile for the current user is hidden (i.e., we are not able to compute it). An MDP with a state that contains a hidden variable becomes a partially observable MDP (POMDP). Since we are unable to *observe* the profile, we will *infer* it from other observable data that we are going to define. As shown by the example of Section 5.1, we might estimate a user profile by using the time spent in the navigation states and the access sequence. To distinguish between profiles P_1 and P_2 while in Video 1, we can take into account the time (long or short) spent in the previous Video 0. In the same way, while in Video 4, we can infer the most probable profile (i.e., P_1 and P_2 resp.) according to the previous visited videos (i.e., 2 and 3 resp.). These information (the time spent in the visited videos and the sequence of visited objects) that enables us to estimate the profile of the user will be our *observations*.

An observation function can be associated with a buffer state, extended with a profile, as shown in Fig. 9(a) and (b). In buffer state σ_4 (resp. σ'_4) corresponding to profile P_1 (resp. P_2) we can observe that the previous state is 3 more (resp. less) often than 2. Moreover, in buffer state σ_1 (resp. σ'_1) corresponding to profile P_1 (resp. P_2), the probabilities to observe a short (resp. long) sojourn time in the previous state are different: $p(t_0 = long|\sigma_1)$ is big, whereas $p(t_0 = long|\sigma'_1)$ is small. Thanks to different distributions of observation probabilities we can distinguish among the hidden states (i.e., distinguish among the profiles).

We now formally describe our POMDP model, which is defined by its underlying MDP $(S; A; T; p; r_t)$ and a set of observations O. There is also an observation function O: $S \to \Pi(O)$ that maps every state s to a probability distribution on the observations' space. The probability to observe o knowing the agent's state s will be referred as $O(s, o) = P(o_t = o|s_t = s)$. Solving POMDPs is a much more difficult task than solving Markov Decision Processes (Singh et al., 1994).

5.3. Formal POMDP model for prefetching

In our case, we generalize MDP into POMDP by introducing observations as those showed by Fig. 11 and by adding a hidden variable (profile) to the buffer states and also taking into account the probability distribution of observations according to previous states and the sojourn time in these states. We show, below, that by factorizing the conditional dependencies between decisional variables, this model maintains the Markov property of the hidden states.

Let s_j be the video visited, B_j be the buffer state, and t_j be the time spent by the user while playing video s_j. Let C denote any other contextual information (such as the video or user genre, the time of video playing, etc.). We should notice that the history of the

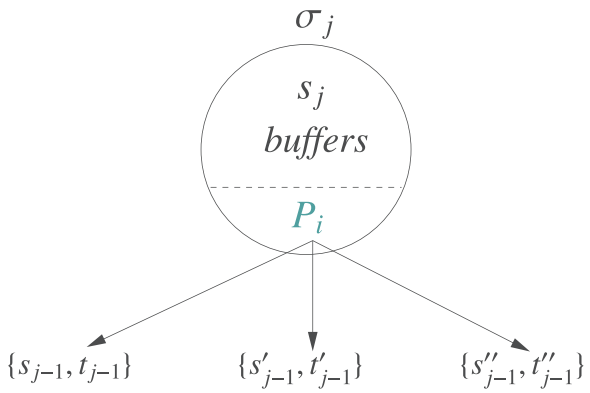


Fig. 11. Several tuples $\{s_{j-1}, t_{j-1}\}$ that may be observed in POMDP state σ_j.

navigation path is somehow hidden inside each profile P. Under the knowledge of each profile P, we make the following assumptions, in which the next object visited, the next buffer states, and the time spent on the current video, does not depend on C.

Assumption 1. The transition probability towards the next object depends only on the current object and the profile of the user P. In other words, the transition process between objects is Markovian:

$$p(s_{j+1}|s_j, P, C) = p(s_{j+1}|s_j, P)$$

Assumption 2. The next buffer state B depends only on its current value, the current object, the prefetching action a, and profile P:

$$p(B_{j+1}|B_j, s_j, a, P, C) = p(B_{j+1}|B_j, s_j, a, P)$$

Assumption 3. The time spent by the user depends only on the current video and the user's profile.

$$p(t_j|s_j, P, C) = p(t_j|s_j, P)$$

The conditional dependencies of the three assumptions above are graphically represented in Fig. 10(a). Under the assumptions above, the prefetching decision mechanism can be modeled using the POMDP illustrated in Fig. 10(b). $\sigma_j = \{s_j, B_j, P\}$ is the POMDP hidden state at instant j and $o_j = \{s_j, B_j, s_{j-1}, t_{j-1}\}$ is the global observation perceived at that moment. This global observation includes both observable elements of the hidden state and the "proper", formal observations t_{j-1} and s_{j-1}.

