Viewport-Aware Deep Reinforcement Learning Approach for 360^o Video Caching

Pantelis Maniotis, *Student Member, IEEE* and Nikolaos Thomos, *Senior Member, IEEE*

***Abstract*—360^o video is an essential component of VR/AR/MR systems that provides immersive experience to the users. However, 360^o video is associated with high bandwidth requirements. The required bandwidth can be reduced by exploiting the fact that users are interested in viewing only a part of the video scene and that users request viewports that overlap with each other. Motivated by the findings of our recent works where the benefits of caching video tiles at edge servers instead of caching entire 360^o videos were shown, in this paper, we introduce the concept of virtual viewports that have the same number of tiles with the original viewports. The tiles forming these viewports are the most popular ones for each video and are determined by the users' requests. Then, we propose a reactive caching scheme that assumes unknown videos' and viewports' popularity. Our scheme determines which videos to cache as well as which is the optimal virtual viewport per video. Virtual viewports permit to lower the dimensionality of the cache optimization problem. To solve the problem, we first formulate the content placement of 360^o videos in edge cache networks as a Markov Decision Process (MDP), and then we determine the optimal caching placement using the Deep Q-Network (DQN) algorithm. The proposed solution aims at maximizing the overall quality of the 360^o videos delivered to the end-users by caching the most popular 360^o videos at base quality along with a virtual viewport in high quality. We extensively evaluate the performance of the proposed system and compare it with that of known systems such as Least Frequently Used (LFU), Least Recently Used (LRU), First In First Out (FIFO), over both synthetic and real 360^o video traces. The results reveal the large benefits coming from reactive caching of virtual viewports instead of the original ones in terms of the overall quality of the rendered viewports, the cache hit ratio, and the servicing cost.**

***Index Terms*—Deep reinforcement learning, 360^o video, tile-encoding, viewport-aware caching.**

I. Introduction

Interactivity in VR/AR/MR systems is facilitated by the use of 360^o video content. However, the interactivity associated with 360^o videos comes with a huge increase in the bandwidth needed to deliver the content to the users. This puts pressure on the network infrastructure demanding further investments to accommodate 360^o video related network traffic. Exploiting 360^o video coding flexibility, i.e., encoding in tiles, and caching at the edge servers, can be a remedy for the problem, as we have shown in [1], [2]. However, existing solutions assume known popularity, which may not always be the case. Further, existing solutions do not scale well with big content because of the cache optimization complexity. This naturally calls for caching systems that exploit tiles encoding and can estimate future content popularity trends, while preserving low complexity and scalability with respect to the number of 360^o video files.

P. Maniotis and N. Thomos are with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom (e-mail: {p.maniotis, nthomos}@essex.ac.uk).

In 360^o videos, a 360^o view of a scene is captured from a single point with the use of an omnidirectional camera. The captured scene is then mapped to the internal part of a spherical surface. Each user is assumed to be placed at the center of the sphere and is interested in watching only a portion of the scene, known as viewport. Typically, each viewport covers 120^o of the entire scene. According to the head movements of the user, the Head Mounted Display (HMD) dynamically alters the part of the scene that will be displayed. To prevent users from experiencing motion sickness and discomfort, the response of the system to the head movements should be as fast as the HMD refresh rate [3]. Considering that the refresh rate may be 120Hz, the whole system should project the requested viewport in less than 10ms. However, state-of-the-art network streaming architectures are not able to respond under these tight time constraints due to the end-to-end delay. Although transmitting the whole scene could help to overcome the above limitation, it is not an efficient strategy since the resolution of a 360^o video is commonly 4K, 8K, or even higher [4]. Thus, it would lead to significant bandwidth waste as only a part of the 360^o video would be eventually displayed.

In edge caching systems, Small Base Stations (SBSs), e.g., picocells and femtocells, are equipped with caches, which can store a limited amount of popular content files. This is inspired by the fact that only a small number of popular content accounts for most of the network traffic load [5]. As a result, when there are multiple content requests for a cached content at an SBS, these can be served from the cache directly instead of obtaining the content through the core network using pricey backhaul links. This allows users to receive the content with lower latency, and the use of the backhaul links is limited. The potential of using edge caching as a solution to address the challenges that 360^o video delivery faces in cellular networks, has been recently studied in [1], [2], [6]. These works showed that offline edge caching can be a prominent solution for 360^o video delivery, in particular, when tiles and layered encoding are used. The main drawback of [1], [2], [6] is that they assume that the content popularity profile is known in advance. However, often in practice, the content popularity changes dynamically and may not be known a priori, or the estimated distribution may not be accurate. For regular videos, this problem has been addressed by online caching schemes [7], [8]. These methods learn the optimal caching policy by observing previous video consumption patterns. Though these methods are efficient for standard videos, they cannot be applied straightforwardly for 360^o video. This is because 360^o videos have considerably larger sizes than traditional videos, which limits the number of videos that can be stored at the SBSs caches. Furthermore, online caching schemes for regular videos have not been designed to take advantage of the fact that large parts of the video scene are never displayed, as is the case of 360^o video where users are interested in watching only a viewport. From the discussion above, it is clear that there is a vast need for online 360^o video caching schemes which exploit 360^o video features and do not necessitate the delivery of the entire video scene.

In this paper, we propose a reactive caching scheme for the transmission of 360^o video in cellular networks. To the best of our knowledge, this is the first online caching scheme for 360^o videos. Our method aims at maximizing the overall quality of the 360^o videos delivered to the users, without requiring *a priori* knowledge of 360^o video and tiles popularity distributions. In the studied problem, the popularity of tiles and videos varies with time, making the examined problem more challenging as 360^o video is characterized by tight delivery deadlines. Our method updates the cached content based on limited observations regarding 360^o video consumption patterns, obtained from previous users' requests. We adopt tiles and layered encoding of 360^o video because of the flexibility they offer to caching algorithms [1], [2], [6]. These methods encode 360^o videos into a number of independently encoded tiles and multiple layers, as shown in Fig. 1a. Encoding in tiles and layers allows network operators to cache in high quality at each SBS only the parts of the scene of each 360^o video (i.e., the tiles that correspond to these parts) that are the most popular to the users. Further, as only some tiles of the 360^o videos are popular, we introduce the concept of *virtual* viewport, which is shaped by the overlap that occurs because of the diverse users' requests for different viewports, as shown in Fig. 1b. Virtual viewports differ from the original ones in that the tiles that comprise them are not necessarily adjacent to each other, i.e., they do not form a rectangular area. A virtual viewport has the same number of tiles with regular viewports, but it consists of the most popular ones. When a user requests a viewport of a 360^o video in a certain quality, then if some tiles of the requested viewport also belong to the virtual viewport that is cached at the SBS that received the request, these tiles will be served from that cache. As a result, storing virtual viewports will lead to an increase in the cache hit ratio, due to the greater flexibility they provide in terms of which tiles to cache in high quality.

In order to determine which videos and virtual viewports to cache in each SBS, we first formulate the problem of 360^o video caching as a Markov Decision Process (MDP). The aim is to find the optimal set of 360^o videos and virtual viewports that should be cached at the SBS so that the overall quality delivered to the users is maximized. This is done by considering a limited history of users' requests. Although MDP offers an elegant way to describe our framework, the requirement of knowing the state transition probabilities makes it hard to evaluate the optimal policy (caching decisions per 360^o video and virtual viewport) for our system. This requirement can be lifted with the use of Q-learning [9]. Despite Q-learning convergence properties, it cannot be trivially applied for large-sized problems. To address this limitation of the Q-learning algorithm, we use the Deep-Q-Network (DQN) [10] variant of Q-Learning. We evaluate the performance of our solution for both real [11] and synthetic 360^o video traces, and compare its performance with that of known schemes such as the Least Frequently Used (LFU), Least Recently Used (LRU), First In First Out (FIFO) algorithms. The results illustrate the advantages of the proposed method compared to its counterparts, in terms of the overall quality users enjoy, the overall cache hit ratio, and the cost of delivering the requested content to the users.

In summary, the main contributions of our work are:

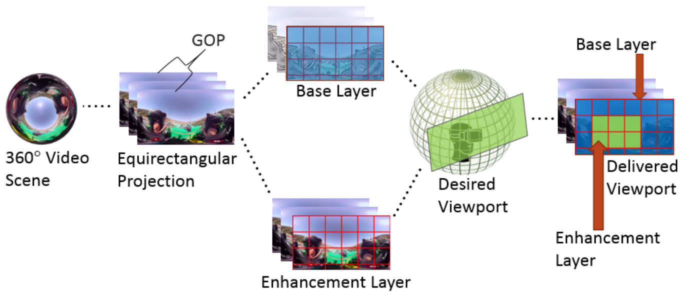
- *Reinforcement Learning framework:* We introduce a novel reinforcement learning framework for optimizing the content cache placement of 360^o videos, by formulating the problem of caching 360^o videos as a Markov Decision Process. Our solution aims at maximizing the overall video quality delivered to the users by taking into account both the 360^o videos and tiles' popularity. As Q-learning cannot be used because of the size of the problem, we employ DQN. DQNs enable us to solve large instances of the online cache optimization problem for 360^o videos.
- *Concept of Virtual Viewport:* We introduce the concept of the virtual viewport, which is shaped by the overlap of the diverse users' requests for different viewports. A virtual viewport is comprised of the most popular tiles of a 360^o video over the users' population. Virtual viewports enable us to reduce the size of the online cache optimization problem for 360^o videos.
- *Evaluation on real and synthetic 360^o video traces:* We extensively evaluate our proposed solution for real navigation patterns extracted from the dataset described in [11], as well as on synthetic navigation patterns in order to show the benefits coming from the introduction of virtual viewports, and also, the impact of different users 360^o video consumption patterns on the overall quality users enjoy.

