Adaptive Mobile VR Content Delivery for Industrial 5.0

Mushu Li
University of Waterloo
Waterloo, Ontario, Canada
m475li@uwaterloo.ca

Jie Gao
Carleton University
Ottawa, Ontario, Canada
jgao@carleton.ca

Conghao Zhou
University of Waterloo
Waterloo, Ontario, Canada
c89zhou@uwaterloo.ca

Xuemin (Sherman) Shen
University of Waterloo
Waterloo, Ontario, Canada
sshen@uwaterloo.ca

Weihua Zhuang
University of Waterloo
Waterloo, Ontario, Canada
wzhuang@uwaterloo.ca

ABSTRACT

Mobile virtual reality (VR) is expected to be a key component of the next-generation industrial internet-of-things, which uses immersive technologies to boost virtualization and facilitate human-machine collaboration in Industry 5.0. In this paper, we design a VR content delivery scheme to enhance VR content playback quality in mobile edge computing. The proposed scheme schedules computing resources on network edge to satisfy VR content requests from multiple user devices while reducing the likelihood of rebuffering and improving content freshness during VR video playback. With limited computing resources at the edge server, we develop a deep reinforcement learning (DRL) approach to determine which requests should be satisfied first, given the network and the service dynamics. By analyzing the network dynamics using the Whittle index method, the proposed DRL-based scheme can improve VR service quality with minimal communication overhead in computing scheduling. Simulation results demonstrate that the proposed scheme significantly improves the quality of service for VR content delivery.

KEYWORDS

Mobile virtual reality, mobile edge computing, deep reinforcement learning, computing resource scheduling.

ACM Reference Format:
Mushu Li, Jie Gao, Conghao Zhou, Xuemin (Sherman) Shen, and Weihua Zhuang. 2022. Adaptive Mobile VR Content Delivery for Industrial 5.0. In *1st Workshop on Digital Twin & Edge AI for IIoT (Digital Twin & Edge AI for Industrial IoT '22), October 17, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566099.3569002

1 INTRODUCTION

In Industry 5.0, artificial intelligence (AI) and human creativity are anticipated to be integrated to enhance manufacturing efficiency [13]. The integration of the above two elements can potentially overcome design challenges in autonomous systems and facilitate personalized manufacturing to meet the diversified demands of customers, as well as provide access for workers and customers to cyber-physical manufacturing [15].

As an immersive technology, mobile virtual reality (VR) offers a potential platform for the integration of AI and human creativity into manufacturing processes [5]. Watching 360° VR videos through a wireless headset gives manufacturing workers a sense of physical presence in a virtual world. By incorporating digital twins and internet-of-things (IoT) sensors, mobile VR can provide a way to integrate human cognition into the digital transformation required for industry 5.0. Several use cases have demonstrated this potential. For example, in industrial IoTs, mobile VR technologies can facilitate maintenance workflow by remotely controlling and resolving issues raised in production. Additionally, mobile VR can enable remote or co-located collaborative design, which allows engineers to create innovative manufacturing products through VR headsets and combine them in a virtual environment [1].

For supporting such novel use cases in Industry 5.0, delivering mobile VR videos at an extremely high data rate is essential, which creates a challenge for current networks. A mobile VR video can have a resolution up to 12K (11520 × 6480 pixels), while conventional videos typically have a resolution of 4K or less [7]. From a communication perspective, VR content delivery yields a heavy demand for bandwidth on both backhaul and wireless links. From a computing perspective, extensive VR video processing and rendering require high computational capability, while GPUs in wireless headsets are usually insufficient for supporting an acceptable frame rate to deliver videos with sufficient low latency [3].

The challenge brought by VR content delivery can be addressed by two suitable content delivery architecture and advanced VR video processing techniques. In terms of content delivery architecture, mobile edge computing leverages computing and storage resources on the edge devices, such as access points, to cache popular VR videos and perform video processing for headsets [6]. In terms of VR video processing techniques, tile-based content delivery methods have been developed to reduce the size of VR videos in content delivery. In such methods, a full 360° VR video is broken up into small spatio-temporal chunks, i.e., tiles, and only the chunks corresponding to the user's current viewing area are sent and rendered at the headset [4].

Edge computing and tile-based content delivery potentially enable mobile VR applications, yet implementing mobile VR in Industry 5.0 is still challenging. Many engineers or workers may coexist under the coverage of an edge server for requesting VR videos with

Digital Twin & Edge AI for Industrial IoT '22, October 17, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9784-1/22/10…$15.00
https://doi.org/10.1145/3566099.3569002

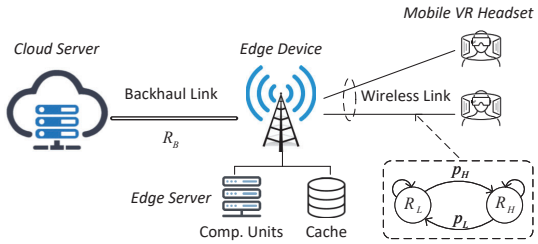

Figure 1: Network model.