Theorem 1. *The process σ is a Markovian decision process. Moreover, the process of global observations o depends only on the underlying Markovian process σ.*

Proof.

$$p(\sigma_{j+1}|\sigma_j, a, C) = p(s_{j+1}, B_{j+1}, P|s_j, B_j, P, a, C) = p(s_{j+1}, B_{j+1}|s_j, B_j, P, a, C)$$

$$= p(s_{j+1}|s_j, B_j, P, a, C) \cdot p(B_{j+1}|s_{j+1}, s_j, B_j, P, a, C)$$

$$= \underbrace{p(s_{j+1}|s_j, P)}_{\text{Assumption 1}} \cdot \underbrace{p(B_{j+1}|s_j, B_j, P, a)}_{\text{Assumption 2}} = p(\sigma_{j+1}|\sigma_j, a)$$

$$p(o_j|\sigma_j, C) = p(s_j, B_j, s_{j-1}, t_{j-1}|s_j, B_j, P, C)$$

$$= p(s_{j-1}|s_j, B_j, P, C) \cdot p(t_{j-1}|s_j, B_j, s_{j-1}, P, C)$$

$$= \underbrace{p(s_{j-1}|s_j, P)}_{\text{Assumption 1}} \cdot \underbrace{p(t_{j-1}|s_{j-1}, P)}_{\text{Assumption 3}} = p(o_j|\sigma_j) \qquad \square$$

Next, we show that memorizing observable variables over a sliding window of finite time period is sufficient for eliminating the ambiguity among the hidden states. In which case, the POMDP model becomes equivalent to an MDP model where the hidden variables are replaced by the history of observations. The idea of using the recent past of the observed process to find the appropriate action for an POMDP has been proposed in Dutech (2000). The author details the so-called adapted policies acting over an extended state space composed of n-order action-observation trajectories and solve the original POMDP by using MDP resolution algorithms on this extended space.

Theorem 2. *If a profile can be completely estimated from the last L observations, then the stochastic process associated with these observations is Markovian.*

Proof. Suppose profile P is completely estimated using the last L observations:

$$p(P|o_j, o_{j-1}, \ldots, o_{j-(L-1)}, C) = p(P|o_j, o_{j-1}, \ldots, o_{j-(L-1)})$$

Then:

$$p(\sigma_j|o_j, o_{j-1}, \ldots, o_{j-(L-1)}, C) = p(s_j, B_j, P|s_j, B_j, s_{j-1}, t_{j-1}, o_{j-1}, \ldots, o_{j-(L-1)}, C)$$

$$= p(P|s_j, B_j, s_{j-1}, t_{j-1}, o_{j-1}, \ldots, o_{j-(L-1)}, C)$$

$$= p(P|o_j, o_{j-1}, \ldots, o_{j-(L-1)})$$

$$= p(\sigma_j|o_j, o_{j-1}, \ldots, o_{j-(L-1)})$$

Let us call O_j the sequence of L last observations at instant j : $O_j = \{o_j, o_{j-1}, \ldots, o_{j-(L-1)}\}$. We are now able to prove that the process $\{O_j\}$ is Markovian:

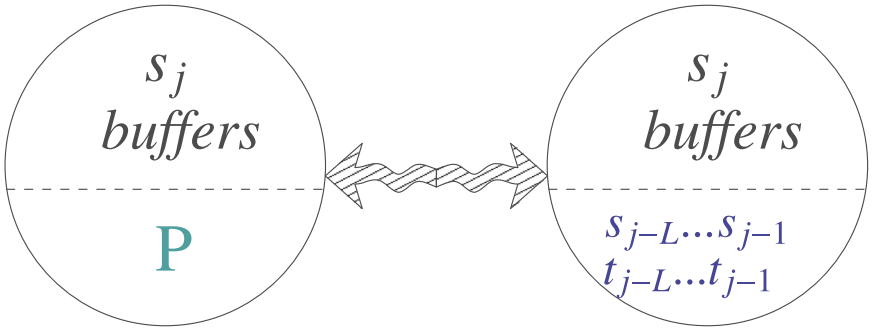
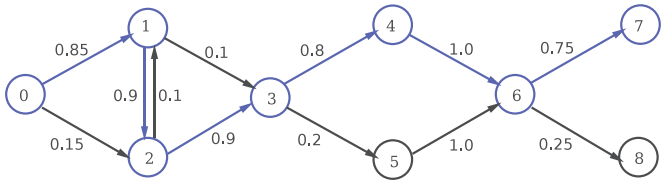
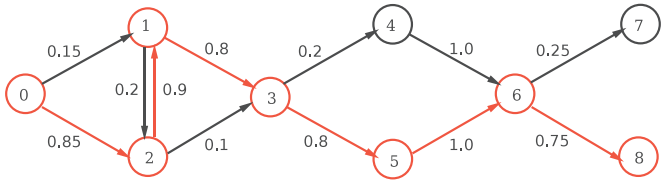


Fig. 12. Bringing the POMDP observations' history into a corresponding MDP state.