The rest of the paper is organized as follows. In Section II, we overview work related to edge caching, reinforcement learning, and tile-encoding of 360^o videos. Next, in Section III, we describe our system setup. Right after, in Section IV we introduce the considered model of the users' requests. Then, we first formulate our problem as an MDP in Section V, and right after in Section VI, we show how DQN can be used to solve the cache placement problem for 360^o videos. In Section VII, we thoroughly evaluate the performance of the proposed scheme and compare it with other methods in the literature. Finally, we draw conclusions in Section VIII.

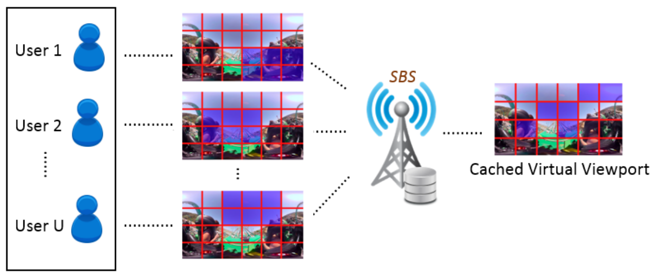
II. Related Work

In this section, we briefly overview the literature related to edge caching, online caching, and tile-based 360^o video streaming.

The use of edge caching has been proposed as an efficient way to bring content closer to the end-users and improve the quality of the delivered content [12], [13]. In addition, caching popular contents at the mobile edge servers has been shown to reduce the usage of the pricey backhaul links [14]–[16] and the network operation cost [17]. The optimal placement of layered



(a) Encoding of 360^o video in two quality layers and several tiles.



(b) Overlapping viewports for various user requests, and virtual viewport.

Fig. 1. Users request different viewports encoded in quality layers and tiles.

videos on edge caching systems is investigated in [18]. The decisions regarding which video layers to cache in each SBS are made by taking into account the caching cost, the available cache capacity at the SBSs, and the social groups formed by mobile users based on their content requests. Differently from [18], caching several representations of multiple videos that correspond to different qualities is examined in [13]. The cached representations are decided so that the aggregate distortion reduction of all the users is maximized while minimizing the cost related to downloading the representations. In [12], the delivery of 4K video quality in LTE-A networks is explored. That work aims to assure for 4K live streaming systems high Quality of Experience (QoE) to the users.

The aforementioned works consider that video popularity profiles are known, which in many cases is not possible. To address this limitation, the content popularity is predicted using reinforcement learning algorithms that exploit the demand history [7], [8], [19], [20]. Specifically, in [7], the SBSs learn the content popularity online, considering the switching cost related to the addition of new files to the cache. Contextual MABs are proposed for online cache optimization in [8] to take advantage of users' characteristics such as age, sex, etc. Neural networks (NN) [19], [20] can be used to decide the optimal cache placement when content popularity is unknown. Specifically, a Deep Reinforcement Learning-based framework aiming to maximize the long-term cache hit ratio is presented in [19]. To limit the action space in [19], an Actor-Critic algorithm based on the Wolpertinger architecture [21] is used. Differently, in [20], an Actor-Critic algorithm is presented where the actor uses the Gibbs distribution, and the critic uses a deep neural network to minimize the average transmission delay. To this aim, the users' scheduling and content caching policies are jointly designed.

The delivery of 360^o videos encoded by advanced video coding standards, e.g., H.265/HEVC, SHVC, that support the encoding of the 360^o videos into a number of quality layers and tiles has been studied in [22]–[25]. These systems exploit the fact that users are interested in viewing only a viewport of the 360^o video scene, and hence there is no need to deliver the whole scene in high quality. Differently from [22]–[25], in our previous work in [1], [2] we proposed a tile-based collaborative caching scheme for 360^o videos for video-on-demand systems, where we showed the benefits coming from making the caching decisions on a per tile basis and the advantages of exploiting SBSs collaboration. In contrast to [1], [2], authors in [6] examine a tile-based caching scheme that aims to optimize the error between the requested and cached tile resolutions across different viewports as well as the coverage of the tiles set. In their work, they examine the caching of tile streams both at different resolutions and in a layered encoding fashion. Differently from [1], [2], [6], authors in [26] examine the joint caching, transcoding, and delivery of 360^o videos. Though the works in [1], [2], [6], [26] showed promising performance for offline caching of 360^o video, they cannot be extended and be used straightforwardly for solving online caching problems studied here, as they assume known popularity. A motion-prediction-based mechanism is proposed in [27], where viewers' motion is predicted with the use of machine learning. Similarly, the navigation behaviors of users when watching 360^o videos on computers has been investigated in [28]. The results show that viewers have similar viewing patterns for certain 360^o video categories. A navigation-aware adaptive streaming strategy is presented in [29], where the aim is to optimize the rate at which a tile is downloaded during the navigation of the 360^o video. The rate per tile optimization problem is formulated as an integer linear programming problem. The proposed solution reveals the benefits of exploiting navigation patterns on both quality and navigation-smoothness. The impact of tile encoding on bandwidth saving, coding efficiency, and scalability is examined in [30], where a tile-aware video streaming system is proposed. The results show that an up to 80% bit-rate reduction is achieved by only streaming the tiles viewed by the user.

III. System Setup

In this section, we first introduce the system model and the network architecture, and then we discuss 360^o video encoding into multiple quality layers and tiles. Finally, we present the employed viewport prediction algorithm.

1) Wireless cellular network: In this paper, we consider a heterogeneous cellular network (HCN), like the one depicted in Fig. 2. The network consists of Small Base Stations, i.e., microcells, and a Macro-cell Base Station (MBS). Let $\mathcal{N} = \{1, \dots n, \dots, N\}$ denote the set of the N SBSs, and $N+1$ represent the MBS. For notational convenience, we also define the augmented set $\mathcal{N}_B = \mathcal{N} \cup N+1$ that includes the SBSs along with the MBS. The MBS is connected to the core network through a high capacity backhaul link, i.e., optical fiber, while the SBSs are connected to the MBS through wireless millimeter-wave links.

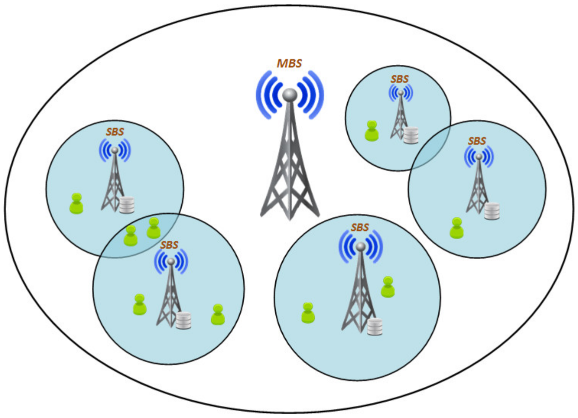


Fig. 2. Considered network architecture.

Let p_n be the communication range of the nth SBS and $\mathcal{P} = \{p_1, \dots p_n, \dots, p_N\}$ be the set that contains the communication ranges of all SBS. The communication range of the MBS is p_{N+1}, and is assumed to be large enough so that the MBS can communicate with all SBSs. Each SBS $n \in \mathcal{N}$ has a cache capacity $C_n \geq 0$, $\forall n \in \mathcal{N}$ where popular content can be cached. We further assume that there are U users forming the set $\mathcal{U} = \{1, \dots u, \dots, U\}$. Since some users may be located in the overlap of the coverage areas of multiple SBSs, these users are assigned to the SBS with the maximum signal-to-interference-plus-noise ratio (SINR).

2) Video Library: We assume that users request 360^o video files from a content catalogue of $V = |\mathcal{V}|$ files, with $\mathcal{V} = \{1, \dots v, \dots, V\}$ being the set of the 360^o videos. Each 360^o video is encoded into G Group of Pictures (GOPs) that form the set $\mathcal{G} = \{1, \dots, g, \dots, G\}$. Each GOP is encoded into L quality layers forming the set $\mathcal{L} = \{1, \dots, L\}$, and M tiles forming the set $\mathcal{M} = \{1 \dots, M\}$. For each tile, the first quality layer is known as the base layer, while the rest $L-1$ layers are called enhancement layers. The acquisition of the base layer of a tile offers reconstruction of that tile at the lowest available quality, while the acquisition of all the layers of a tile up to the lth gradually improves the reconstruction quality of that tile. For each GOP, in order to satisfy a user demand for a requested viewport at a certain quality, the user has to acquire the base layer for all the tiles of the video along with all the enhancement layers corresponding to the demanded quality for the tiles that form the requested viewport.

3) Viewport Prediction: A critical component of 360^o video streaming is the Viewport Prediction (VP) [31]–[33]. The aim of VP is to predict the requested viewport by a user in the *near* future (e.g., 1-2 sec), and prefetch it to the user. This is essential to provide smooth playback, as SBSs are not able to respond instantly to the user head movements due to the end-to-end delay.

VP can be done by observing the most recently requested frames by a user. These past requests are used to forecast the viewport that will be requested in the next few seconds. Such an approach is examined in [31], [32], where authors use variants of the linear regression algorithm to predict the users' head movements. A more naïve approach is presented in [33], where VP is performed assuming that the users' head orientation will not change in the next 3 seconds.

In our system, to perform viewport prediction, we use the Last Sample Replication (LSR) algorithm [32]. We have selected this algorithm because of its low complexity, however, our framework can be used with any VP algorithm. Based on the LSR, the predicted viewport of the GOP $g+1$ is assumed to be the same as the one that was requested in the GOP g. For the first GOP, without loss of generality, we assume that the predicted viewport is the requested viewport. Although the employment of advanced VP algorithms [34], [35] would further improve the accuracy of the predicted viewports, we do not adopt such algorithms as we aim to show the advantages coming from caching. Further, the employment of more advanced prediction algorithms would increase the complexity of our system. However, the conclusions regarding how tile-encoding, layered encoding and online caching impact 360^o video delivery systems would not be altered by the employed VP algorithm.