ultra-low latency requirements. While the content delivery latency can be reduced by edge caching and computing, real-time delivery latency (~ 20 milliseconds [10]) is still difficult to achieve due to limited computing units and communication bandwidth associated with the edge server. Proactive content delivery approaches have been adopted in [8, 14] as a way to reduce delay, in which videos to be rendered are requested and downloaded in advance based on the predicted viewing trajectory or request profiles of VR users. However, the issue of resource scheduling for satisfying multiple VR content delivery requests needs to be addressed, and dynamics in the network and the VR service can affect the resource scheduling policies. Specifically, besides the dynamics introduced by the communication channels and the request arrival process, the accuracy of viewpoint prediction introduces additional uncertainty and has a significant impact on content delivery [9]. In addition, if a VR video is downloaded proactively and rendered by the headset after a period of time, the video may be updated in the interactions later. As a result, the freshness of the downloaded content degrades over time, and the downloaded copy can become outdated [16]. For example, in VR-enabled collaborative design, virtual objects may be modified by other designers after the video content containing the objects is downloaded to the VR headset of a designer, which results in outdated content being played. A computing scheduling scheme should be tailored for VR applications based on their content delivery features and the aforementioned dynamics.

In this paper, we design a VR content delivery scheme applicable to mobile edge computing. The research objective is to improve VR service quality, including reducing the likelihood of rebuffering and improving content freshness, i.e., the freshness of the downloaded content played at VR headset. To meet stringent content delivery requirements, we adopt proactive content delivery and propose a machine learning-based solution to schedule content requests received at an edge server. Specifically, a Whittle index (WI) based scheduling policy is used at the edge server to determine and prioritize requests. A weight parameter, i.e., the Whittle index, is calculated for each request. Then, a deep reinforcement learning (DRL) scheme is proposed to obtain the Whittle indexes of individual headsets according to the dynamics of individual user headsets and the network. The contributions of the paper are two-fold:

(1) We propose a DRL-based computing scheduling scheme that maximizes VR service quality given limited network resources through capturing and adapting to the network and the VR service dynamics.
(2) Facilitated by the WI scheduling policy, the proposed DRL-based scheme allows distributed learning and low communication overhead in real-time computing scheduling.

2 NETWORK MODEL

Fig. 1 shows the considered VR content delivery scenario, in which an edge server, located at the edge device, serves multiple VR users with headsets. The number of users in the considered network is denoted by U. The edge server accesses the cloud server via a wired link with data rate R_B and delivers VR videos to the user headsets through wireless communication links.

User headsets play 360° 3D stereoscopic VR videos. Video processing, including projection of 2D monoscopic videos to 3D stereoscopic videos and stitching multiple videos, can consume extensive computing resources. Since user headsets have very limited computing capability for completing the video processing within a desirable motion-to-photon latency (~20 ms), all VR videos are processed at the edge server. User headsets generate content delivery requests periodically for requesting VR videos to be rendered in future. We refer to the $k-th$ time interval when a headset generates request as time slot k, and the time slot is synchronized among user headsets. The edge server is equipped with E computing units to process the VR content requests from user headsets, and each computing unit can process only one computing task at a time. After a VR video is processed by the edge server, the server will deliver the video to the user headset. A mmWave band is used to communicate with a user headset, and a sub-6GHz band acts as a backup if the high-speed mmWave band in outage [6]. We use a two-stage Markov chain to model the data rate for the wireless link. The transmission rate is $R_E = R_H$ for the high-speed mmWave band and $R_E = R_L$ for the backup band. The probabilities of the high-speed state (with rate R_H) transiting from and to the low-speed state (with rate R_L) are denoted by p_L and p_H, respectively. As a result, the average data rate from the edge server to a user headset is $\bar{R}_E = (R_L + R_H)/(p_L + p_H)$.

2.1 VR Content Model

Two components comprise the complete VR video: graphic content, which are videos to be rendered at the headset, and auxiliary files, which are real-time updates to be stitched onto the graphic content in order to render the interactions. The graphic content can be stored at the edge and the cloud servers in advance, while the auxiliary files may be generated and updated to the servers in real time. We assume that both the cloud and the edge servers have up-to-date auxiliary files due to their relatively small data size and negligible transmission latency.

The tile-based content delivery solution is adopted for delivering and storing graphic content. Specifically, the whole VR video c can be divided into I blocks and J segments in spatial and temporal dimensions, respectively, and a divided VR video chuck is referred to as a tiled VR video. A tuple $t = (i, j, c)$ can represent a tiled VR video in VR video c in i-th blocks in the spatial area and in j-th time segment of the whole VR video. Furthermore, denote the auxiliary file generated in time slot k to be stitched into a tiled VR video t by $l_{t,k}$. If auxiliary file $l_{t,k}$ is not updated at the servers in time slot k, the auxiliary file in time slot $k-1$ is used, i.e., $l_{t,k} = l_{t,k-1}$. To satisfy a content delivery request from a user headset, multiple tiled VR videos associated with the request and auxiliary files applicable to the tiled VR videos are stitched together.