(a) Profile P_1



(b) Profile P_2

Fig. 13. Graph of transition probabilities associated to user profiles.

$$
\begin{aligned}
p(O_{j+1}|O_j, a, C) &= p(o_{j+1}, o_j, \ldots, o_{j-(L-2)}|o_j, \ldots, o_{j-(L-2)}, o_{j-(L-1)}, a, C) = p(o_{j+1}\\
&|o_j, \ldots, o_{j-(L-2)}, o_{j-(L-1)}, a, C) = \sum_{\sigma_{j+1}} p(o_{j+1}\sigma_{j+1}\\
&|o_j, \ldots, o_{j-(L-2)}, o_{j-(L-1)}, a, C) = \sum_{\sigma_{j+1}} p(\sigma_{j+1}\\
&|o_j, \ldots, o_{j-(L-2)}, o_{j-(L-1)}, a, C)\cdot p(o_{j+1}|\sigma_{j+1})\\
&= \sum_{\sigma_{j+1}} \left(\sum_{\sigma_j} p(\sigma_{j+1}\sigma_j|o_j, \ldots, o_{j-(L-2)}, o_{j-(L-1)}, a, C) \right) \cdot p(o_{j+1}|\sigma_{j+1})\\
&= \sum_{\sigma_{j+1}} \left(\sum_{\sigma_j} p(\sigma_j|o_j, \ldots, o_{j-(L-2)}, o_{j-(L-1)}, a, C)\cdot p(\sigma_{j+1}|\sigma_j, a) \right) \cdot p(o_{j+1}|\sigma_{j+1})\\
&= \sum_{\sigma_{j+1}} \left(\sum_{\sigma_j} p(\sigma_j|o_j, \ldots, o_{j-(L-2)}, o_{j-(L-1)})\cdot p(\sigma_{j+1}|\sigma_j, a) \right) \cdot p(o_{j+1}|\sigma_{j+1})\\
&= p(O_{j+1}|O_j, a) \qquad \square
\end{aligned}
$$

It is therefore natural to solve our prefetching problem using the memory of recent navigations instead of the profile. These observations give hints on the profile and allow to find optimal prefetching actions. A state of the obtained MDP is given by Fig. 12.

Assuming that the profile can be estimated from the last L observation, we can therefore transform our POMDP into the MDP whose states are sequences of last L observations. There is much redundancy in composing such sequence, however, and we may eliminate part of this redundancy using the conditional dependencies previously seen (Fig. 10(a)). In fact, we can keep only the following information in the buffer state: the objects visited $\{s_j, \ldots, s_{j-L}\}$, the times spent $\{t_{j-1}, \ldots, t_{j-L}\}$ and the current buffers B.

6. Experimental results

We validate our new, history-based model and show its abilities of integrating various profiles by proposing the optimal action for each profile. First, we present two examples that show that our approach is well defined:

1. Profiles with the same average playing time and different transition probabilities.
2. Profiles with the same transition probabilities and different playing times.

Second, we present a more complex scenario involving four user profiles and show, how the degree of history can give us hints on the underlying profile, thus improving the overall latency obtained by the history-based MDP policies.

6.1. Profiles with the same average playing time and different transition probabilities

The two navigation profiles that we consider are shown in Fig. 13 (transition probabilities) and Table 4 (characteristics of media objects). Average playing times for these profiles are the same (second line of Table 4). In these simulations, we consider a variable bandwidth from 3 to 4 Mbps; from the spare bandwidth the prefetching agent uses a maximum of 50%.

First, we simulate a navigation corpus for each profile and deduce the corresponding optimal policies: $\pi^*_{P_1}$ and $\pi^*_{P_2}$. Second, we mix the two corpus and, starting from these mixed navigations, we obtain two optimal policies ($\pi^*_{m,H=0}$ and $\pi^*_{m,H=1}$) with respect to the model that has been used: MDP without history (Section 3) and MDP with a 1 degree history (Section 5.3). Then these two policies are compared on the mixed corpus. Thanks to its memory, the history-based policy is able to exhibit the two mixed profiles and applies an optimal policy to each of them. Table 5 shows latencies obtained for the compared policies.

Let us give an intuitive explanation of these results. The relatively short time spent on the first part of multimedia document (components 0, 1, 2) makes essential the prefetching decision on component 3. Here, the agent should choose between 4, 6, and 7 for profile P_1 and between 5, 6 and 8 for P_2. For profile P_1, stochastic conditions (bandwidth, playing time, transitions) direct the agent to prefetch 7 as soon as it gets to 3. Similarly, for profile P_2, the agent decides to prefetch 8 while in 3.

In the mixed corpus, the agent equitably sees transitions to 4 and 5 from 3. In the same way, components 7 and 8 can be equitably reached. This makes the agent choose 6 (in 3) which is not an optimal action neither for P_1 nor for P_2. This gives an explanation for the deceiving performance of history-less optimal policies when using a corpus mixing different profiles. In this case, by simply memorizing the last visited component allows the agent to distinguish between the two profiles as early as when it is in 3. Therefore it chooses to prefetch 7 if it comes from 2 (profile P_1) and 8 if it comes from 1 (profile P_2). One can notice that the average latency observed for the mixed corpus with $\pi^*_{m,H=1}$ gets very close to the average optimal latency for the profile (i.e., the first row in Table 5).

Table 5
Average latencies for two profiles according to the resolution method.