4) End-to-end-delay: As we already mentioned, for each GOP $g \in \mathcal{G}$, all the tiles encoded at the base quality along with all the enhancement layers up to the targeted quality for the tiles that form the output viewport of the VP algorithm, need to be prefetched to the users within a specific time window. Failing to deliver these tiles on time would lead to buffer underruns, as the tiles would not be available to the buffer at the time they should be displayed. This would lead to degraded QoE, as tiles that are not delivered on time are discarded. Let us denote by d_n the time needed to transmit one Mbit from the nth SBS to a user, and d_{N+1} the time needed to transmit one Mbit from the backhaul of the MBS to a user. Obviously, $d_{N+1} > d_n$, due to the additional time needed to initially fetch data from the backhaul of the MBS to the SBS. The timely delivery of the tiles of each GOP must respect the following equation:

$$\sum_{n \in \mathcal{N}_B} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} o_{vglm} \cdot d_n \cdot q_{vglm}^{nu} \leq t_{disp}, \forall v \in \mathcal{V}, \forall u \in \mathcal{U}, g \in \mathcal{G} \tag{1}$$

where o_{vglm} is the size of the mth tile encoded in the lth quality layer of the gth GOP of the vth 360^o video. The variable q_{vglm}^{nu} takes the value 1 when the mth encoded tile of the lth quality layer of the gth GOP of the vth 360^o video is delivered to the uth user from the cache of the nth SBS ($n \in \mathcal{N}$) or the MBS ($n = N+1$), and 0 otherwise. The parameter t_{disp} denotes the playback duration of each GOP. This constraint determines whether the tiles of the $(g+1)$th GOP can be prefetched at the user's buffer, as the user is watching the GOP g. We would like to note that under the constraint (1), the employed buffer size is equal to 1 GOP. This is because as the user interacts through the scene in the GOP g, the output of the VP algorithm for the GOP $g+1$ is revealed. In addition, the use of a short buffer limits the bandwidth loss occurring due to users changing the 360^o videos they want to watch.

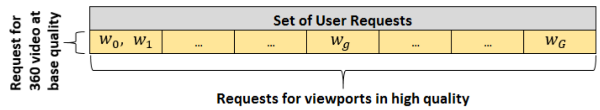


Fig. 3. User requests for a 360^o video.

IV. Users' Requests Model and Cache Update Schedule

In this section, we present the considered users' requests model and the cache update schedule. We assume that for each cached 360^o video, our system caches all the tiles at the base quality layer for all the GOPs, as well as the tiles of a $virtual$ viewport for each GOP in high quality. Recall that a viewport consists of k tiles that form a rectangular area, while a virtual viewport is comprised of the k most popular tiles, which do not necessarily form a rectangular area, as shown in Fig. 1b.

We assume that time is slotted in T time slots, and each time slot has the duration of one GOP. When a user is interested in watching a 360^o video with duration of G GOPs, they should send G consecutive requests,[1] as shown in Fig. 3. The first request is special and comprises a request w_0, which is used by our algorithm in Section VI to predict the popularity of each 360^o video, and a request w_1 for obtaining the viewport for the first GOP. The request w_0 is to acquire the 360^o video at the base quality for all GOPs. As we will show in Section VI this request is used to reduce the size of the optimization problem. Though all the tiles encoded at the base layer are requested at the first time slot, they may be delivered to the users along with the enhancement layer tiles. We assume that the request w_0 occurs at the same time slot with the request w_1. The w_gth request, $g \in \{1, \ldots, G\}$, is to obtain the tiles that comprise the requested viewport, and belong to the gth GOP, in high quality. These requests are used by our algorithm presented in the next sections to calculate the popularity of each tile per GOP. For notational convenience, we denote the ith set of requests $\{w_0^i, w_1^i, \ldots, w_G^i\}$ made by a user for a 360^o video by $\mathcal{W}^i$, while $\mathcal{W} = \cup_i \mathcal{W}^i$ contains all the sets of users' requests. Hereafter, we drop the index of the ith set of requests when is not needed.

The decisions of which tiles of a 360^o video to cache at an SBS and in what quality are made online, i.e., when the content is requested. Specifically, when a user request w_0 arrives at an SBS in the time slot 1, if the tiles of the requested 360^o video at base quality are not cached in it, a decision has to be made regarding whether to cache them. This decision depends on the popularity of the video. If the decision is to cache these tiles, all the tiles of the base layer for all GOPs will start being fetched through the backhaul and cached at the SBS, replacing the tiles of another 360^o video that will be evicted. When the user issues further requests w_g for receiving tiles of the GOPs of the 360^o video in high quality, our system uses the viewport prediction algorithm described in Section III to decide which tiles of the predicted viewport will be fetched from the backhaul so that they can be delivered to the user on time. When the tiles that form the predicted viewport arrive at the SBS, we identify two cases: (a) if the decision for w_0 was not to cache the 360^o video in base quality, the fetched tiles will be delivered to the user but these tiles will not be cached at the SBS, as the video is not popular enough, (b) if the decision for w_0 was to cache the 360^o video in base quality, then for each request $w_g, g \in G$, in case some (or none) of the tiles that form the predicted viewport are cached in high quality a soft cache hit [36] will occur. In such a case, the cached tiles of the predicted viewport will be served to the user directly from the SBS, while the tiles of the predicted viewport that are not cached at the SBS will be fetched to the SBS from a remote content server through the backhaul link of the MBS and be delivered to the user if the end-to-end constraint permits. Then, a decision is made about whether to cache some or all of the tiles that were fetched through the backhaul. The latter decision reflects tiles' popularity of a 360^o video.

The proposed cache optimization algorithm regarding which tiles to cache is presented in the next sections.

[1] When a user wants to stop watching a video, they halt sending requests for the following GOPs.

V. MDP Formulation

In this section, we formulate the problem of caching 360^o videos in cellular networks as a Markov Decision Process [37]. Since in our setting users can download the requested content only from the SBS that they are connected to, each SBS optimizes the cache use and the content replacement strategy independently of each other. Hereafter, following reinforcement learning terminology, SBSs are also called agents.

State Space: In the considered setting, the SBS $n \in \mathcal{N}$ can be in a state $s \in \mathcal{S}$, where $\mathcal{S}$ represents the set of all possible states. Each state is characterized by the features extracted from observations of users' past requests, considering fixed observation windows. Below we describe the features we consider.

The *first feature* has two components that refer to the total number of requests for each cached 360^o video that occurred in: a) a short-term window of H_s sets of user requests (see Fig. 3), and b) a long-term window of H_l sets of user requests. This feature associated with the cache of SBS $n \in \mathcal{N}$ can be described by the vector $\mathbf{x}^n = [\mathbf{x}_s^n \ \mathbf{x}_l^n]$ with $\mathbf{x}_f^n = [x_{f,i}^n]$, $\forall f \in \{s, l\}$, $\forall i \in 1, \ldots, C$, and $x_{f,i}^n \in \{1, \ldots, H_f\}$. $x_{f,i}^n$ refers to the total number of times the video in the ith cache position was requested (either in short-term or long-term). Thus, the feature space $\mathcal{X}_f^n$ is given by $\{1, \ldots, H_f\}^C$ and the overall feature space is $\mathcal{X}^n = \mathcal{X}_s^n \times \mathcal{X}_l^n$. Recall that, C is the cache capacity of the SBS. It is worth noting that the above definition of features reduces the feature space drastically, as features are computed for all the tiles (cached videos) in base quality instead of each tile in base quality independently.

Similarly, the *second feature* has two components that correspond to the total number of requests for tiles in high quality of the cached 360^o videos that happened during: a) the short-term window of H_s sets of user requests, and b) the long-term window of H_l sets of user requests. This feature is associated with the cache of SBS $n \in \mathcal{N}$ and is computed for each cached tile in high quality of GOP g, when request $w_g,\ g > 0$ is processed. Let the vector $\mathbf{y}^n = [\mathbf{y}_s^n\ \mathbf{y}_l^n]$ describe this feature, where $\mathbf{y}_f^n = [y_{f,i,j}^n]$, $\forall f \in \{s, l\}$, $\forall j \in \{1, \dots, k\}$, $\forall i \in \{1, \dots, C\}$ and $y_{f,i,j}^n \in \{1, \dots, H_f\}$. $y_{f,i,j}^n$ denotes the number of times the jth tile of the ith cached 360^o video was requested at the nth SBS. Thus, the feature space $\mathcal{Y}_f^n$ is given by $\{1, \dots, H_f\}^{kC}$ and the overall feature space for the cache space at the nth SBS is given by $\mathcal{Y}^n = \mathcal{Y}_s^n \times \mathcal{Y}_l^n$.

Finally, the *third feature* has two components that correspond to the number of times the examined item (tile in high quality or 360^o video in base quality) was requested at the nth SBS: a) in the short-term window of H_s sets of user requests, and b) in the long-term window of H_l sets of user requests. Specifically, when the examined item is a 360^o video in base quality, this feature refers to the total number of times this video was requested at the nth SBS. This is the case when a request w_0 is received. When the examined item is a tile of a 360^o video in high quality, i.e. for requests $w_g,\ g > 0$, the feature corresponds to the total number of times the examined tile was requested. The feature vector is defined as $\mathbf{z}^n = [\mathbf{z}_s^n\ \mathbf{z}_l^n]$ with $\mathbf{z}_f^n = [z_f^n]$, $\forall f \in \{s, l\}$, and $\forall z_f^n \in \{1, \dots, H_f\}$. z_f^n stands for the total number of times the item (tile in high quality or 360^o video in base quality) was requested. Thus, the feature space $\mathcal{Z}_f^n$ is given by $\{1, \dots, H_f\}$ and the overall feature space for the examined item is $\mathcal{Z}^n = \mathcal{Z}_s^n \times \mathcal{Z}_l^n$.

Following the above definitions of the features, the overall state space is given by:

$$\mathcal{S}^n = \mathcal{X}^n \times \mathcal{Y}^n \times \mathcal{Z}^n. \tag{2}$$

Hereafter, we drop the superscript of the state space and use $\mathcal{S}$ as each SBS makes decisions independently of each other.

