Content Prediction for Proactive Tile-Based VR Video Streaming in Mobile Edge Caching System

Qiuming Liu[1,2(✉)], Hao Chen[1], Yang Zhou[3], Dong Wu[3], Zihui Li[1], and Yaxin Bai[1]

[1] School of Software Engineering, Jiangxi University of Science and Technology, Nanchang 330013, China
liuqiuming@jxust.edu.cn, 6720210688@mail.jxust.edu.cn
[2] Nanchang Key Laboratory of Virtual Digital Factory and Cultural Communications, Nanchang 330013, China
[3] Information and Communication Branch, State Grid Jiangxi Electric Power Co., Nanchang 330095, China

Abstract. Content prediction can avoid VR video streaming delay in mobile edge caching system. To reduce request delay, popular content should be cached on edge server. Existing work either focuses on content prediction or on caching algorithms. However, in the end-edge-cloud system, prediction and caching should be considered together. In this paper, we jointly optimize the four stages of prediction, caching, computing and transmission in mobile edge caching system, aimed to maximize the user's quality of experience. We propose a progressive policy to optimize the four steps of VR video streaming. Since the user's QoE is determined by the performance of the resource allocation and caching algorithm, we design a caching algorithm with unknown future request content, which can efficiently improve the content hit rate, as well as the durations for prediction, computing and transmission. We optimize the four stages under arbitrary resource allocation and simulate the proposed algorithm according to the degree of overlap, as well as completion rate. Finally, under the real scenario, the proposed algorithm is verified by comparing with several other caching algorithms, simulation results show that the user's QoE is improved under the progressive policy and the proposed algorithm.

Keywords: VR video streaming · Content prediction · Caching algorithm · Quality of experience · Edge computing

1 Introduction

VR video has ultra-high resolution (e.g. 16K), which requires high computing rate and transmission rate to ensure low delay. Nowadays, computing and transmission resources cannot support the rapid transmission of ultra-high resolution

C. Wu et al. (Eds.): MONAMI 2023, LNICST 559, pp. 254–265, 2024.
https://doi.org/10.1007/978-3-031-55471-1_19

VR video. In view of the above VR transmission difficulties, researchers have proposed some technical methods to improve the quality of VR transmission. At each moment, the area that a human sees is only a part of the VR video, which called the field of view (FoV) [1]. Dividing the video into tiles and focusing on the parts in the FoV can reduce the amount of content transmitted. The motion to photon (MTP) delay causes dizziness for users watching VR videos. Proactive tile-based VR video streaming can avoid the MTP delay [2], which sends the predicted content to the user's head mounted display (HMD) in advance. Caching popular content on the edge server and providing VR content directly to users at the edge can reduce backhaul link pressure and request delay. The above technical methods improve the quality of VR video transmission from three perspectives: reducing transmission content, prediction and caching. Next, we discuss the current research status of VR transmission.

In the existing works, VR video streaming research almost always divides VR video into tiles and system mainly transmits the content in FoV [3–12]. There are also many articles study caching algorithm that can cache popular content on the edge server to reduce remote request delay [4–7]. In the prediction study of FoV, researchers prefer to use machine learning to design predictors [8–12]. However, the above work either only designs the cache algorithm, or only designs the predictor, or only jointly optimizes the prediction, computation and transmission in end-edge system. These works do not jointly optimize prediction, computing and transmission under delay constraints in end-edge-cloud system (EECS) to improve user's quality of experience (QoE).

In this paper, we study the impact of prediction, computing, transmission on QoE under delay constraints. The main contributions are summarized as follows.

- We jointly optimize prediction, computing, and transmission under delay constraints in EECS. The relationship between four steps is analyzed and QoE is maximized under different resource allocations. We propose a progressive policy to balance the four steps to make QoE as big as possible. The use of the progressive policy makes 93.178% of the experimental segments achieve the maximum QoE. In the remaining experimental segments where the maximum value is not obtained, the maximum difference rate between the QoE obtained by the progressive strategy and the maximum value is 0.406298%, which is very close to the maximum value.
- We design a caching algorithm to improve the efficiency of caching and thus improve the overall QoE, which can effectively reduce the request delay. The time saved is used for prediction, computing, transmission to improve prediction performance and completion rate. Then, our algorithm provides better QoE among the six algorithms simulated.