Users Corpus	MDP models	Profile P_1	Profile P_2
Separate corpus	Memory-less	**8.772**	**10.483**
Mixed corpus	Memory-less ($\pi^*_{m,H=0}$)	12.435	15.814
	With memory ($\pi^*_{m,H=1}$)	**8.782**	**10.494**

Table 4
Characteristics of components of multimedia document.

Object (i)	0	1	2	3	4	5	6	7	8
Duration d_i (s)	10	10	10	60	180	180	10	300	300
Bitrate br_i (Kbps)	2400	2400	2400	500	500	500	3200	3200	3200
Buffer B_i (Kb)	7200	7200	7200	1200	1200	1200	18,000	24,000	30,000

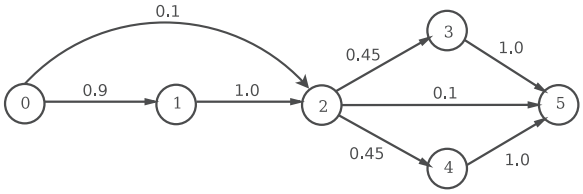


Fig. 14. Transition graph shared by the two user profiles.

Table 6
Access times for each of the two profiles.

Object (i)	Duration (s)	Profile P_1 (%)	Profile P_2 (%)	br_i (Kbps)	Buffer B_i (Kb)
0	10	20	50	1200	600
1	10	30	80	1200	600
2	100	90	10	500	500
3	100	10	90	500	1000
4	100	10	90	500	500
5	50	100	100	2400	7200

Table 7
Latencies for profiles of Fig. 14 obtained for different models.

Users Corpus	MDP models	Profile P_1	Profile P_2
Separate corpus	Memory-less	**2.790**	**2.208**
Mixed corpus	Memory-less ($\pi^*_{m,H=0}$)	2.924	2.836
	With memory ($\pi^*_{m,H=1}$)	**2.793**	**2.210**

Table 8
Characteristics of video streams for the four profiles.

Object (i)	0	1	2	3	4	5	6	7	8
Duration d_i (s)	40	25	35	30	60	70	50	55	65
Bitrate br_i (Kbps)	1675	1760	1814	1656	2412	3210	3196	3048	3096
Playout B_i (s)	8.2	8.3	7.9	8.7	9.8	9.7	9.9	9.3	9.6

6.2. Profiles with the same transition probabilities and different playing times

Similar results are obtained if profiles differ only by the average time spent in the components. By keeping the same methodology, we use profiles described in Fig. 14 and Table 6 and show that a history-based optimal policy is capable of exhibiting them. In these simulations, we consider a variable bandwidth from 1.5 to 2 Mbps; from the spare bandwidth, the prefetching agent uses a maximum of 50%. For each profile, the average time associated to the component i is represented by a percentage of the duration d_i.

Table 7 confirms the same good, predictable results. The intuitive explanation behind these results is very simple: while playing components 1 and 2 the prefetching agent would see if the user has spent less (profile P_1) or more (profile P_2) time so far and therefore guess the optimal action for the remaining content.

6.3. More complex scenario

In this case, we would like to show that providing a sufficient history level, our prefetching policies could improve their performances as if they knew intimately the composition of a mixed navigation corpus composed of several user profiles. More precisely, we consider a collection of 9 movie trailers collected from YouTube and store this content on a server located in a research lab from Toulouse.

The collection of video trailers was requested multiple times from different client stations located in Bucharest. For each video stream, we use YouTube JavaScript Player API to capture two events: *unstarted* and *playing*. Averaging the time difference between these two events gives us an estimation of each video start-up latency. Considering also the average video bit rate, we estimated the size of play-out buffer for each video. The characteristics of those 9 videos and their average playing are shown in Table 8.

For this case, we consider four navigation user profiles shown in Fig. 15. The intuitive explanation beyond their design is that