Action Space: As we mentioned in Section IV, users' requests w_0 correspond to a request for a 360^o video in base quality for all the GOPs of this video, while requests $w_g \in \mathcal{W}$ with $g \in 1, \dots, G$ stand for a request for a viewport of the gth GOP encoded in high quality.

When an SBS receives a request from a user, there are three possible cases regarding what data is cached at the SBS: a) no data for the requested 360^o video is cached, b) the 360^o video is cached at the base quality and the predicted viewport is cached at high quality and, c) the 360^o video is cached at the base quality, but a different viewport is cached at high quality.

In case a user requests a 360^o video that is not cached at the SBS, this has to be fetched through the backhaul and be delivered to the user. Fetching content through the backhaul adds cost to the network operator and increases the delay experienced by the users. When no data of a 360^o video are cached at the SBS, a user request $w_g \in \mathcal{W}$ with $g \in \{0, \dots, G\}$ is processed as follows. To accommodate a request w_0, all the tiles of the requested 360^o video encoded at the base quality will start being fetched through the backhaul and delivered to the user for all the GOPs. For each of the following requests w_g with $g \in \{1, \dots, G\}$, all the tiles of the viewport indicated by the prediction algorithm in Section III encoded in high quality will be fetched through the backhaul in high quality, and be delivered to the user. Therefore, when the requested 360^o video is not cached at the SBS, there are two types of possible actions: a) to leave the cached content at the SBS unchanged, or b) to evict the tiles of a cached 360^o video from the cache of the SBS and replace them with the tiles of the requested one. Thus, there are $C + 1$ possible actions. Let the set $\mathcal{A}_1 = \{A_{10}, A_{11} \dots, A_{1i}, \dots, A_{1C}\}$ denotes all the possible actions when a video is not cached at the SBS. A_{10} stands for the case the cached content at the SBS is left unchanged, and A_{1i} means that all the tiles of the ith cached video at the SBS will be replaced by the corresponding tiles of the requested 360^o video.

If both the requested 360^o video encoded in the base quality and the tiles that form the predicted viewport for the examined GOP, e.g., $g \in \{1, \dots, G\}$ encoded in high quality are cached at the SBS, the request w_g will be served from the cache, and no action will be taken. Then, a decision regarding whether to cache the tiles of the predicted viewport for the next GOP, i.e., $g+1$, is made. This happens because our scheme employs the LSR algorithm, as we described in Section III-3.

Finally, if the 360^o video is cached at the base quality, but a different viewport than the predicted one is cached at the SBS at high quality, the requested tiles that are not cached have to be fetched through the backhaul, and then be served to the user. In that case, the possible actions are the following: a) to leave the cached viewport unchanged, or b) to cache some of the tiles, which were not part of the predicted viewport, and were fetched through the backhaul. To limit the action space, we assume that each action concerns only one tile that may be updated at the SBS cache. In this way, the agent takes sequentially actions for all tiles that were fetched through the backhaul in terms of whether to cache them at the SBS or not. This process is repeated until a decision is made for all the fetched tiles. Since each viewport consists of k tiles, the possible actions for a tile form the set $\mathcal{A}_2 = \{A_{20}, A_{21} \dots, A_{2j}, \dots, A_{2k}\}$. The action A_{20} denotes the case where the cached content is left unchanged, while the action A_{2j} corresponds to the case where the candidate tile will replace the jth tile in high quality of the requested 360^o video that was cached at the SBS. We consider that a GOP is fully processed when a decision has been made for all the tiles that were fetched through the backhaul. After completing the sequential decisions, the cached virtual viewport for the considered video is updated. Next, the subsequent GOP is processed in a similar way. We would like to note that the use of virtual viewports and the decomposition of actions on per tile basis permits to greatly reduce the action space as otherwise, the action space would have been comprised of all possible viewports.

Considering the above, the overall action space $\mathcal{A}$ is defined as:

$$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2. \tag{3}$$

Reward:

We define the reward of each action to be the average distortion reduction the users will experience in the next H sets of users' requests. Thus, given a state $s \in \mathcal{S}$, the reward of taking action $a \in \mathcal{A}$ is calculated as:

$$r(s,a) = \frac{1}{H} \sum_{h \in \mathcal{H}} \sum_{v \in \mathcal{V}} \sum_{g \in \mathcal{G}} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} \mathbb{1}(\phi_{h,v,g,l,m}) \cdot \delta_{v,g,l,m} \tag{4}$$

When we process the ith set of requests $\mathcal{W}^i$, the set $\mathcal{H} = \{\mathcal{W}^{i+1}, \ldots, \mathcal{W}^{i+h}, \ldots, \mathcal{W}^{i+H}\}$ contains the next H sets of user requests. In our formulation, the reward in (4) is obtained after the next H sets of user requests have occurred [38]. The term $\phi_{h,v,g,l,m}$ represents the mth tile of the lth quality layer of the gth GOP of the vth 360^o video of the $\mathcal{W}^{i+h}$th set of user requests. The term $\delta_{v,g,l,m}$ denotes the distortion reduction achieved by obtaining the corresponding tile. The indicator function $\mathbb{1}(\phi_{h,v,g,l,m})$ in (4) is defined as:

$$\mathbb{1}(\phi_{h,v,g,l,m}) = \begin{cases} 1, & \text{if } \phi_{h,v,g,l,m} \text{ can be delivered} \\ & \text{on time for } \mathcal{W}^{i+h}. \\ 0, & \text{if } \phi_{h,v,g,l,m} \text{ cannot be delivered} \\ & \text{on time for } \mathcal{W}^{i+h}. \end{cases}$$

Optimization Problem:

In order to quantify how good a particular state s is, we estimate the value function. This function corresponds to the expected discounted reward of policy π when starting from a state s and then following this policy. The value function is formally expressed as:

$$V_\pi(s) = E_\pi[G_\tau | S_\tau = s] = E_\pi[\sum_{\kappa=0}^{\infty} \gamma^\tau R_{\tau+\kappa+1} | S_\tau = s] \tag{5}$$

where G_τ, R_τ, and S_τ are the expected reward, the immediate reward and the state at time τ, respectively. The parameter $0 \leq \gamma \leq 1$ is called discount rate and gradually discounts the effect of an action to future rewards. If $\gamma = 0$, the agent is "myopic" and maximizes the immediate reward. As γ approaches 1, the objective takes into account future rewards more strongly, and the agent becomes farsighted. The above equation can be rewritten as a Bellman equation [39] as follows:

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V_\pi(s')] \tag{6}$$

where $p(s',r|s,a)$ is the transition probability from the state s to the state s' by taking the action a with a reward r.

VI. DQN BASED CACHE OPTIMIZATION

The main challenge to solve (6) is the requirement to know the transition probabilities $p(s',r|s,a)$. For the studied problem, continuous computation of the transition probability matrix is necessary because of the non-stationary requests' dynamics, which is computationally demanding. To overcome this problem, we can adopt the Q-learning algorithm [9], which learns the optimal policy through interaction with the environment. Q-learning uses the $Q(s,a)$ values instead of using the value function in (6). These values reflect how "good" is to take action a when in state s. Similarly, $Q_\pi(s,a)$ represents how good it is to take action a when starting from state s, and thereafter follow the policy π. This is defined as follows:

$$Q_\pi(s,a) = E_\pi[\sum_{k=0}^{\infty} \gamma^\tau R_{\tau+k+1} | S_\tau = s, A_\tau = a] \tag{7}$$

where A_τ is the action at time τ.

The optimal policy is the one that maximizes the expected reward for all states and is given by:

$$\pi^\star(s) = \arg\max_{a \in \mathcal{A}} (Q(s,a)), s \in \mathcal{S} \tag{8}$$

To determine the optimal policy $\pi^\star(s)$, the Q-learning algorithm updates the $Q(s,a)$ values iteratively. Specifically, the $Q(s,a)$ values are updated according to the formula:

$$Q(s_\tau, a_\tau) = (1 - \alpha_\tau) Q(s_\tau, a_\tau) + \alpha_\tau [R_\tau + \gamma \max_{a \in \mathcal{A}} Q(s_{\tau+1}, a)] \tag{9}$$

where α_τ is the learning rate at time τ. The learning rate corresponds to the rate at which newly acquired information overrides old one.

Q-learning can select actions using policies such as the ϵ-greedy, where $\epsilon \in [0,1]$, which ensures that random actions are always explored and overfitting is avoided. According to ϵ-greedy policy, the action resulting in the maximum $Q(s_\tau, a_\tau)$ value is selected with probability $1 - \epsilon$, and a random action is selected with probability ϵ. The Q-learning algorithm is guaranteed to converge to the optimal solution [40] when all the state-action pairs are visited infinitely often, and the learning rate α_τ satisfies the following conditions:

$$\sum_{\tau=0}^{\infty} \alpha_\tau(s,a) = \infty \quad \text{and} \quad \sum_{\tau=0}^{\infty} \alpha_\tau^2(s,a) < \infty, \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \tag{10}$$

The Q-learning algorithm is an efficient method to determine the optimal policy when the state-action space is small. However, when the state-action space grows, the lookup table where the $Q(s,a)$ values are stored becomes prohibitively large. To overcome this drawback of Q-learning, we employ a Deep Reinforcement Learning (DRL) [10] approach. Using DLR the $Q(s,a)$ values are approximated by a Deep Neural Network (DNN). The DRL framework consists of two phases: a) the offline phase where the DNN is trained, and b) the online phase during which the actual caching decisions are made.

During the offline phase, the DNN is initially built by selecting some random weights θ. Then, the DNN is trained with a number of historic transition profiles, as in [41]. These profiles correspond to request patterns experienced in the past, and are available to network operators, as they maintain statistics of past requests. The training of the DNN is performed in a mini-batch manner. Specifically, at each training epoch, a sample of the transition profiles and their estimated Q values are obtained by randomly sampling the experience replay memory D, which has capacity N_D. This mechanism is used to remove the correlations between observations, while the transitions between the states become more independent and identically distributed. We would like to note that while the videos and tiles' popularity changes over time, our system uses the same model, i.e., employs the same DNN for all GOPs. This makes the DNN able to generalize between similar states and take actions in terms of which content to cache at the SBSs. Hence, the number of GOPs does not have an impact on the complexity of our model.