2 Model System

In the EECS, each VR video V_i consists S segments, and each segment is composed by M tiles. We assume that the storage size of the MEC buffer is denoted by B, which means that the MEC can cache B tiles. User's HMD can record data

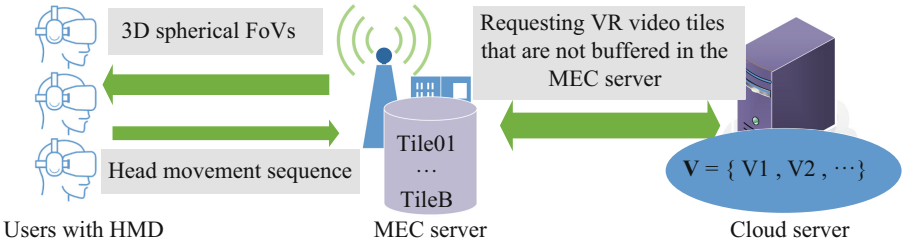

Fig. 1. End-Edge-Cloud System

of head movement sequence, and send the recorded data to the MEC server. The head motion sequence is used to predict FoV. Both HMD and the MEC server can render the VR videos. We only consider the rendering on the MEC server, since it would cause delay in HMD. The VR video stream is rendered to 3D spherical FoV by MEC and then sent to the user.

We consider an EECS with a proactive tile-based VR video streaming as shown in Fig. 1. The EECS includes some users with HMD, a MEC server co-located with base station (BS), and a back-end cloud. Clients request VR videos according to unpredictable patterns. If the edge sever holds the contents, it will response the request immediately. Otherwise, the edge server would resort to the cloud server, where holds the entire VR videos $V = \{V_1, V_2, \cdots\}$. Forwarding content requests to the back-end cloud incurs request delay.

In order to avoid MTP delay, the content required by the $(s+1)$th segment should be sent to the user's HMD before t^e as shown in Fig. 2. Therefore, when the user's HMD plays (s)th segment, the MEC server is processing the tiles needed for $(s+1)$th segment. The request delay t_{req} can be determined after the prediction is completed. Within the time delay t_{req}, the progressive policy allocates the duration of each task according to the optimization results. The predicted FoV tiles must be rendered as 3D spherical FoV to start transmission, so the runtime of the four-step task is serial. Each segment contains four steps: prediction, request delay, computing and transmission, which are processed serially in each segment. Next, we illustrate the processing of $(s+1)$th segment as an example.

We use $[t^b, t^e]$ to represent the playback time of (s)th segment. The $(s+1)$th segment VR video streaming is serially processed as follows. Prediction stage: The MEC server observes in the observation window t_{pdc} and maps the recorded data into predicted tiles. Request delay: Predicted tiles that are not in the MEC server buffer should be requested from the back-end cloud with a request delay t_{req}, and the MEC server caches popular tiles. Computing stage: On the MEC server, predicted tiles are rendered into 3D spherical FoV in duration t_{cpt}. Transmission stage: The processed tiles are transmitted to the HMD with the duration t_{tra}. One segment duration denoted as T_{seg}, which is the total proactive streaming time available for prediction, caching, computing, and transmission, *i.e.* $t_{pdc} + t_{req} + t_{cpt} + t_{tra} \leq T_{seg}$.

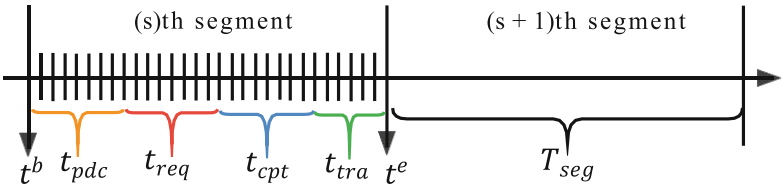

Fig. 2. Four processing steps within a segment

2.1 Delay Model

Due to the limited edge buffer size, the MEC server cannot cache all VR videos. Therefore, the content that is not at the edge needs MEC server to forward the request to the cloud for acquisition. The delay of requesting tiles to the cloud is related to the number of tiles being requested remotely. Then, the request delay is expressed as:

$$t_{req} = n_0 \cdot (T_e \cdot d_0), \tag{1}$$

where T_e is the average transmission time of per tile at the edge. The number of remote request tiles is expressed as n_0. For the same data, the delay of remote cloud request is d_0 times of edge transmission time [13].