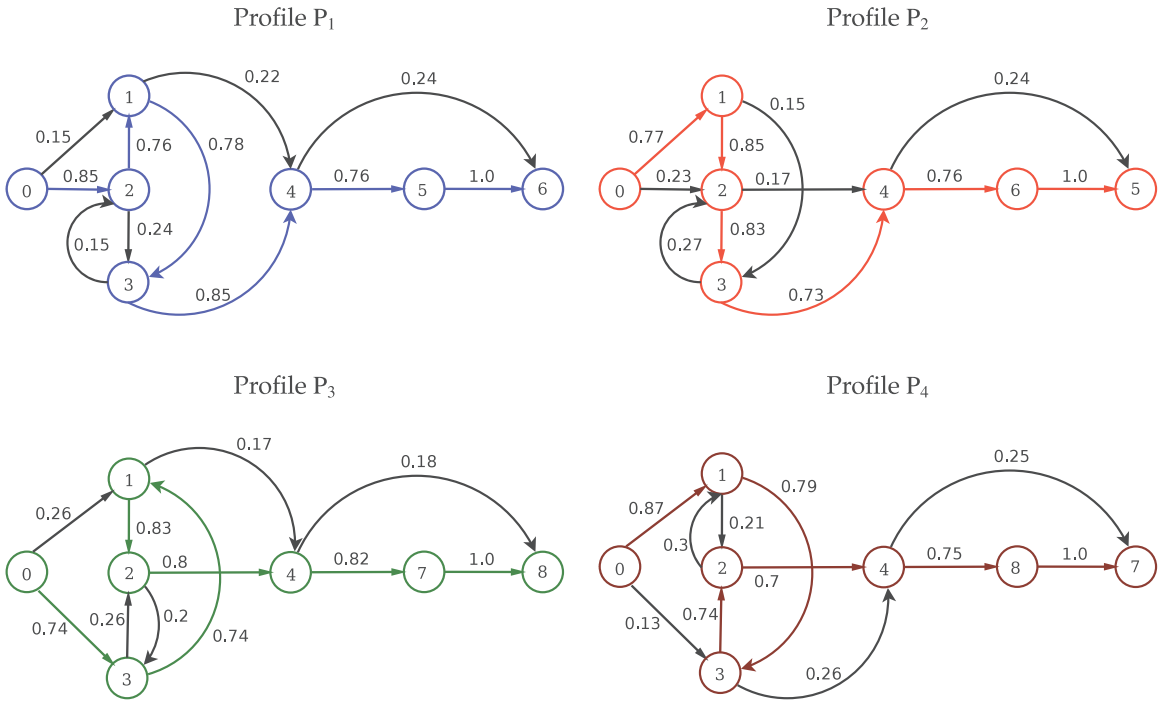


Fig. 15. Four navigation user profiles.

Table 9
Average latencies for all policies: heuristics and optimal with various degrees of history.

Policy Type	Heuristics	Optimal policies		
Simple prefetching	*BEST-FIRST*	$H = 0$	$H = 1$	$H = 2$
	15.405	13.620	12.310	11.029
	–	12%	20%	**28%**
Proportional prefetching	*PROP*	$H = 0$	$H = 1$	$H = 2$
	12.990	12.305	11.668	10.707
	–	5%	10%	**18%**
Sequential prefetching	*MAINLINE*	$H = 0$	$H = 1$	$H = 2$
	14.254	13.155	12.105	10.987
	–	8%	15%	**23%**

Table 10
Average number of video stalls for simple policies (1 + stands for the initial buffering).

QoE Metric	Dummy policy	*BEST FIRST*	Optimal policies		
			$H = 0$	$H = 1$	$H = 2$
# of video stalls	1+4.211	1+1.562	1+1.218	1+1.037	1+0.785
Mean opinion score	2.0630	2.718	2.954	3.154	3.424

each profile can be characterized by a significant sequence, namely:

P_1: $0 \to 2 \to 1 \to 3 \to 4 \to 5 \to 6$ P_2: $0 \to 1 \to 2 \to 3 \to 4 \to 6 \to 5$ P_3: $0 \to 3 \to 1 \to 2 \to 4 \to 7 \to 8$ P_4: $0 \to 1 \to 3 \to 2 \to 4 \to 8 \to 7$

One can easily observe that, following the path of those remarkable sequences, when the prefetching agent arrives in video stream 4, looking one step behind (i.e., a history $H = 1$) can only help him distinguish between two profiles classes: $\{P_1, P_2\}$ if it comes from 3 and $\{P_3, P_4\}$ if it comes from 2. Nevertheless, increasing the history (i.e., looking two steps behind) allows the prefetching agent to completely *infer* to the 4 profiles involved. For example, if it comes from the path $1 \to 3 \to 4$, then the presumed profile (i.e., statistically speaking) is clearly P_1.

As before, we conduct a series of simulations, mixing the four profiles and apply our optimal prefetching policies with different degrees of history. In these simulations, the bandwidth varies uniformly between 3 and 4 Mbps and from the spare bandwidth, the prefetching agent uses a maximum of 50%. We solve the model using *Q-learning* and obtain for each category of policy (i.e., simple, proportional and sequential) three policies corresponding with three history levels (i.e., $H = 0$, $H = 1$, $H = 2$). The key aspect in these simulations is that the policy is not aware of the various profiles whose it is applied to. One can notice the decrease of these latencies with the history level, for each category of policy (Table 9).

One interesting aspect of our experimental part concerns the video quality results. As we handle the original video sequences, we do not address any compression issue like PSNR. Even if we do not perform any subjective evaluation like MOS (i.e., mean opinion score), minimizing the latency should lead to an increase of QoE (i.e., Quality of Experience). However, counting the number of video stalls (start-up latencies greater than 3 s) we could estimate the MOS parameter on a scale from 0 to 5, by following an existing model proposed by Hossfeld et al. (2011). Table 10 shows an important gain in MOS (i.e., approx. 0.7) just by using a heuristic simple policy (i.e., BEST-FIRST) while using an optimal policy enriched by a 2-step history leads to an even significant gain in the quality of experience.

7. Related work

Prefetching is a prominent problem in many computer science research areas. One of the most recent instance of such problems deals with mobile applications. The system presented by Parate et al. (2013) mitigates launch latencies by making applications prefetch practical on mobile phones. A prediction model is learnt from real-life logs based on the probability distribution of the applications to be used in the future.