To stabilize DNN training, apart from the experience replay, we use the mechanism of the fixed target network [40]. According to this mechanism, a second DNN is employed,

Algorithm 1 DRL Framework

Offline Phase
Initialize the evaluation network with weights θ
Initialize the fixed target network with weights θ'
Initialize the experience buffer D with capacity N_D
Initialize a random exploration process
Train the DNN with features (s, a) and outcomes $Q(s, a)$ in a mini-batch manner
Online Phase
for each time slot **do**
  for each user request in a time slot **do**
    for each candidate item of a user request **do**
      Receive observation s_τ
      if the candidate item is not cached at the SBS **then**
        With probability $1 - \epsilon$ **select**
          $a_\tau = \arg\max_{a \in \mathcal{A}} Q(s_\tau, a, \theta)$
        Otherwise,
          $a_\tau \leftarrow$ random action
        Take action a_τ and observe r_τ, $s_{\tau+1}$
        Store the tuple $(s_\tau, a_\tau, r_\tau, s_{\tau+1})$ in the experience replay buffer D
      end if
      Update cache hit ratio
      Update Feature Space
      if Modulo(w, $\mathcal{N}_B$)==0 **then**
        Sample $\mathcal{M}_B$ tuples from D
        Update DNN by minimizing $Loss(\theta)$ in (11)
        Update fixed target network weights
      end if
    end for
  end for
end for

which is called fixed-target network. This network has the same architecture as the original DNN that is used for the function approximation (evaluation network). Not using a separate network to estimate the target Q values would lead to destabilization. This would happen because as the Q values (output of the evaluation network) are updated towards the target values (calculated by (9)), the target values will also be updated in the same direction. To overcome this problem, the weight parameters of the target network are kept fixed and are copied from the evaluation network only every N_T steps. Thus, using a second network to estimate the target Q-values leads to a more stable training, since the Q-values obtained from the evaluation network are updated towards a target that is kept fixed (for a number of steps).

When the offline phase is completed, the obtained weights θ are used to initialize the DNN in the online phase. During this phase, if the candidate item (360^o video in base quality or tile in high quality) is not cached at an SBS, the agent takes an action according to the ϵ-greedy policy (i.e., it decides whether to cache the item or not and what content will be replaced), and then proceeds to the next state. In this way, new actions are always explored, and cached content whose popularity the algorithm overestimated in the past will not stay in the cache forever. After the execution of each action, the tuple $(s_\tau, a_\tau, r_\tau, s_{\tau+1})$ is stored in the experience replay buffer D, in order to be used later for the training of the DNN.

In the online phase, the DNN is trained in a similar way to the offline phase, where a batch of $\mathcal{M}_B$ transition profiles is randomly sampled from the experience replay memory D every $\mathcal{N}_B$ steps. The DNN is trained towards the target Q values using the back-propagation method, by minimizing the loss function $Loss(\theta)$. The loss function is given by:

$$Loss(\theta) = \frac{1}{\mathcal{M}_B} \sum_{i \in \mathcal{M}_B} (y_i - Q(s_i, a_i, \theta))^2 \qquad (11)$$

where $y_i = r_i + \max_{a'_i} Q(s'_i, a'_i, \theta_i^-)$ represents the target Q value of the ith sample, and $\theta_i^- = \theta_{i-\mathcal{N}_\mathcal{B}}$. This loss function is used in both offline and online phases. We would like to note that the continuous update of our DNN in the online phase helps our model to keep track of gradual changes that may occur in the consumption patterns of the users' requests. This is because, in general, network operators have fallback mechanisms to account for potential changes in the popularity model, as this affects the provided quality of service.

The overall DRL framework is presented in Algorithm 1.

VII. Performance Evaluation

In this section, we examine the performance of the proposed DQN-based online caching algorithm for 360^o videos in cellular networks. First, we describe the schemes under comparison and provide the simulation setup. Next, we show the convergence of the loss function during the training of the DNN. Then, we analyze the impact of various system parameters on the performance of the system. Finally, we demonstrate how the viewports' popularity shapes the popularity of each tile.

A. Simulation Setup

Let us describe the main characteristics of the schemes under comparison and the proposed scheme:

1) *Least Frequently Used (LFU):* In this scheme, the network operator keeps track of the number of requests that occurred for each cached 360^o video. When a user request arrives at an SBS, then: a) if the requested 360^o video is not cached at it, all the tiles of the 360^o video that was requested the least number of times will be evicted from the cache of the SBS. Then, for all the GOPs, all the tiles of the requested 360^o video encoded at the base layer along with the tiles of the predicted viewport in high quality will be cached at the SBS; b) if the 360^o video is already cached at the base quality for all the GOPs, but some of the cached tiles in high quality are different from the ones that belong to the predicted viewport, these

tiles will be evicted and be replaced by the tiles of the predicted viewport.

2) *Least Recently Used (LRU):* In this scheme, the network operator keeps track of how recent are the requests that occurred for each cached 360^o video. When a user request happens at an SBS, then: a) if the requested 360^o video is not cached at the SBS, all the tiles of the 360^o video that were requested the least recently will be evicted from the SBS cache. Next, all the tiles of the requested 360^o video will be cached at the SBS at the base quality for all GOPs along with the tiles of the predicted viewport in high quality; b) if the 360^o video is cached at the SBS, for each GOP, if some of the cached tiles in high quality are different from the ones of the predicted viewport, these tiles will be replaced by the corresponding tiles of the predicted viewport.
3) *First In First Out (FIFO):* In this scheme, the network operator keeps track of when the requests for each cached 360^o video occurred. When a user request arrives at an SBS, then: a) if the requested 360^o video is not cached at the SBS, all the tiles of the 360^o video that was cached the earliest will be evicted from the SBS. Then, for all GOPs, all the tiles of the requested 360^o video encoded at the base layer, along with, for each GOP, the tiles of the predicted viewport in high quality will be cached at the SBS in the place of the evicted tiles; b) if the 360^o video is cached at the SBS, then for each GOP, if some of the cached tiles in high quality are different from the ones forming the predicted viewport, these tiles will be evicted, and be replaced by the tiles of the predicted viewport.
4) *Proposed Scheme:* In the proposed scheme, the caching decisions are made exploiting observations derived from past users' requests. This scheme employs the DQN algorithm presented in Section VI to decide on the cache updates. For each cached 360^o video, all the tiles at the base quality along with the most popular tiles in high quality that form a virtual viewport, are cached at the SBS for all the GOPs.

We selected LFU, LRU, and FIFO policies as comparison schemes as they are common cache updates methods used for standard videos in the literature. Similar to the proposed scheme, the schemes under comparison cache per video the same number of tiles. We advanced the performance of these schemes by considering that users who want to receive 360^o videos are interested in viewing only a viewport. To the best of our knowledge, there are no sophisticated online caching policies for 360^o videos that take into account that users are interested in viewing only a viewport.

For the sake of simplicity, all the conducted experiments are done assuming a single SBS and an MBS. This does not affect the derived conclusions, as SBSs make caching decisions independently of each other. As we have already mentioned in Section III, although SBSs' coverage area may overlap, users are assigned to a single SBS, i.e., the one with the maximum SINR. The exploitation of opportunities arising because of the overlapped coverage areas is part of our future work. The exploitation of users association with multiple SBSs would require changes in our framework, which would increase the complexity. This problem can be addressed using multi-agent concepts [42], but the main challenge would be how to deal with the tight delivery deadlines. We would like to emphasize that our algorithm can be applied to networks with an arbitrary number of SBSs. This is because as each user is assigned to a single SBS, our algorithm can run in parallel for each SBS. The coverage range of the SBS is set to be $P_n = 300$m, while the coverage range of the MBS is $P_{N+1} = 2000$m, and is large enough to permit the communication with the SBS. The delay needed to obtain one Mbit from the SBS is $d_n = 1/14$ sec/Mbit, while the delay to deliver one Mbit from the backhaul of the MBS to the user is $d_{N+1} = 1/2.9$ sec/Mbit. The cache capacity of the SBS is set to be enough to store 10% of the 360^o videos of the content library. The number of users is $U = 200$ who are randomly placed in the coverage area of the SBSs. Recall that, when a 360^o video is cached at the SBS, this means that for each GOP, all the tiles are cached at the base quality, and the tiles that form a virtual viewport are cached in high quality.

The content library contains $V = 500$ videos, while each video is encoded in 30 GOPs. The duration of each GOP is assumed to be $t_{disp} = 1$ sec. Each GOP is encoded into $M = 12$ tiles, where each tile is encoded into $L = 2$ quality layers. The bitrate of the base layer is 2 Mbps, while the bitrate of the enhancement layer is 12 Mbps. The size of each viewport consists of 4 tiles, while the available viewports are the ones depicted in Fig. 4. The distortion reduction achieved by obtaining a tile at the base quality layer is 30 dB, while the distortion reduction achieved by receiving a tile at the enhancement quality layer is 10 dB. The probability of a 360^o video to be requested from a user follows the Zipfian distribution [43], as it is common to the literature. The shape parameter of the Zipfian distribution is set to $\eta_v = 1$. The probability of a 360^o video $v \in \mathcal{V}$ to be selected under the Zipfian distribution is given by:

$$p_v = \frac{1/v^{\eta_v}}{\sum_{v \in \mathcal{V}} 1/v^{\eta_v}}.$$

We consider realistic navigation patterns, extracted from the dataset in [11], from which we sampled 200 trajectories of head movements. These trajectories are obtained from 10 different videos, where for each video, we sampled 20 different trajectories. With equal probability, we mapped the index of each one of the $V = 500$ videos from the content library to one of the 10 sampled videos of the dataset. Then, for each of the $V = 500$ videos of the content library, according to its mapped index, we selected one of the 20 available trajectories uniformly at random. We used Zipf distribution with different shape parameters in the range [0.8, 2.5] for generation of the training and testing sets.