2.2 Computing Model

Rendering the predicted tiles within the computing duration t_{cpt}. The computing resources of rendering VR video on the MEC server are equally allocated to each user. Then, the computing rate for one user is expressed as:

$$C_{cpt} \triangleq \frac{C_{all}}{K \cdot U_0}(bits/s), \tag{2}$$

where C_{all} is the total computing resource (FLOPS), which should be equally allocated to K users. The FLOPs required to render one bit is expressed as U_0.

2.3 Transmission Model

The proactive tile-based VR video streaming in one segment has undergone three-step processing, which should be transmitted to HMD with duration t_{tra}. Then the duration t_{tra} is expressed as:

$$t_{tra} \triangleq \frac{(R_w \cdot R_h \cdot b) \cdot N_{fov}}{C_e \cdot \gamma}(s), \tag{3}$$

where γ is the Lossless compression ratio, and the transmission rate from MEC server to the user's HMD is expressed as C_e. The number of tiles in the FoV is expressed as N_{fov}. The corresponding transmission rate is expressed as C_{tra}. R_h and R_w are the pixels in high and wide of a tile, respectively. b is the number of bits per pixel.

3 Problem Formulation

The goal of this paper is to improve the quality of user experience, and the user's QoE is closely related to prediction, request delay, computing, and transmission. Prediction determines the accuracy of the content received by the user's HMD. We use the degree of overlap (DoO) to represent the predicted performance. DoO represents the proportion of correctly predicted tiles in FoV. We use completion rate to represent the performance of computing and transmission. DoO and completion rate affect user's QoE in content quality and content quantity respectively. The caching algorithm affects request delay. This section formulates the problem of maximizing QoE through completion rate and DoO.

3.1 Performance Metric of QoE

We use an existing predictor, and mainly analyze the effect of the observation window on the quality of the prediction. The larger the observation window, the better the prediction results [3]. We need to optimize the three-step task under delay constraints before the next segment request arrival, so we use the fitting function to represent DoO, which is expressed as:

$$DoO(t_{pdc}) = a_3 \cdot t_{pdc}^3 + a_2 \cdot t_{pdc}^2 + a_1 \cdot t_{pdc} + a_0, \tag{4}$$

where $\{a_0, a_1, a_2, a_3\}$ are the fitted coefficients. We only need to know the size of the observation window t_{pdc} to get the $DoO(t_{pdc})$.

The performance of computing and transmission is expressed by the completion rate, which can be expressed as:

$$R(t_{cpt}, t_{tra}) \triangleq min\left\{1, \frac{C_{cpt} \cdot t_{cpt}}{s_{cpt} \cdot N_{fov}}, \frac{C_{tra} \cdot t_{tra}}{s_{tra} \cdot N_{fov}}\right\}, \tag{5}$$

where s_{cpt} and s_{tra} are the number of bits to compute and transmit in one tile respectively.

The user's QoE is determined by the VR content quality and VR content integrity. QoE can be expressed by completion rate and DoO [3], which is defined as:

$$QoE(t_{pdc}, t_{cpt}, t_{tra}) \triangleq DoO(t_{pdc}) \cdot R(t_{cpt}, t_{tra}), \tag{6}$$

where $t_{pdc} + t_{req} + t_{cpt} + t_{tra} \leq T_{seg}$.

3.2 Joint Optimization

Under the known conditions of predictor, computing rate C_{cpt} and transmission rate C_{tra}, we optimize the duration of prediction, computing and transmission under delay constraints to maximize QoE, which can be expressed as:

$$\mathbf{P1}: \max_{t_{pdc}, t_{cpt}, t_{tra}} DoO(t_{pdc}) \cdot R(t_{cpt}, t_{tra}) \tag{7a}$$

$$s.t. \quad t_{pdc} + t_{req} + t_{cpt} + t_{tra} \leq T_{seg} \tag{7b}$$

$$t_{pdc} \geq \tau, \quad t_{req}, t_{cpt}, t_{tra} \geq 0 \tag{7c}$$

The total duration of the four steps should be less than or equal to one segment duration T_{seg}. τ is the minimum value of the observation window. Next section, we solve the **P1** problem, propose progressive policy and VIE online caching algorithm.