Historically, the concept of prefetching has long been used in a variety of distributed and parallel systems to hide communication latency (Culler et al., 1998). In operating systems research, a notable work in automatic prefetching for reducing latency in operating systems is due to Griffioen and Appleton (1994). They proposed predicting future file accesses based on past file activities that are characterized by probability graphs. More recently, Web prefetching (Chen and Zhang, 2003) has been extensively studied as a basic mechanism for downloading documents in advance while a user is surfing. Since the Web surfer follows the hyperlinks in an unpredictable way, the choice of the Web pages to be prefetched must be computed online by a prefetching policy. A generalization of this concept to multimedia playback is natural. Prefetching in the context of non-linear access to media data is also studied in Krishnamoorthi et al. (2014) for on-demand streaming of branched video.

In this paper, we address the problem of prefetching in the context of a collection of streaming media (e.g., videos). Our approach targets optimization for different types of prefetching strategies and a well defined performance metric, i.e., the user-perceived latency. Moreover, our prefetching model remains optimal even in the presence of multiple user profiles, adapting the best strategy for each of them. To the best of our knowledge, achieving optimality for the prefetching problem, in the context of multiple user profiles has never been addressed before. The next section presents the most important performance metrics for the prefetching problem in the context of web or media prefetching. Then, we briefly discussed some relevant works that target optimization according to some of these metrics.

7.1. Prefetching performance metrics

The prefetching performance criteria could be classified into 3 main categories (Domènech et al., 2006):

- *How accurate is the prediction*: The first category is often used when comparing prediction techniques and includes those metrics that quantify both the efficiency and the efficacy of the technique. For example, precision measures the ratio of good predictions to the number of predictions (Davison, 2004). Byte precision measures the percentage of prefetched bytes that are subsequently requested (Bouras et al., 2004). Precision just evaluates the algorithm without considering physical system restrictions (e.g., cache, network or time restrictions) and it can be seen as a theoretical metric. Recall measures the percentage of requested objects that were previously prefetched.
- *The cost of prefetching:* The second category quantifies the additional cost that prefetching incurs (e.g., increases in network traffic or computation time); and can be seen as complementary measures. Traffic increase quantifies the increase of network traffic (in bytes) due to unsuccessfully prefetched documents. It is also called wasted network traffic, extra bytes, network traffic, or bandwidth ratio (Bouras et al., 2004). Server load ratio is

Table 11
Prefetching policies comparison.

Prefetching properties	Heuristics (2002)[a]			Optimal policies		
	BEST FIRST	*PROP*	*MAINLINE*	(2007)[b]	(2014)[c]	This paper
Prefetched stream(s)	1	*K* in parallel	*K* in sequential	1	*K* in parallel	*K* in parallel and sequential
Viewing duration variability	–	–	–	✓	✓	✓
Bandwidth variability	–	–	–	✓	–	✓
Multiple user profiles	–	–	–	–	–	✓

[a] Tuah et al. (2002).
[b] Charvillat and Grigoras (2007).
[c] Morad and Jean-Marie (2014b), see also derived heuristics (Morad and Jean-Marie, 2014a).

defined as the ratio between the number of requests for server service when prefetching is employed to the number of requests for server service when prefetching is not employed. Some researchers also discuss how the overhead impacts on the server performance. In this manner, prediction time quantifies the time that the predictor takes to make a prediction, a metric used in Dongshan and Junyi (2002) to compare different predicting algorithms.

- *Decrease in latency*: Finally, the third category summarizes the performance achieved by the system from the users' point of view. Several names have been used for this metric such as latency, access time, or responsiveness (Khan and Tao, 2005). The three categories are closely related since, in order to achieve a good overall performance, prefetching systems must trade off the contents prediction (first category) and the system cost increase due to prefetching (second category). Metrics belonging to this category include those aimed at measuring the end-to-end responsiveness (e.g., user or proxy related latencies), some of which are difficult to quantify.

Beyond the three categories of metrics above, some authors (Wu and Kshemkalyani, 2006) propose the use of performance metrics combining mixed predictive metrics with resource utilization metrics. They seek to find the right balance between quality of prediction and consumption of bandwidth. These metrics generally increase with the quality of the prediction and decrease with the extra bandwidth generated. Consequently, the value reflects the system performance of prefetching, in its entirety. By optimizing these metrics, policies even more sophisticated for optimal prefetching can be considered.

Our research belongs to the third category as we choose to optimize the average latency perceived by user along its navigation through a multimedia document content. Several other work deals with this optimization problem and three representative approaches will be presented further.

7.2. Prefetching optimization approaches

The approach developed in Tuah et al. (2002) provides a sequential policy aiming to optimize the latency at each navigation step. In their work, the authors assume the knowledge of the available bandwidth, the size of the media objects, and the time spent by the user in each object. Moreover, the media objects are available only by download and not by a streaming process. Henceforth, the prefetching process occupies all the available bandwidth during the presentation of the current media object. At each user request, a new optimization prefetching problem is solved based on the current media demand and network bandwidth. The prefetching is modeled as a maximization problem under constraints and solved (in a approximated manner) by decomposing it into two stretch knapsack sub-problems. Pons (2005) improves this framework using a Markov model to estimate object's access probabilities.