We assume that the total number of sets of users' requests is $W = 10000$. The short-term time window refers to $H_s = 300$ sets of user requests,[2] while the long-term time window corresponds to $H_l = 1000$ sets of user requests. The reward in (4) is calculated for the next $H = 1000$ sets of user requests.

[2]Each set of requests corresponds to the tiles of a single video demanded by a user.

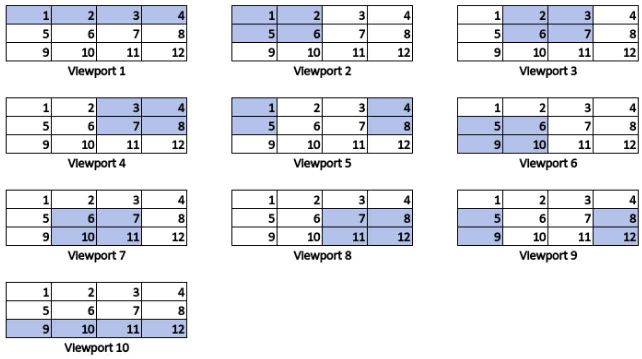


Fig. 4. Considered set of viewports. The light blue area highlights the area covered by the viewports.

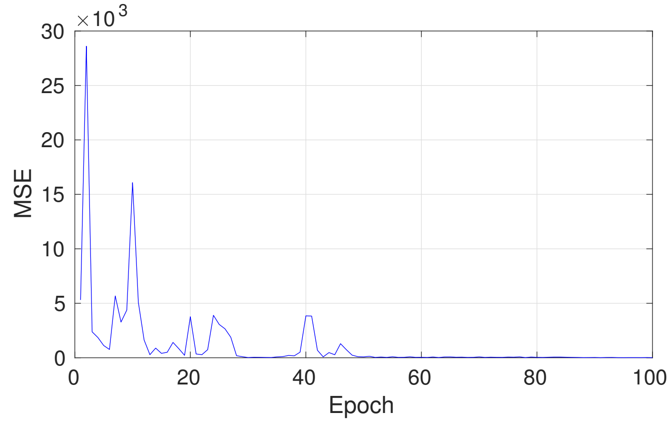


Fig. 5. MSE of the loss function with respect to the training epochs.

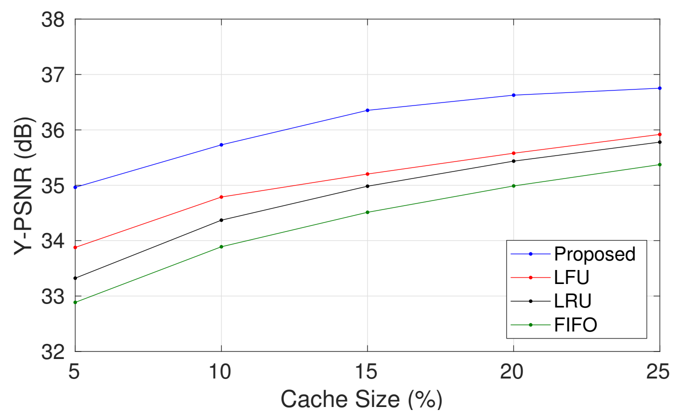


Fig. 6. Y-PSNR of the rendered viewports with respect to the cache size for all the schemes under comparison.

B. Deep Neural Network Training

We consider a Deep Neural Network (DNN), which consists of four fully connected layers, i.e., the input layer, two hidden layers, and the output layer. As the cache capacity of our system is C, the input layer consists of $10C + 2$ nodes that reflect the vector size of each state. The hidden layers and the output layer consist of $5C + 1$ nodes, as there are $5C + 1$ total actions. The activation function of the hidden layers is the "ReLu", while the activation function of the output layer is the "linear" function. The loss function for the training of the DNN in the offline phase is the same as the one that is used in the online phase, and is given in (11). The DNN is trained with the *Adam* optimizer. The DNN is trained for 100 epochs with historic transition profiles, as explained in Section VI in order to become sufficiently accurate. The transition profiles were generated following the Zipfian distribution, where the shape parameters were varied in the range [0.8, 2.5] to diversify the users' requests. The learning rate is set to be $\alpha = 0.001$, while the ϵ-greedy parameter is set to $\epsilon = 0.05$. The discount factor is set to be $\gamma = 0.6$. The experience replay buffer is set to be $D = 2000$, while the mini-batch size is set to $\mathcal{M}_B = 32$. The mini-batch samples are obtained every $\mathcal{N}_{\mathcal{B}} = 200$ requests. The convergence of the loss function during the training phase for the basic scenario is presented in Fig. 5. When the DNN is trained with different system settings than the ones of the basic scenario, a similar convergence behavior is noticed. We would like to note that our employed DNN is trained using a variety of consumption patterns, while the environment in the online phase is considered unknown. These patterns are available to network operators through collecting statistics regarding consumed data. These statistics are obtained by analyzing consumption patterns that occurred over previous years. In the online phase, changes in the popularity model happen gradually, and this is why our model can learn them fast with a small loss.

C. System Parameter Analysis

1) Cache Size: First, we examine the impact of the cache size on the overall quality of the rendered viewports. To this aim, we vary the cache capacity C in the range [5, 25]% of the size of the content library. As we can see in Fig. 6, the proposed scheme outperforms the LFU, LRU and FIFO schemes, for all cache sizes. In particular, for typical cache capacity sizes, i.e., [5-10]%, the performance gap between the proposed scheme and the LFU, the LRU and the FIFO is about 1 dB, 1.5 dB, and 2 dB, respectively. This is because the proposed scheme achieves a better cache hit ratio, as shown in Fig. 7. The increased cache hit ratio of the proposed scheme is attributed to the use of the DQN that learns from the experience of the past observations, which content should be cached. In addition, unlike LFU, LRU and FIFO, where the cached tiles in high quality correspond to actual viewports, in the proposed algorithm, the tiles that will be cached for each 360^o video in high quality correspond to virtual viewports. This provides us with greater flexibility to decide the cached tiles. The effect of the increased cache hit ratio on the quality of the rendered viewports comes from the fact that the tiles that are delivered from the cache of the SBS to the users are delivered with a smaller delay. Hence, more tiles are delivered in total to the users under the considered tight time constraints. When the cache capacity is large, i.e., 25%, the performance gap between the proposed algorithm and the LFU, the LRU and the FIFO schemes closes to about 0.8 dB, 1 dB and 1.4 dB, respectively. This happens because as the cache capacity becomes larger, most of the popular content is stored in the SBS cache for all the schemes.

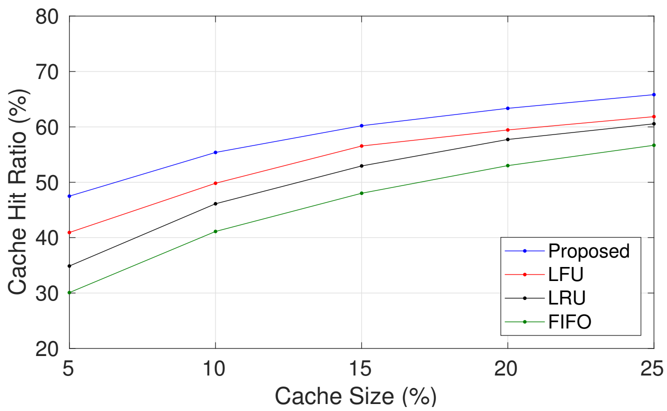


Fig. 7. Cache Hit Ratio with respect to the cache size for all the schemes under comparison.

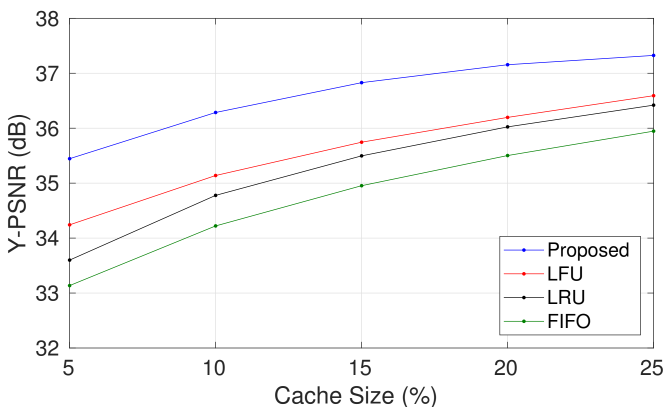


Fig. 8. Y-PSNR of the rendered viewports with respect to the cache size assuming perfect VP for all the schemes under comparison.

To understand the impact of the prediction scheme, we examine in Fig. 8 the impact of the cache size on the overall quality of the rendered viewports when perfect viewport prediction is assumed for the same setting, as in Fig. 6. By comparing Figs. 6 and 8, we can observe that the quality of the rendered viewports in case of perfect VP improves for all the schemes under comparison. Further, we can note that the comparative performance among all schemes is similar in both cases, regardless of the accuracy of the VP mechanism. We would like to emphasize that assuming perfect prediction is an oracle and is used here only as a benchmark. Overall, online caching offers great performance gains because more tiles are delivered to the users under tight delivery constraints.