4 Optimal Durations and Progressive Policy and VIE Online Caching Algorithm

4.1 Optimal Durations

We solve the **P1** problem to obtain t^*_{pdc}, t^*_{cpt} and t^*_{tra}, where t^*_{pdc}, t^*_{cpt} and t^*_{tra} maximize the objective problem **P1**. The solution process is: we decompose the **P1** problem into optimizing the completion rate first and then solving the **P1** problem. The sum of computing and transmission duration is expressed as t_{ct}, i.e. $t_{ct} = t_{cpt} + t_{tra}$. According to the KKT condition, the first step optimization completion rate $\underset{t_{cpt},t_{tra}}{max}\ R(t_{cpt}, t_{tra})$ can be solved:

$$\begin{aligned}
t^*_{cpt}(t_{ct}) &= \begin{cases} \frac{C_{tra}s_{cpt}}{C_{tra}s_{cpt}+C_{cpt}s_{tra}}t_{ct}, & t_{ct} \le T^{max}_{ct} \\ \alpha, \alpha \in (\frac{s_{cpt}N_{fov}}{C_{cpt}}, \infty), & t_{ct} > T^{max}_{ct} \end{cases} \\
t^*_{tra}(t_{ct}) &= \begin{cases} \frac{C_{cpt}s_{tra}}{C_{tra}s_{cpt}+C_{cpt}s_{tra}}t_{ct}, & t_{ct} \le T^{max}_{ct} \\ \beta, \beta \in (\frac{s_{tra}N_{fov}}{C_{tra}}, \infty), & t_{ct} > T^{max}_{ct} \end{cases} \\
R^*_{(t_{cpt},t_{tra})}(t_{ct}) &= \begin{cases} \frac{t_{ct}}{T^{max}_{ct}}, & t_{ct} \le T^{max}_{ct} \\ 1, & t_{ct} > T^{max}_{ct} \end{cases},
\end{aligned} \tag{8}$$

where only t_{ct} is a variable, we get the solution related to t_{ct}. In the above formula, $T^{max}_{ct} \triangleq \frac{s_{tra}N_{fov}}{C_{tra}} + \frac{s_{cpt}N_{fov}}{C_{cpt}}$. T^{max}_{ct} is the minimum value required for t_{ct} when the completion rate up to 100%, so t_{ct} does not need to exceed T^{max}_{ct}. T^{max}_{cr} represents the configuration of computing resources and transmission resources, which is a known value. In the case of $t_{ct} \le T^{max}_{ct}$, problem **P1** can be re-writte as:

$$\mathbf{P2}: \underset{t_{pdc},t_{ct}}{max} \quad DoO(t_{pdc}) \cdot \frac{t_{ct}}{T^{max}_{ct}} \tag{9a}$$

$$s.t. \quad t_{pdc} + t_{ct} + t_{req} \le T_{seg} \tag{9b}$$

$$t_{pdc} \ge \tau, \quad 0 \le t_{ct} \le T^{max}_{ct} \tag{9c}$$

According to the KKT condition, the available result of **P2** is :

$$\begin{aligned}
t^*_{pdc}(t_{req}) &= max\{(T_{seg} - t_{req}) - T^{max}_{cr}, \tau\} \\
t^*_{ct}(t_{req}) &= min\{T^{max}_{cr}, (T_{seg} - t_{req}) - \tau\}
\end{aligned} \tag{10}$$

Finally, we get $t^*_{pdc}(t_{req})$ and $t^*_{ct}(t_{req})$. Through $t^*_{ct}(t_{req})$, we can also get $t^*_{cpt}(t_{ct})$, $t^*_{tra}(t_{ct})$ and $R^*_{(t_{cpt},t_{tra})}(t_{ct})$, and QoE is expressed as $DoO(t^*_{pdc}(t_{req})) \cdot \frac{t^*_{ct}(t_{req})}{T^{max}_{ct}}$. The optimization result still has a variable t_{req}. Next, we design a progressive strategy to maximize QoE as much as possible in the case of t_{req} uncertainty, and design an online caching algorithm to reduce t_{req}.

4.2 Progressive Policy

Because t_{req} cannot be known before prediction, progressive policy is designed to make the QoE of each segment as close as possible to the optimal value. The prediction window expands step by step from the smallest τ is the progressive meaning. Each prediction window can get the corresponding request delay t_{req}, then the t_{ct} is obtained through the optimization results, and the QoE value under this prediction window is calculated.