A second approach aiming to reach the optimality for the sequential policy proposed in Tuah et al. (2002) is presented by Angermann (2002). The prefetching mechanism uses a Markov model that is downloaded to the client's system that becomes personalized to the client's distinct behavior. The framework used is slightly different from Tuah et al. (2002) and allows the media objects to be stored on different servers, the cache storage is supposed unlimited and the viewing time for each object is not anymore constant but distributed following a Zipf low (Breslau et al., 1999). Under these hypothesis, the author shows that the media objects should be prefetched in decreasing order of their access probabilities.

The two previous approaches provide, under specific hypothesis, optimal (or near optimal) prefetching policies to reduce the latency perceived by the user. Nevertheless, their optimality is limited at a single navigation step. Just like some heuristics seen in Section 2, these strategies target only immediately reachable media objects (i.e., a single transition). Therefore, they provide only a local optimality. Reaching an in-depth optimality, i.e., reducing the global navigation latency, implies considering not only immediate but also late effects of a prefetching decision.

The third approach (Khan and Tao, 2001) targets this goal under the hypothesis that the network bandwidth is constant and the percentage allocated to the prefetching process is given by a parameter β. As in our case, each media object i is composed of a lead segment (i.e., the amount b_i) and a stream segment. The authors provide principles and several hints to solve the problem without offering a rigorous solution for the general case. For particular cases, they suggest a prefetching algorithm that gives the priority to the more popular and smaller media objects.

Recently, prefetching policies for VoD systems and P2P video streaming have been devised to anticipate seeking actions or VCR operations. Any seek operation from a user usually results in latency and learning segment access probability is a key problem. Optimality of prefetching policies in this P2P context is twofold. The distributed nature of media data and varying popularity lead to specific optimization criteria (He et al., 2009). Similarly, the prefetching and caching of online TV services provided by a hosting service is examined in Krishnappa et al. (2011b), where the authors proposed the prefetch of the most popular videos of a week. The idea of prefetching videos/scenes which are watched with a high probability is used in our work too, but we integrate a profiling strategy and dynamically personalize the media delivery while a specific user with a specific profile is browsing a collection of media.

Another work that extend our initial MDP model by Charvillat and Grigoras (2007) is presented in Morad and Jean-Marie (2014a, b). The authors formally consider non-elementary MDP policies and buffer states (e.g., proportional policies as defined in Section 2) but adopt less realistic assumptions: no randomness in the prefetching process, deterministic bandwidth, deterministic bandwidth allocations, users committed to view a selected video

entirely, etc. They also consider only one averaged user profile while our model is capable of demixing several user profiles.

8. Conclusion

The problem we address in this paper is the reduction of navigation latencies in the context of non-linear media accesses by means of prefetching policies. We deliver rigorous solutions to this problem thanks to our modeling by Markov Decision Processes and their associated resolution algorithms. Our approach differentiates from long-term prefetching techniques by several aspects. First, we believe that prefetching decisions while navigating in a collection of media objects should not be fixed before hand. Second, these decisions should take into account the existing links (semantically or structurally) between media objects together with a short-term memory witnessing the user behavior. Third, we believe that the prefetching process should use only the spare bandwidth available that is, by its very nature, random.

The strength of our MDP models is their capacity to integrate both user behavior and resource availability. The key points of our prefetching policies are twofold: they are optimal in the sense of our Markovian models and they can be computed automatically. Moreover, we derived a hierarchy of prefetching policies classes and obtained, inside each class, the optimal policy. Among them, the more sophisticated are able, under specific hypothesis, to manage the coexistence of different user profiles. In this sense, we contribute to the modern momentum whose aim is to mix modern technology components (often observable) with human factors (often partially observable) to manage dynamic execution contexts. We believe that our prefetching framework can also be adapted to other video interactions (e.g., zooming, panning, and seeking) or other contexts such as 3D streaming. These are the main avenues for our future research.

References

Angermann, M., 2002. Analysis of speculative prefetching. SIGMOBILE Mob. Comput. Commun. Rev. 6 (April (2)), 13–17.

Bouras, C., Konidaris, A., Kostoulas, D., 2004. Predictive prefetching on the web and its potential impact in the wide area. World Wide Web 7 (2), 143–179.

Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S., 1999. Web caching and Zipf-like distributions: evidence and implications. In: The Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, pp. 126–134.

Charvillat, V., Grigoras, R., 2007. Reinforcement learning for dynamic multimedia adaptation. J. Netw. Comput. Appl. 30 (3), 1034–1058.

Chen, X., Zhang, X., 2003. A popularity-based prediction model for web prefetching. Computer 36 (March (3)), 63–70.

Cisco, 2015. Cisco Visual Networking Index: Forecast and Methodology, 2014–2019. White Paper, May.