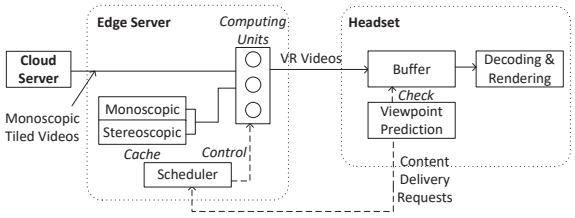

Figure 2: Content delivery model.

Due to VR headsets playing stereoscopic VR videos, two types of tiled VR videos can be stored on servers: monoscopic and stereoscopic videos. We assume that the cloud server has all monoscopic tiled VR videos. The edge server needs to convert monoscopic tiled VR videos into stereoscopic ones during the stitching process before the videos are delivered to headsets. Therefore, besides monoscopic tiled VR videos, the edge server can cache stereoscopic ones to reduce computing time in video delivery, while the size of stereoscopic tiled VR videos is greater than the size of monoscopic ones. The sets of monoscopic and stereoscopic tiled VR videos stored at the edge server are denoted by $\mathcal{B}_e^M$ and $\mathcal{B}_e^S$, respectively.

2.2 Content Delivery Model

The VR content delivery model is illustrated in Fig. 2. In each time slot, the headset renders VR videos under the user's field of view, tracks the user's viewpoint trajectory, and predicts the VR videos to be rendered in the subsequent time slots [14]. Denote the location of user u's viewpoint in the spherical domain in time slot k by $v_{u,k}$. The predicted viewpoints in subsequent P time slots of slot k is denoted by vector $\hat{\mathbf{v}}_u = [\hat{v}_{u,k+1}, \ldots, \hat{v}_{u,k+P}]$, where element $\hat{v}_{u,k}$ represents the predicted viewpoint of user u in time slot k. The headset stores the downloaded VR videos in a local buffer, and these videos are fetched according to user viewpoint movement, i.e., rendering the viewpoints. The set of viewpoints that can be rendered by the VR videos of user headset u's buffer in time slot k is denoted by $\mathcal{B}_{u,k}$.

After the prediction, the headset generates a set of VR content requests proactively for rendering future viewpoints. The content delivery request set from headset u in time slot k is denoted by set $\mathcal{D}_{u,k}$, where an element of $\mathcal{D}_{u,k}$ is a content delivery request including the viewpoint to be rendered and the corresponding time slot, i.e., $d_{u,k} = (\hat{v}_{u,k'}, k')|_{k'>k}$. A request set can contain at most two requests: one request for the next viewpoint that cannot be rendered by any VR videos in the local buffer, referred to as the new content request, and one request for the next viewpoint that can be rendered by a VR video in the local buffer but require updating, referred to as the update request. In the case of VR video without interactions, no auxiliary files are required, and the request set only includes the new content request.

The edge server receives requests from multiple headsets in each time slot. A scheduler at the edge server identifies which request to respond to first when a computing unit is available. Let $C(v_{u,k})$ represent the set of tiled VR videos to be stitched together to render the videos under the user's field of view when the viewpoint is $v_{u,k}$. If request $d = (v, k')$ is selected to be satisfied, the associated files, i.e., $C(v)$ and corresponding auxiliary files, are fetched from the cloud server and/or the cache of the edge server. The computing and transmission delays are modeled in the next section. Upon the VR video associated with the selected request being delivered, the corresponding computing unit becomes available for processing video for the next scheduled request.

If the requested VR video is delivered to the headset, the VR video will be stored in the local buffer. The latest time slot when the VR video for rendering viewpoint $v \in \mathcal{B}_{u,k}$ is downloaded is denoted by $g'(v)$, and $g'(v) = -1$ if the VR video for rendering viewpoint v is not downloaded to the buffer yet. Given that the latest auxiliary file update made for rendering viewpoint v is in time slot $g(v)$, if the VR videos in the buffer for rendering viewpoint v is up-to-date, $g'(v) > g(v)$. As VR video plays in time slot k, if the up-to-date VR video is stored, the headset renders the VR video directly without any penalty. If an outdated version of the requested VR video is stored in the buffer, the headset can render the VR video but service degradation happens due to the outdated rendered video. Lastly, if no VR video is available in the buffer to render the viewpoint, a rebuffering event occurs. We assume that that exclusive bandwidth and computing units at the edge server are reserved to resolve rebuffering events.