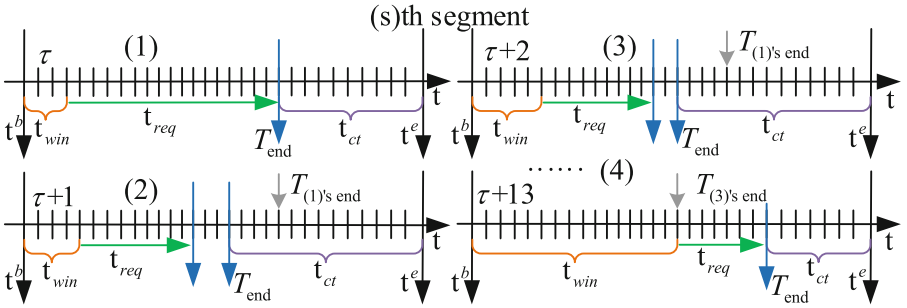

Fig. 3. An example of progressive policy operation

Figure 3 shows the process of progressive policy implementation. In (1) state, The observation starts from t^b, and the current observation window size is $t_{win} = \tau$. After observation, the request delay t_{req} corresponding to the predicted data is obtained, and t^*_{ct} is calculated according to the optimization result formula (10). According to the above information, if we want to select the durations allocation in (1), we can determine that the observation window can be extended to T_{end} at most. The observation window expands one step to state (2), where $t_{win} = \tau + 1$. In state (2), t^*_{ct} reaches t^{max}_{ct}, so the extensible space of state (2) is larger than t_{req}. We compare the QoE values of state (2) and state (1), and assume that the QoE of state (1) is larger. Therefore, we discard all the information of state (2) and only record the durations allocation and maximum extension time $T_{(1)'send}$ of state (1). The observation window expands one step to state (3), where $t_{win} = \tau + 2$. We assume that the QoE of state (3) is larger than that of state (1). So we only keep the information of state (3). Next, we omit the multi-step extension processing and assume that the QoE of state (3) is the largest, directly to the last state (4). The observation window has been extended to the maximum extension time $T_{(3)'send}$, and the QoE is less than the QoE of state (3), so we use the durations allocation of state (3).

The progressive policy is adopted in each segment, which pseudo code is shown in Algorithm 1.

Algorithm 1. Progressive Policy

$t_{win} \leftarrow \tau$, $Data \leftarrow None$
do {
  $N_{fov} = \text{predict}(t_{win})$
  $n_0 = N_{fov} - N_{inB}$, $t_{req} = n_0 \cdot T_e \cdot d_0$
  $t_{ct} = \min\{T_{ct}^{max}, (T_{seg} - t_{req}) - \tau\}$
  $QoE = DoO(t_{win}) \cdot \frac{t_{ct}}{T_{ct}^{max}}$
  if (this QoE is bigger)
    $T_{end} = 1 - min(T_{ct}^{max}, 1 - t_{win} - t_{req})$
    $Data \leftarrow \{T_{end}, QoE\}$
  $t_{win} \leftarrow t_{win} + 1$
}**while** $(t_{win} < T_{end})$
Output: QoE

4.3 The VIE Online Caching Algorithm

Our online caching algorithm consists of two parts: Caching popular tiles to the MEC buffer and removing unpopular tiles from the MEC buffer. We propose the *recent victor download and recent failure deletion* (VIE) online caching algorithm, which assumes the future request content is unknown.

Definition 1. ***Recent victor download:*** *Each tile is recorded when it was requested. Tile te_j is not in the MEC server buffer, and te_i is in the buffer. $te_i(t)$ represents whether tile te_i is requested at time t. $te_i(t) = 1$ indicates that the tile te_i is requested at t. $te_i(t) = 0$ indicates that the tile te_i is not requested at time t. At time t_0, if there exists a positive integer λ and a boundary η such that te_i and te_j satisfies:*

$$\sum_{t=t_0-\lambda}^{t_0} te_j(t) \geq \sum_{t=t_0-\lambda}^{t_0} te_i(t) + \eta \tag{11}$$

Then te_j will be cached in the MEC server buffer.