Culler, D., Singh, J.P., Gupta, A., 1998. Parallel Computer Architecture: A Hardware/Software Approach. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, August.

Davison, B.D., 2004. Learning web request patterns. In: Web Dynamics—Adapting to Change in Content, Size, Topology and Use, pp. 435–460.

Domènech, J., Gil, J.A., Sahuquillo, J., Pont, A., 2006. Web prefetching performance metrics: a survey. Perform. Eval. 63 (October(9)), 988–1004.

Dongshan, X., Junyi, S., 2002. A new Markov model for web access prediction. Comput. Sci. Eng. 4 (6), 34–39.

Dutech, A., 2000. Solving POMDPs Using Selected Past Events.

Fomin, F.V., Giroire, F., Jean-Marie, A., Mazauric, D., Nisse, N., 2014. To satisfy impatient web surfers is hard. Theor. Comput. Sci. 526, 1–17.

Griffioen, J., Appleton, R., 1994. Reducing file system latency using a predictive approach. In: USTC'94: Proceedings of the USENIX Summer 1994 Technical Conference on USENIX Summer 1994 Technical Conference. USENIX Association, Berkeley, CA, USA, pp. 13–13.

He, Y., Shen, G., Xiong, Y., Guan, L., 2009. Optimal prefetching scheme in p2p vod applications with guided seeks. IEEE Trans. Multimedia 11 (1), 138–151.

Hossfeld, T., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P., Schatz, R., 2011. Quantification of youtube qoe via crowdsourcing. In: ISM.

Khan, J.I., Tao, Q., 2001. Prefetch scheduling for composite hypermedia. In: IEEE International Conference on Communications, vol. 3, pp. 768–773.

Khan, J.I., Tao, Q., 2005. Exploiting webspace organization for accelerating web prefetching. Web Intell. Agent Syst. 3 (2), 117–129.

Krishnamoorthi, V., Carlsson, N., Eager, D., Mahanti, A., Shahmehri, N., 2014. Quality-adaptive prefetching for interactive branched video using http-based adaptive streaming. In: Proceedings of the ACM International Conference on Multimedia, MM '14, pp. 317–326.

Krishnappa, D.K., Khemmarat, S., Gao, L., Zink, M., 2011b. On the feasibility of prefetching and caching for online tv services: a measurement study on Hulu. In: Proceedings of the 12th International Conference on Passive and Active Measurement, PAM'11. Springer-Verlag, Berlin, Heidelberg, pp. 72–80.

Krishnappa, D.K., Zink, M., Griwodz, C., 2013a. What should you cache?: a global analysis on youtube related video caching. In: Proceedings of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV 1013, Oslo, Norway, February 27, 2013, pp. 31–36.

Krishnappa, D.K., Zink, M., Griwodz, C., Halvorsen, P., 2013b. Cache-centric video recommendation: an approach to improve the efficiency of youtube caches. In: Multimedia Systems Conference 2013, MMSys '13, Oslo, Norway, February 27–March 1, 2013, pp. 261–270.

Liang, K., Hao, J., Zimmermann, R., Yau, D.K., 2015. Integrated prefetching and caching for adaptive video streaming over http: an online approach. In: Proceedings of the 6th ACM Multimedia Systems Conference. ACM New York, NY, USA., pp. 142–152.

Morad, O., Jean-Marie, A., 2014a. On-demand prefetching heuristic policies: a performance evaluation. In: The 29th International Symposium on Computer and Information Sciences (ISCIS 2014), Krakow, Poland, pp. 317–324, October.

Morad, O., Jean-Marie, A., 2014b. Prefetching control for on-demand contents distribution: a Markov decision process model. In: IEEE 22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2014), Paris, France, September.

Parate, A., Böhmer, M., Chu, D., Ganesan, D., Marlin, B.M., 2013. Practical prediction and prefetch for faster access to applications on mobile phones. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp'13, pp. 275–284.

Pons, A.P., 2005. Improving the performance of client web object retrieval. J. Syst. Softw. 74 (3), 303–311.

Puterman, M.L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edition. John Wiley & Sons, Inc., New York, NY, USA.

Singh, S.P., Jaakkola, T., Jordan, M.I., 1994. Learning without state-estimation in partially observable Markovian decision processes. In: International Conference on Machine Learning, pp. 284–292.

Tuah, N.J., Kumar, M., Venkatesh, S., 1998. Investigation of a prefetch model for low bandwidth networks. In: Proceedings of the 1st ACM International Workshop on Wireless mobile multimedia. ACM, New York, NY, USA., pp. 38–47.

Tuah, N.J., Kumar, M., Venkatesh, S., Das, S.K., 2002. Performance optimization problem in speculative prefetching. IEEE Trans. Parallel Distrib. Syst. 13 (5), 471–484.

Watkins, C.J., Dayan, P., 1992. Q-learning. Mach. Learn. 8 (3), 279–292.

Wu, B., Kshemkalyani, A.D., 2006. Objective-optimal algorithms for long-term web prefetching. IEEE Trans. Comput 55 (January (1)), 2–17.