2.3 Delay Model

Three components contribute to content delivery delay: transmission delay for downloading monoscopic video from the cloud server to the edge server, computing delay for stitching and processing the VR video, and transmission delay for downloading the stereoscopic video from the edge server to the user headset. Given that request $d = (v, k')$ is selected by the scheduler, a computing unit is occupied. The edge server checks whether any tiled video in $t \in C(v)$ has not been cached. If so, the tiled videos that have not been cached are downloaded from the cloud server, which transmission delay is

$$T^{\mathrm{B}}(d) = \sum_{t \in C(v)/\{C(v) \cap \{\mathcal{B}_e^M \cup \mathcal{B}_e^S\}\}} w_t^{\mathrm{M}} R_B^{-1} \quad (1)$$

where the parameter w_t^{M} represents the data size of monoscopic tiled video t. When all tiled videos in set $C(v)$ are available at the edge server, the selected computing unit processes those tiled videos. The computing delay includes two components: the delay for converting monoscopic tiled videos into stereoscopic tiled videos and the delay for stitching the tiled videos and auxiliary files together. The corresponding computing delay is

$$T^{\mathrm{C}}(d) = \sum_{t \in C(v)/\{C(v) \cap \mathcal{B}_e^S\}} \chi_1 w_t^{\mathrm{M}} f^{-1} + \sum_{t \in C(v)} \chi_2 (w_t^{\mathrm{S}} + w_t^{\mathrm{A}}) f^{-1} \quad (2)$$

where parameters χ_1 and χ_2 denote the number of cycles to process a bit of data for monoscopic to stereoscopic converting and for stitching, respectively. The parameter f denotes the computing frequency in a computing unit. The parameters w_t^{S} and w_t^{A} are the data size of stereoscopic tiled video t and the auxiliary files applied to tiled video t, respectively. At last, the processed VR videos are delivered to the edge server, and the delay is

$$T^{\mathrm{E}}(d) = \sum_{t \in C(v)} \alpha (w_t^{\mathrm{S}} + w_t^{\mathrm{A}}) R_E^{-1} \quad (3)$$

where the value of R_E can be either R_L or R_H, following the two-stage Markov chain of the wireless channel model, and the parameter α represents the change in data size after stitching and decoding.

Therefore, the overall delay of content delivery for request d is

$$T(d) = T^{\mathrm{B}}(d) + T^{\mathrm{C}}(d) + T^{\mathrm{E}}(d). \tag{4}$$

Let δ denote the time length of a time slot. Given the current time slot k, if $T(d) > (k' - k) \times \delta$, the VR video cannot be delivered in time, resulting in rebuffering and outdated video. Denote the long-term popularity that viewpoint v is requested by a user headset by p_v, which can be obtained in advance from the historical VR playback profiles [12]. Given the sets of the cached tiled VR videos $\mathcal{B}_e^S$ and $\mathcal{B}_e^M$, the average content delivery time can be obtained by

$$\bar{T} = \sum_{v \in \mathcal{V}} p_v \{T^{\mathrm{B}}(v) + T^{\mathrm{C}}(v) + \sum_{t \in C(v)} (w_t^{\mathrm{S}} + w_t^{\mathrm{A}}) \bar{R}_E^{-1}\} \tag{5}$$

where set $\mathcal{V}$ is the set of all possible viewpoints.

3 PROBLEM FORMULATION

Due to limited computing resources, the scheduler at the edge server should first respond to the request that can reduce rebuffering and improve content freshness, as much as possible. However, the network and the service dynamics complicate scheduling decisions. First, satisfying requests based on inaccurate viewpoint prediction may waste computing and communication resources since the requested VR videos may not be used by the headsets. Furthermore, the content update frequency indicates the freshness of the VR contents in the buffer. Premature downloading of the popular VR video may result in the rendering of outdated content. Additionally, channel conditions and request generation dynamics also affect scheduling decisions.

We refer to the likelihood that a VR video in the buffer matches the current viewpoint as hit probability. Higher the hit probability means fewer rebuffering event. The objective of computing scheduling in content delivery is to jointly maximize the hit probability and the freshness of the rendered VR content. Denote a scheduling variable by $a_{u,k}$, i.e., $a_{u,k} = 1$ if the request from user headset u in time slot k is scheduled, and $a_{u,k} = 0$ otherwise. Let function $f(v_{u,k}; \mathcal{B}_{u,k})$ represent the VR content freshness for rendering viewpoint $v_{u,k}$. To model the content freshness degradation overtime due to auxiliary file update, we use exponential functions of the time gap between the current time slot and the time slot that the auxiliary file is updated in this paper as an example, but the proposed scheme is not limited to any specific type of functions. The function $f(v_{u,k}; \mathcal{B}_{u,k})$ is defined by

$$f(v_{u,k}; \mathcal{B}_{u,k}) = \begin{cases} \exp\{-\eta[k - g(v_{u,k})]\}, & \text{if } g(v_{u,k}) > g'(v_{u,k}) \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

where the non-negative parameter η represents the rate for freshness degradation of the VR content. The value of the function, referred to as the freshness score, decreases with the time gap between the current time slot and the time slot of the auxiliary file update. Then, the objective function can be formulated as

$$\max_{a_{u,k}, \forall u,k} \lim_{K \to \infty} \frac{1}{K} \mathbb{E}\Big[\sum_{u=1}^{U} \sum_{k=1}^{K} f(v_{u,k}; \mathcal{B}_{u,k}) \mathbf{1}[v_{u,k} \in \mathcal{B}_{u,k}] \Big] \tag{7}$$

where function $\mathbf{1}[x]$ is equal to one if x is true and zero otherwise. We aim to maximize the time-averaged number of events, in which the viewpoints can be rendered by the VR video in the local buffer,