According to Definition 1, *recent victor download* algorithm judge the situation in the recent period according to the boundary parameter η. If a tile te_j that has not been cached in the MEC buffer in the recent period of time is requested more times than the tile cached in the MEC buffer, then we cache te_j. We call te_j the victor.

Definition 2. ***Recent failure deletion:*** *Tile te_i cached in the buffer corresponds to a positive integer ζ_i as small as possible. te_i satisfies the requested number at time $t_0 - \zeta_i \leq t \leq t_0$ is greater than or equal to η, that is, $\sum_{t=t_0-\zeta_i}^{t_0} te_i(t) \geq \eta$. Selecting the tile te_i with the biggest ζ_i to remove from MEC server.*

According to Definition 2, algorithm selects the te_i corresponding to the largest ζ_i to delete. ζ_i represents the size of the time range. In the experiment, we set $\eta = 2d_0$.

5 Simulation

We refer to the real scenario in [14], which includes 50 users and 10 VR videos. Each VR video is 1 minute long and divided into 60 segments. d_0 is set to 12 [13]. The pixel of the tile is 192×192, i.e. $R_w = 192$ and $R_h = 192$, and the number of bits per pixel is 12, i.e. $b = 12$. Lossless Coding Compression Rate Set to 2.41, i.e. $\gamma = 2.41$. T_{cr}^{max} is set to { 0.2s, 0.25s, 0.3s, ···,50s }. We use the existing RL predictor for simulation. Users requests VR video with Zipfian distribution [5], so the probability that the VR video $v \in V$ is selected for viewing is:

$$P_v = \frac{1/v^{\eta_v}}{\sum_{v \in V} 1/v^{\eta_v}} \tag{12}$$

We verified the performance of the VIE algorithm in terms of user's QoE and completion rate. The proposed algorithm is compared with five caching algorithms, which are:

- Static Offline: The most popular tiles are cached on edge server, which contains all future request information, and the cached content is not updated.
- LFU [4,5]: The number of times each tile has been requested is recorded, and caching newly arrived tiles by removing the least frequently used tiles.
- LRU [4,5]: The last requested time of each tile is recorded, and caching newly arrived tiles by removing least recently used tiles.
- FIFO [5]: Each requested tile is queued, and newly arriving tiles are cached by dequeuing.
- Randomized: Tiles are randomly cached in the buffer.

We first simulate the progressive policy. We use randomized algorithm and set the edge buffer to be 10% of the total data to conduct experiments on 1764690 segments. The experimental results show that 1644308 segments have achieved the maximum QoE, which accounts for 93.178% of the total experiments. In the remaining 6.822% results, the QoE obtained by the progressive policy is very close to the maximum value, the difference rate in the range of [0.227357%, 0.406298%]. Six examples of the progressive policy that did not achieve the maximum value in one segment are shown in Table 1.

Table 1. Six examples that do not reach the maximum QoE.

Minimum QoE	Maximum QoE	QoE of Progressive Policy	Difference Rate
0.149304	0.415359	0.413940	0.341632%
0.109049	0.465753	0.464640	0.238968%
0.090175	0.385383	0.383946	0.372876%
0.125859	0.352531	0.351507	0.290471%
0.079028	0.257461	0.256875	0.227607%
0.092893	0.396753	0.395804	0.239192%

In Fig. 4, we use boxplots to evaluate the performance of the six algorithms on QoE. Information can be obtained from the figure: **a)** The VIE algorithm achieves the highest QoE among the six algorithms. **b)** The baseline randomized algorithm is significantly lower than the other five algorithms. **c)** The overall QoE of LFU [4,5], LRU [4,5] and FIFO [5] is lower than the VIE algorithm.

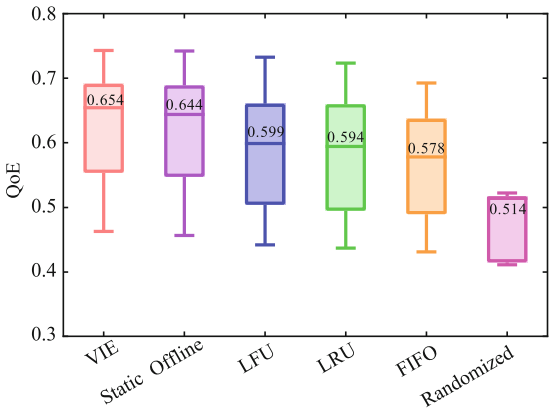

Fig. 4. Comparison of the VIE caching algorithm and other five algorithms on QoE. The cache size is 15% of total data and T_{cr}^{max} is 0.8 s.