2) Video popularity distribution: In Fig. 9, we analyze the impact of the skewness parameter of the Zipfian distribution, which characterizes the 360^o video popularity. Specifically, we alter the shape parameter η_v in the range [0.8, 1.6] and measure the overall quality of the rendered viewports for all the schemes under comparison. We note that an increase in the value of the Zipf shape parameter η_v leads to an increase in the overall rendered quality for all the schemes. This is because bigger values of η_v mean that the video popularity distribution gets steeper, i.e., a smaller number of 360^o videos is popular, which increases the efficiency of the cache utilization. We can further observe that as the users' requests concern a

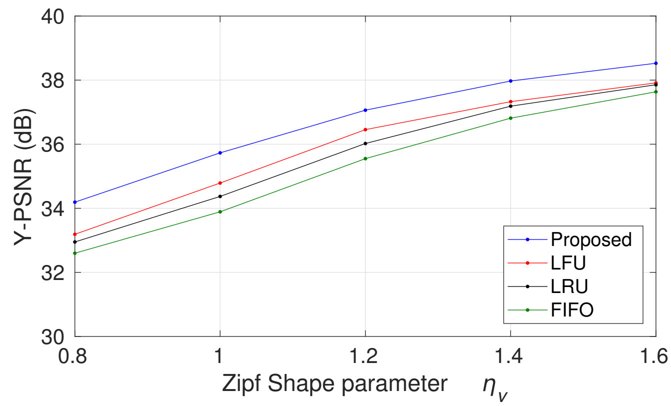


Fig. 9. Y-PSNR of the rendered viewports with respect to the Zipf shape parameter of the 360^o videos for all schemes under comparison.

smaller number of videos (big η_v values), the performance gap between the proposed algorithm and the LFU, the LRU, and the FIFO schemes decreases. For example, as the skewness parameter changes from 0.8 to 1.6, the performance gap between the proposed algorithm and the LFU decreases from ~ 1 dB to ~ 0.6 dB. This is attributed to the fact that as a smaller number of 360^o videos becomes popular, most of these videos will be cached at the SBS for all the schemes.

3) Viewports' popularity distribution: Besides video popularity, we examine the impact of viewports' popularity. We first assume that the viewports' popularity follows a Zipfian distribution with skewness parameter η_p. To analyze the impact of the skewness parameter on the quality of the rendered viewports, we vary the shape parameter η_p in the range [0.5, 2.5]. The performance of the schemes under comparison is depicted in Fig. 10. From the results, we can note that an increase of the skewness parameter η_p leads to an increase in the overall quality of the rendered viewports for all the examined schemes. This is because as the parameter η_p increases, the user requests for the various parts of the 360^o video scenes become less diverse. Thus, the cache effectiveness is improved. In addition, as the skewness parameter changes from 0.5 to 2.5, the performance gap between the proposed algorithm and the LFU increases from about 0.5 dB to about 0.65 dB, respectively. This is because unlike LFU, in the proposed algorithm, the caching decisions for the tiles that will be cached in high quality are made for virtual viewports, which offers increased flexibility in the caching decisions regarding which tiles to cache. Thus, as the requests for the various viewports become less diverse, the performance gains in the proposed algorithm increase. Similar conclusions can be drawn by comparing the proposed scheme with the LRU and FIFO schemes.

In Fig. 11, we evaluate the cache hit ratio of the proposed scheme for: a) our basic scenario where the requests for the viewports are according to the dataset [11], b) the case where the requests for the viewports follow the Zipfian distribution while the shape parameter η_p takes a value from the range [0.5, 1.5], and c) the case where all the user requests are for one viewport, which we term as "Selective". To this aim, we vary the cache size from 5% to 15% of the content library. As

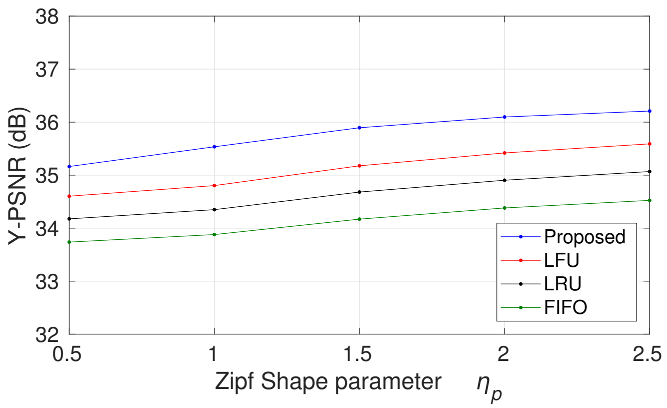


Fig. 10. Y-PSNR of the rendered viewports with respect to the Zipf shape parameter of the viewports for all schemes under comparison.

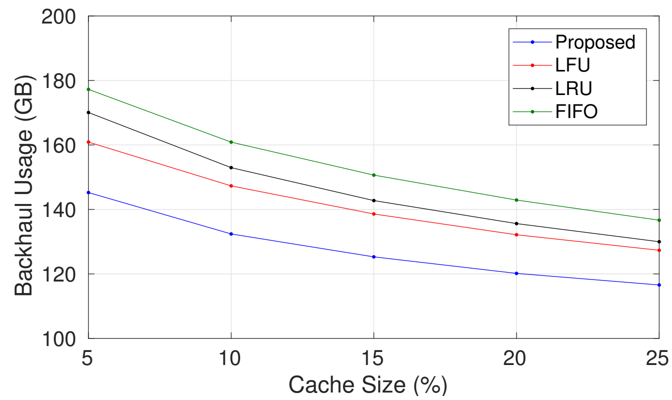


Fig. 12. Backhaul usage with respect to the Cache Size for all schemes under comparison.

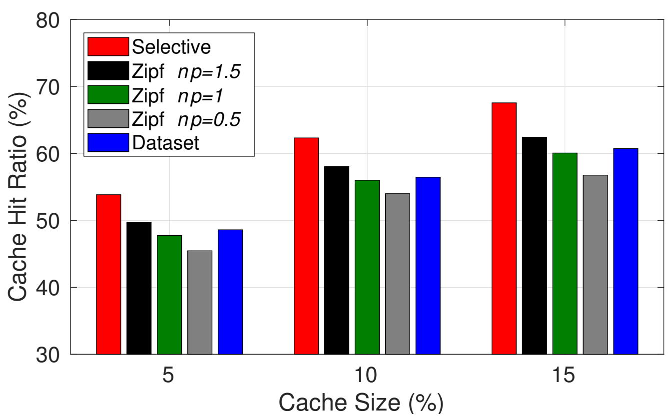


Fig. 11. Cache hit ratio with respect to the cache size for the proposed scheme considering different viewport popularity distributions.

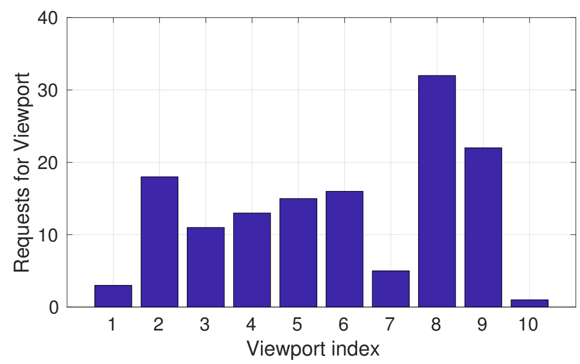


Fig. 13. Total amount of requests for each one of the available viewports.

we can observe, the "Selective" distribution achieves a better cache hit ratio in all cases. This is expected, as when the viewports follow either the dataset or the Zipfian distribution, the requests for the viewports are diverse, while in case of the Selective distribution, all requests are for one viewport. In addition, the cache hit ratio is better when the skewness parameter is higher as described above, while the performance of the dataset, is comparable with the case when the skewness parameter is $\eta_p = 1$.

4) Backhaul Usage: In Fig. 12, we compare the performance of all the schemes under comparison in terms of the backhaul usage. This is a very important performance indicator of the caching schemes since field trials [44] have shown that by reducing the backhaul usage, the network service cost is also reduced. To this end, we vary the cache size in the range [5, 25]% of the content library and measure the backhaul usage, in terms of the bandwidth that should be communicated to satisfy the demands. As expected, an increase in the cache size leads to a decrease in the backhaul usage for all cases. This is because as the cache size increases, more videos will be able to be stored at the SBS cache, thus, more content will be served locally to the users. In addition, we can note that as the cache size increases, the performance gap between the proposed method and the other schemes under comparison decreases. Specifically, as the cache size increases from 5% to 25%, the performance gap between the proposed method and the LFU decreases from about 15.6 GB to approximately 10.7 GB. This is because as the cache size increases, most of the requested content will be able to be cached at the SBS, and thus, the effectiveness of the caching improves for all schemes.

D. Overlap between Viewports

In this section, we present how the overlap between the various viewports shapes the popularity of each tile. To this aim, we examine the popularity of each viewport, along with the popularity of each tile. These popularities are computed by measuring the frequency of occurrence of a request w_g in a window of the previous $H_l = 1000$ sets of user requests. The popularity of each viewport is depicted in Fig. 13 and the popularity of each tile is depicted in Fig. 14. Although the most popular viewport is the viewport 8 (see the viewports illustrated in Fig. 4), by observing the Fig. 14, we can see that the most popular tiles do not correspond to the tiles of that viewport. The overlap between the diverse requests for the various viewports is what determines the popularity of each tile. Thus, by using *virtual* viewports, which consist of the most popular tiles, the most popular tiles can be cached at the SBS. This results in higher cache hit ratio and better quality for the rendered viewports.

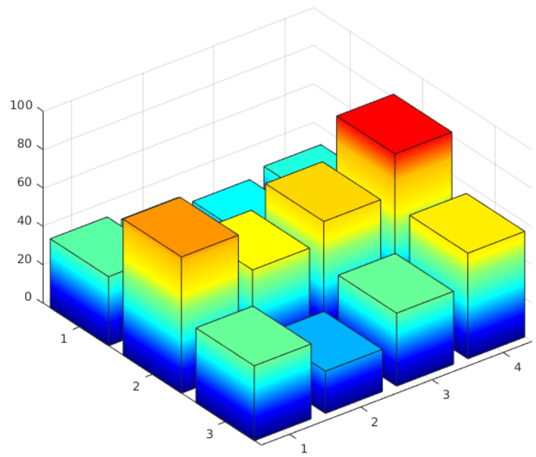


Fig. 14. Total amount of requests for each one of the in high quality encoded tiles.