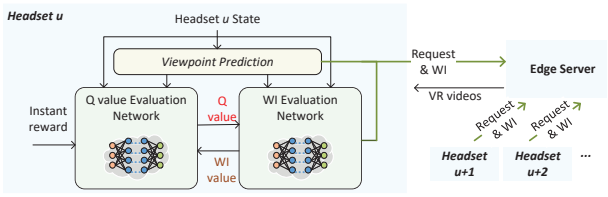

Figure 3: DRL-based computing scheduling scheme.

weighted by the corresponding freshness score. Due to limited computing units, we formulate a stochastic constraint:

$$\lim_{K \to \infty} \frac{1}{K} \mathbb{E}\Big[\sum_{u=1}^{U} \sum_{k=1}^{K} a_{u,k} \Big] \leq \frac{E \times \delta}{\bar{T}}. \tag{8}$$

where $\bar{T}/\delta$ represents the average number of time slots for delivering a VR video. The constraint in (8) is a relaxed constraint and indicates that less than E number of content delivery requests are simultaneously processed at the edge server on average.

Several factors affect the scheduling decision. Generally, the edge server would prioritize delivery of VR content that is urgently needed. However, if the user experiences poor channel quality or the VR video takes a long time to deliver, the user headset may not receive the requested VR video on time. In addition, when the user viewpoint changes rapidly, or the associated auxiliary files are frequently updated, the requested VR video may render viewpoints only for a few slots and become outdated soon. In contrast, when the user viewpoint moves slowly and in the unpopular area, the requested video chunk can support the viewpoint for multiple slots with a high freshness score. Those dynamics will be evaluated by the scheduler when the computing resource is limited and only a few requests can be responded at a time.

4 DRL-BASED COMPUTING SCHEDULING

To solve problem (7), we utilize the WI scheduling policy in restless multi-armed bandit (RMAB) and propose a DRL-based scheme to evaluate the WI in the scheduling policy.

4.1 Whittle Index Formulation

We first consider the case of only one content request in request set $\mathcal{D}_{u,k}$ from user headset u in time slot k, which is denoted by $d = (v, k')$. Then, the problem (7) with stochastic constraint (8) is an RMAB problem. For U user headsets, we define U corresponding controlled Markov chains. The state of the Markov chain for user headset u in time slot k is denoted by $S_{u,k}$, and the state space for user headset u is denoted by $\mathcal{S}^u$. A state $S_{u,k}$ consists of the historical viewpoint trajectory of user headset u, i.e., $\{v_{u,k^*}, k^* = k - P, \ldots, k\}$ and the local buffer state, i.e., $\mathcal{B}_{u,k}$. In addition, the information of the request d is also in the state, including the viewpoint to be rendered and the corresponding rendering time slot, i.e., $d = (v, k')$, the most recent time slots when corresponding VR video rendering v is updated and downloaded, i.e., $g(v)$ and $g'(v)$, and the set of tiled VR videos cached at the edge server.

The control variable for the Markov chain is $a_{u,k}$, which is binary: active when $a_{u,k} = 1$ and passive when $a_{u,k} = 0$. Consider the time length for a state transition as an epoch, which is denoted by $\hat{\phi}$. The average time length of an epoch is $\phi = \bar{T}/(E \times \phi)$, which

is utilized to find tractable state transition probabilities. The probability of transiting from state $S_{u,k}$ to $S_{u,k+\hat{\phi}}$ can be approximated by $P(S_{u,k+\phi}|S_{u,k}, a_{u,k})$, where $\sum_{S_{u,k+\phi}\in\mathcal{S}^u} P(S_{u,k+\phi}|S_{u,k}, a_{u,k}) = 1$. The reward for scheduling a request is defined to be the number of future viewpoints that can be rendered from granting request d, weighted by the corresponding freshness score. Specifically, the reward is given by

$$R(S_{u,k}, a_{u,k}) = \begin{cases} \sum_{k'=k+\phi}^{\infty} \kappa^{k'-k} f(v; \mathcal{B}_{u,k}) \mathbf{1}[v \in (\mathcal{B}_{u,k})], & \text{if } a_{u,k} = 1; \\ 0, & \text{if } a_{u,k} = 0. \end{cases} \tag{9}$$

The discounted reward model is applied in (9), where κ is a discount factor. The problem is now an RMAB problem. Whenever a computing unit is available, we select one request from a headset out of U headsets to schedule with unknown transition probabilities.

Our method of solving the RMAB problem is based on the WI, which is a heuristic solution with excellent empirical performance [11]. Specifically, the WI-based method defines a subsidy, i.e., the WI, for user headsets that requests are not scheduled by the edge server according to the current state of each headset. The edge server can receive a higher long-term reward by scheduling the requests with a higher subsidy. Therefore, the edge server can schedule the computing resources by comparing the WIs of requests. The WI of state $S_{u,k}$ can be obtained by

$$\lambda(S_{u,k}) = R(S_{u,k}, 1) + \kappa^{\phi}\Big\{ \sum_{S_{u,k+\phi}\in\mathcal{S}^u} P(S_{u,k+\phi}|S_{u,k}, 1) \max_{a_{u,k+\phi}} \{Q(S_{u,k+\phi}, a_{u,k+\phi})\} - \sum_{S_{u,k+\phi}\in\mathcal{S}^u} P(S_{u,k+\phi}|S_{u,k}, 0) \max_{a_{u,k+\phi}} \{Q(S_{u,k+\phi}, a_{u,k+\phi})\}\Big\}. \tag{10}$$

In (10), $Q(S_{u,k}, a_{u,k})$ is the state-action value, i.e., Q value, for state $S_{u,k}$ and action $a_{u,k}$. Eq. (10) presents the WI of one request from user headset u in time slot k. If request set $\mathcal{D}_{u,k}$ has more than one element, the user headset may generate WIs for all requests $d \in \mathcal{D}_{u,k}$ respectively, where the WI of request d is denoted by $\lambda(S_{u,k}(d))$, and $S_{u,k}(d)$ represents the state for user headset u in time slot k associated with request $d \in \mathcal{D}_{u,k}$. In time slot k, the edge server collects WIs $\{\lambda(S_{u,k}(d)), d \in \mathcal{D}_{u,k}\}$ from all user headsets and schedules the request with the highest WI value if a computing unit is available. The proof of indexability of the RMAB problem is omitted due to the page length limit.

4.2 DRL-based Scheduling

Eq. (10) can be leveraged to obtain the WI while the state-action transition probability is unknown. We adopt the deep reinforcement learning approach to approximate the WI value for each request. Two neural networks are constructed for each headset. One neural network approximates the Q value. Meanwhile, the other neural network approximates the WI value. The neural network weight vectors corresponding to the Q value and the WI value are denoted by $\boldsymbol{\theta}_Q$ and $\boldsymbol{\theta}_W$, respectively.

The proposed DRL-based computing scheduling scheme is illustrated in Fig. 3. At the beginning of a time slot k, user headset u observes its state $S_{u,k}$, predicts the future user viewpoints using existing viewpoint prediction technologies [8], and obtains request set $\mathcal{D}_{u,k}$. The Q value and WI evaluation networks approximate the set of Q values and WI values regarding state $S_{u,k}$, denoted by $Q(S_{u,k}, a_{u,k}|\boldsymbol{\theta}_Q)$ and $\lambda(S_{u,k}, a_{u,k}|\boldsymbol{\theta}_W)$, respectively. With the set of requests, the approximated WI values are sent to the edge server, and the edge server schedules the request with the highest WI. Furthermore, at each time slot, the weight vectors $\boldsymbol{\theta}_Q$ and $\boldsymbol{\theta}_W$ are updated according to the reward obtained in the real-time scheduling. In particular, if any request $d \in \mathcal{D}_{u,k}$ of user headset u is scheduled, the user headset observes the reward as given by (9). Otherwise, we use the approximated subsidy $\lambda(S_{u,k}|\boldsymbol{\theta}_W)$ as the reward, which represents the value gap between two successive states due to an unsatisfied request. According to [2], the weight vector $\boldsymbol{\theta}_Q$ is updated by minimizing the loss function, given by

$$L(\boldsymbol{\theta}_Q) = \frac{1}{|\mathcal{D}_{u,k}|} \sum_{d\in\mathcal{D}_{u,k}} [Q(S_{u,k}, a_{u,k}|\boldsymbol{\theta}_Q) - Q(S_{u,k+\phi}, a_{u,k}|\boldsymbol{\theta}_Q) \lambda(S_{u,k}|\boldsymbol{\theta}_W)(1 - a_{u,k}) - R(S_{u,k}, a_{u,k}) a_{u,k}]^2. \tag{11}$$

Furthermore, we generate a reference WI value obtained by the Q value and WI value neural networks, where

$$\hat{\lambda}(S_{u,k}) = \lambda(S_{u,k}|\boldsymbol{\theta}_W) - \varphi[Q(S_{u,k}, 0|\boldsymbol{\theta}_Q) - Q(S_{u,k}, 1|\boldsymbol{\theta}_Q)] \tag{12}$$