VIII. Conclusion

In this work, we studied the problem of delivering 360^o videos in mobile networks using edge caching for unknown content popularity. We formulated the caching placement/eviction problem as a MDP that aims at maximizing the overall quality of the videos delivered to the users. To deal with the dimensionality problem, we employ a DQN solution that exploits the patterns from the observations in the sequence of users' requests, in order to learn for each state, which cache update action should be taken. In this way, we are able to cache the 360^o videos that are predicted to be the most popular, along with for each GOP, a virtual viewport. To evaluate our method, we use both real and synthetic navigation patterns. We extensively compare our proposed method with the LFU, LRU, and FIFO schemes. The results show that the proposed method outperforms its counterparts. This improved performance is attributed to the exploitation of the tiles' popularity and the use of virtual viewports instead of the original ones, which increases the flexibility in the caching decisions. While at this work, we assumed that all users have the same requirements, i.e., demand of 360^o videos encoded at the same quality, as part of our future work, we plan to study the case users having various quality requirements. In such case, our system could still be used, but the complexity of the problem will increase because the state-action space would grow in order to capture the fact that users have diverse capabilities.

References

[1] P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360^o videos in heterogeneous networks," in *Proc. of IEEE 21st Int. Workshop on Multimedia Signal Processing (MMSP'19)*, Kuala Lumpur, Malaysia, Malaysia, Sep. 2019.

[2] ——, "Tile-based joint caching and delivery of 360^o videos in heterogeneous networks," *IEEE Trans. on Multimedia*, vol. 22, no. 9, pp. 2382–2395, Sep. 2020.

[3] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. of IEEE Int. Conf. on Communications (ICC'17)*, Paris, France, May 2017.

[4] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "360 video viewing dataset in head-mounted virtual reality," in *Proc. of the 8th ACM on Multimedia Systems Conf. (MMSys'17)*, Taipei, Taiwan, Jun. 2017.

[5] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network - measurements, models, and implications," *Comput. Netw.*, vol. 53, no. 4, pp. 501–514, Mar. 2009.

[6] G. Papaioannou and I. Koutsopoulos, "Tile-based caching optimization for 360^o videos," in *Proc. of the 20th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing, Mobihoc '19*, Catania, Italy, Jul. 2019.

[7] P. Blasco and D. Gündüz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *Proc. of IEEE Int. Symp. on Information Theory*, Honolulu, HI, USA, Jun. 2014.

[8] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.

[9] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, May 1992.

[10] H. Li, T. Wei, A. Ren, Q. Zhu, and Y. Wang, "Deep reinforcement learning: Framework, applications, and embedded implementations," vol. abs/1710.03792, 2017. [Online]. Available: http://arxiv.org/abs/1710.03792

[11] F. Duanmu, Y. Mao, S. Liu, S. Srinivasan, and Y. Wang, "A subjective study of viewer navigation behaviors when watching 360-degree videos on computers," in *Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME'18)*, San Diego, CA, USA, July 2018.

[12] C. Ge, N. Wang, W. K. Chai, and H. Hellwagner, "QoE-assured 4K HTTP live streaming via transient segment holding at mobile edge," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1816–1830, Aug. 2018.

[13] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "QoE-driven mobile edge caching placement for adaptive video streaming," *IEEE Trans. on Multimedia*, vol. 20, no. 4, pp. 965–984, Apr. 2018.

[14] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, Sept. 2016.

[15] J. Poderys, M. Artuso, C. M. O. Lensbl, H. L. Christiansen, and J. Soler, "Caching at the mobile edge: A practical implementation," *IEEE Access*, vol. 6, pp. 8630–8637, Feb. 2018.

[16] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Edge-caching wireless networks: Performance analysis and optimization," *IEEE Trans. on Wireless Communications*, vol. 17, no. 4, pp. 2827–2839, Apr. 2018.

[17] S. Zhang, N. Zhang, P. Yang, and X. Shen, "Cost-effective cache deployment in mobile heterogeneous networks," *IEEE Trans. on Vehicular Technology*, vol. 66, no. 12, pp. 11 264–11 276, Dec. 2017.

[18] Z. Su, Q. Xu, F. Hou, Q. Yang, and Q. Qi, "Edge caching for layered video contents in mobile social networks," *IEEE Trans. on Multimedia*, vol. 19, no. 10, pp. 2210–2221, Oct. 2017.

[19] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. of the 52nd Annual Conf. on Information Sciences and Systems (CISS)*, Princeton, NJ, USA, Mar. 2018.

[20] Y. Wei, Z. Zhang, F. R. Yu, and Z. Han, "Joint user scheduling and content caching strategy for mobile edge networks using deep reinforcement learning," in *Proc. of IEEE Int. Conf. on Communications Workshops (ICC Workshops)*, Kansas City, MO, USA, May 2018.

[21] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," 2015. [Online]. Available: http://arxiv.org/abs/1512.07679

[22] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile based HEVC video for head mounted displays," in *Proc. of IEEE Int. Symp. on Multimedia (ISM'16)*, San Jose, CA, USA, Dec. 2016.

[23] M. Hosseini, "View-aware tile-based adaptations in 360 virtual reality video streaming," in *Proc. of IEEE Virtual Reality (VR'17)*, Los Angeles, CA, USA, Mar. 2017.

[24] J. Le Feuvre and C. Concolato, "Tiled-based adaptive streaming using MPEG-DASH," in *Proc. of the 7th Int. Conf. on Multimedia Systems*, Klagenfurt, Austria, 2016.

[25] A. Mahzari, A. Taghavi Nasrabadi, A. Samiei, and R. Prakash, "Fov-aware edge caching for adaptive 360^o video streaming," in *Proc. of the 26th ACM Int. Conf. on Multimedia*, ser. MM '18, Seoul, Rep. Korea, Oct. 2018.

[26] Q. Lu, C. Li, J. Zou, K. Tang, Q. Wang, and H. Xiong, "Transcoding-enabled edge caching and delivery for tile-based adaptive 360-degree video streaming," in *Proc. of IEEE Visual Communications and Image Processing (VCIP'19)*, Sydney, Australia, Australia, Dec. 2019.

[27] Y. Bao, H. Wu, A. A. Ramli, B. Wang, and X. Liu, "Viewing 360 degree videos: Motion prediction and bandwidth optimization," in *Proc.*

of IEEE 24th Int. Conf. on Network Protocols (ICNP'16), Singapore, Singapore, Nov. 2016.
[28] F. Duanmu, Y. Mao, S. Liu, S. Srinivasan, and Y. Wang, "A subjective study of viewer navigation behaviors when watching 360-degree videos on computers," in *Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME'18)*, San Diego, CA, USA, Jul. 2018.
[29] S. Rossi and L. Toni, "Navigation-aware adaptive streaming strategies for omnidirectional video," in *Proc of. IEEE 19th Int. Workshop on Multimedia Signal Processing (MMSP'17)*, Luton, UK, Oct. 2017.
[30] W. C. Lo, C. L. Fan, S. C. Yen, and C. H. Hsu, "Performance measurements of 360^o video streaming to head-mounted displays over live 4g cellular networks," in *Proc. of 19th Asia-Pacific Network Operations and Management Symp. (APNOMS'17)*, Seoul, South Korea, Sep. 2017.
[31] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proc. of the 24th Annual Int. Conf. on Mobile Computing and Networking, MobiCom '18*, New Delhi, India, 2018.
[32] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai, "A two-tier system for on-demand streaming of 360 degree video over dynamic networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 43–57, March 2019.
[33] M. Xiao, C. Zhou, Y. Liu, and S. Chen, "Optile: Toward optimal tiling in 360-degree video streaming," in *Proc. of the 25th ACM Int. Conf. on Multimedia, MM '17*, Mountain View, California, USA, Apr. 2017.
[34] M. Assens, K. McGuinness, X. Giró, and N. E. O'Connor, "Saltinet: Scan-path prediction on 360 degree images using saliency volumes," *CoRR*, vol. abs/1707.03123, 2017. [Online]. Available: http://arxiv.org/abs/1707.03123
[35] A. D. Aladagli, E. Ekmekcioglu, D. Jarnikov, and A. Kondoz, "Predicting head trajectories in 360^o virtual reality videos," in *Proc. of Int. Conf. on 3D Immersion (IC3D'17)*, Brussels, Belgium, Dec 2017.
[36] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1300–1313, June 2018.
[37] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.
[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.
[39] R. E. Bellman, *Dynamic Programming*. Dover Publications, 2003.
[40] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.
[41] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
[42] E. Nisioti and N. Thomos, "Robust Coordinated Reinforcement Learning for MAC Design in Sensor Networks," *Journal Selected Areas on Communications*, vol. 37, no. 10, pp. 2211–2224, Oct. 2019.
[43] L. Alexander, R. Johnson, and J. Weiss, "Exploring zipf's law," *Teaching Mathematics and Its Applications: Int. Journal of the IMA*, vol. 17, no. 4, pp. 155–158, Dec. 1998.
[44] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *Proc. of IEEE Conf. on Computer Communications, INFOCOM'14*, Toronto, ON, Canada, Apr. 2014.

Pantelis Maniotis received his diploma in Electrical and Computer Engineering from the Aristotle University of Thessaloniki in 2015 and his PhD from the School of Computer Science and Electronic Engineering at the University of Essex in 2020. His interests fall in the areas of multimedia technologies, wireless communications, Virtual and Augmented Reality, edge caching, and machine learning.

Nikolaos Thomos (S'02-M'06-SM'16) is an Associate Professor at the University of Essex, UK and the deputy director of research at School of Computer Science and Electronic Enineering. Previously, he was senior researcher at the Ecole Polytechnique Fédérale de Lausanne (EPFL), and the University of Bern, Switzerland. He received the Diploma and Ph.D. degrees from Aristotle University of Thessaloniki, Greece in 2000 and 2005 respectively. He is an elected member of IEEE MMSP Technical Committee (MMSP - TC) for the period 2019 - 2022. His research interests include machine learning for communications, multimedia communications, network coding, information-centric networking, source and channel coding, device-to-device communication, and signal processing. He received the highly esteemed Ambizione career award from Swiss National Science Foundation (SNSF) in 2008.