In (12), φ is the step size for approximating the WI value. Under the accurate approximation of $\lambda(S_{u,k})$, $Q(S_{u,k}, 0|\boldsymbol{\theta}_Q)$ should be equal to $Q(S_{u,k}, 1|\boldsymbol{\theta}_Q)$. Thus, the weight vector $\boldsymbol{\theta}_Q$ is updated by minimizing the loss function for evaluating the WI value, given by

$$L(\boldsymbol{\theta}_W) = [\varphi(Q(S_{u,k}, 0|\boldsymbol{\theta}_Q) - Q(S_{u,k}, 1|\boldsymbol{\theta}_Q))]^2. \tag{13}$$

The proposed DRL-based computing scheduling scheme is summarized in Alg. 1, where φ_Q and φ_W are the learning rates for evaluating weight vectors $\boldsymbol{\theta}_Q$ and $\boldsymbol{\theta}_W$, respectively. At each time slot, a user headset uploads a WI, which is a float number (4 bytes), for computing scheduling. Compared to sending the full headset state to the edge server for scheduling, the proposed scheme can significantly reduce communication overhead.

Algorithm 1 DRL-based computing scheduling scheme

```
Initialize weight vectors θ_W and θ_Q for all headsets. ϵ = 1.
for time slot k = 1, ..., K do
    for user headset u = 1 : U do
        Request new video chunks D_{u,k} and report WI set {λ(S_{u,k}(d); θ_W), d ∈ D_{u,k}}.
    end for
    With probability ϵ, randomly schedule a request; otherwise, schedule request d* with the highest WI.
    Update slot k = k + T(d*).
    For each user headset: θ_Q = θ_Q − φ_Q ∇L(θ_Q), θ_W = θ_W − φ_W ∇L(θ_W).
    If ϵ > ϵ_min, ϵ = β_attn ϵ.
end for
```

5 SIMULATION RESULTS

In this simulation, we use the headset tracking dataset consisting of 48 people watching three different 360° video in the coverage of an edge server[12]. The length of a time slot is 33 ms, and all VR videos are updated every four seconds. In each time slot, the headset uses a well-trained LSTM layer with 50 neurons to predict the viewpoint location for the next time slots with the prediction accuracy in 94%.

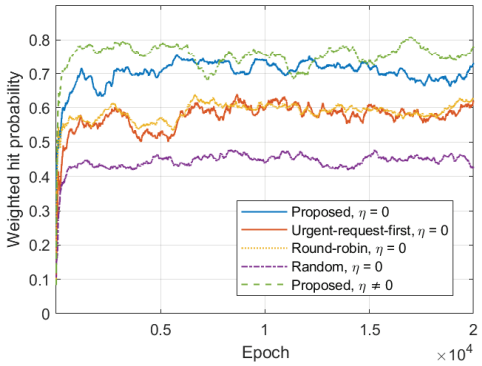

Figure 4: Weighted hit probability of scheduling schemes

The historical viewpoints in 50 time slots are used for viewpoint prediction. During each time slot, a user headset only requests that one VR video that has not been updated or downloaded yet be rendered next. In addition, each headset estimates the Q value and the WI using two neural networks that contain CNN layers to evaluate the VR videos stored in the local buffer, LSTM layers to analyze time correlations in viewpoint movement, and multiple fully connected layers. In specific, the network for evaluating the WI Value includes two $(3 \times 3 \times 10)$ convolution layers, which are connected by a (2×2) pooling layer, one LSTM layer with 50 neurons, followed by five fully connected layers with [125, 64, 64, 24, 1] neurons. The network for evaluating Q Value is the same as the one for WI value but has only three fully connected layers with [125, 64, 1] neurons. Other parameter settings are: $R_H = 250$ Mb/s, $R_L = 100$ Mb/s, $\varphi_Q = 5e-4$, $\varphi_Q = 1e-4$, and $\beta_{attn} = 0.9995$.

The performance of the proposed DRL-based scheduling scheme is compared with three benchmark schemes: urgent-request-first, round-robin, and random. The urgent-request-first scheme schedules the request with VR video to be rendered soon. The round-robin scheme schedules requests in a sequential manner. The random scheme schedules requests randomly using a uniform distribution. We also consider two types of freshness tolerance: the rate $\eta = 0$, i.e., no reward can be obtained if the rendered VR content is outdated, and η is proportional to the popularity of viewpoints. The results in Fig. 4 shows the average hit probability weighted by the freshness score. When $\eta = 0$, by comparing WI-based systems to round-robin schemes and urgent-request-first systems, the proposed scheme can improve the hit probability by approximately 15%. This is because the proposed scheme evaluates the scheduling decision while considering the impact of channel dynamics, request generation dynamic, prediction error, and viewpoint dynamics. When $\eta > 0$, the weighted hit probability increases due to the tolerance for delayed updates. In such a case, the proposed scheme provides scheduling decisions while considering the impact of dynamic content updates. The convergence performance of the proposed DRL-based scheduling scheme is shown in Fig. 5. The proposed scheme increases the weighted hit popularity during exploring the environment and achieves convergence after 12,000 epochs.

6 CONCLUSION

VR technologies drive industrial IoT towards Industry 5.0. In this paper, we have designed a computing resource scheduling scheme for real-time VR content delivery. The proposed scheme has explored the impact of the dynamics in the mobile VR application and the network on scheduling decisions. We have formulated a WI-based method to capture the long-term rewards for scheduling decisions and proposed a DRL-based scheme to obtain the WI. Leveraging the WI reported with the VR content requests, the proposed scheme can effectively schedule VR videos with low overhead while adapting to network dynamics. In the future, we will study the interplay between the content delivery scheme and the edge caching policy to further improve VR service quality.

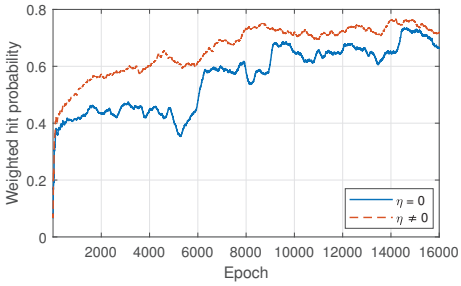

Figure 5: Convergence performance of the DRL-based scheduling scheme

REFERENCES

[1] Next G Alliance. 2022. *White paper: 6G Applications and Use Cases.* https://nextgalliance.org/white_papers/6g-applications-and-use-cases/

[2] K. Avrachenkov and V. Borkar. 2022. Whittle index based Q-learning for restless bandits with average reward. *Automatica* 139 (2022), 110186.

[3] K. Boos, D. Chu, and E. Cuervo. 2016. FlashBack: Immersive Virtual Reality on Mobile Devices via Rendering Memoization. In *Proc. 14th Annu. Int. Conf. Mobile Syst., Appl., and Services.* 291–304.

[4] J. Dai, Z. Zhang, S. Mao, and D. Liu. 2020. A View Synthesis-Based 360° VR Caching System Over MEC-enabled C-RAN. *IEEE Tran. Circuits Syst. Video Technol.* 30, 10 (2020), 3843–3855.

[5] P. Lin, Q. Song, D. Wang, F. R. Yu, L. Guo, and V. Leung. 2021. Resource Management for Pervasive-Edge-Computing-Assisted Wireless VR Streaming in Industrial Internet of Things. *IEEE Trans. Ind. Informatics* 17, 11 (2021), 7607–7617.

[6] Y. Liu, J. Liu, A. Argyriou, and S. Ci. 2019. MEC-Assisted Panoramic VR Video Streaming Over Millimeter Wave Mobile Networks. *IEEE Trans. Multimedia* 21, 5 (2019), 1302–1316.

[7] P. Maniotis, E. Bourtsoulatze, and N. Thomos. 2020. Tile-Based Joint Caching and Delivery of 360° Videos in Heterogeneous Networks. *IEEE Trans. Multimedia* 22, 9 (2020), 2382–2395.

[8] A. Nasrabadi, A. Samiei, and R. Prakash. 2020. Viewport Prediction for 360° Videos: A Clustering Approach. In *Proc. 30th ACM Workshop Netw. Operating Syst. Support for Digital Audio Video.* 34–39.

[9] C. Perfecto, M. S. Elbamby, J. D. Ser, and M. Bennis. 2020. Taming the Latency in Multi-User VR 360°: A QoE-Aware Deep Learning-Aided Multicast Framework. *IEEE Trans. Commun.* 68, 4 (2020), 2491–2508.

[10] S. Shi, V. Gupta, M. Hwang, and R. Jana. 2019. Mobile VR on Edge Cloud: A Latency-Driven Design. In *Proc. 10th ACM Multimedia Syst. Conf.* 222–231.

[11] P. Whittle. 1988. Restless bandits: Activity allocation in a changing world. *J. Appl. Probab.* (1988), 287–298.

[12] C. Wu, Z. Tan, Z. Wang, and S. Yang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. In *Proc. 8th ACM Multimedia Syst. Conf.* 193–198.

[13] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang. 2021. Industry 4.0 and Industry 5.0—Inception, conception and perception. *J. Manufacturing Syst.* 61 (2021), 530–535.

[14] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao. 2018. Gaze Prediction in Dynamic 360° Immersive Videos. In *Proc. 2018 IEEE/CVF Conf. Comput. Vision Pattern Recognition.* 5333–5342.

[15] S. Zeb, A. Mahmood, S. Khowaja, K. Dev, S. Hassan, N. Qureshi, M. Gidlund, and P. Bellavista. 2022. Industry 5.0 is Coming: A Survey on Intelligent NextG Wireless Networks as Technological Enablers. *arXiv preprint* (2022). arXiv:2205.09084.

[16] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri. 2017. Towards Efficient Edge Cloud Augmentation for Virtual Reality MMOGs. In *Proc. Second ACM/IEEE Symp. Edge Comput.* Article 8, 14 pages.