In Fig. 5, we use boxplots to evaluate the performance of the six algorithms on completion rate. The buffer size is set to 15% of total data, and T_{cr}^{max} is set to 0.8 s. The completion rate of the VIE algorithm is the highest and random algorithm is the lowest. The completion rates of LFU [4,5], LRU [4,5], and FIFO [5] are significantly smaller than the VIE algorithm.

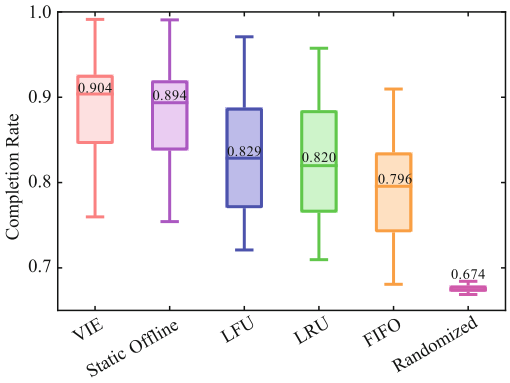

Fig. 5. Comparison of the VIE caching algorithm and other five algorithms on completion rate. The cache size is 15% of total data and T_{cr}^{max} is 0.8 s.

In Fig. 6, The graph illustrates that the average QoE increases as resources increase. The smaller the T_{cr}^{max}, the stronger the computing and transmission capabilities, and the higher the average QoE. Five different colored lines represent different edge buffer size. The edge cache size is proportional to the average QoE.

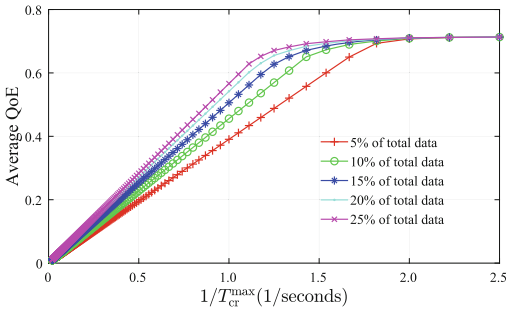

Fig. 6. The variation of the average QoE of the VIE algorithm with different cache sizes and different T_{cr}^{max}.

Under the duration allocation of the progressive policy, 93.178% of the data achieves the maximum QoE, and the maximum difference rate in QoE that does not reach the maximum value is 0.406298%. The experimental results show that the VIE algorithm significantly outperforms the other five caching algorithms in terms of user's QoE and completion rate. The static offline algorithm is only a little worse than the VIE algorithm, but the static offline algorithm needs to know all the request information in advance, which is unrealistic.

6 Conclusion

In this paper, we investigate the optimization of proactive tile-based VR video streaming in mobile edge caching system. We jointly optimized prediction, computing, transmission under delay constraints to maximize QoE, and proposed a progressive policy. A good caching algorithm can improve the optimization results of progressive policy. We design an online caching algorithm to improve user's QoE, which caches popular tiles by a threshold to improve cache performance. The six caching algorithms are simulated in real scenario using existing predictor. The VIE algorithm is compared with five other caching algorithms in terms of user's QoE and completion rate. Experiments verified the performance of the progressive policy and the performance of the VIE algorithm.

Acknowledgment. This work was supported in part by Culture and Art Science Planning Project of Jiangxi Province (No. YG2018042), Humanities and Social Science Project of Jiangxi Province (No. JC18224).

References

1. Kattadige, C.: PhD forum: encrypted traffic analysis & content awareness of 360-degree video streaming optimization. In: 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) (2021)
2. Fan, C.L., Lo, W.C., Pai, Y.T., Hsu, C.H.: A survey on 360° video streaming: acquisition, transmission, and display. ACM Comput. Surv. (CSUR) **52**(4), 1–36 (2019)
3. Wei, X., Yang, C., Han, S.: Prediction, communication, and computing duration optimization for VR video streaming. IEEE Trans. Commun. **69**(3), 1947–1959 (2021)
4. Mahzari, A., Nasrabadi, A.T., Samiei, A., Prakash, R.: FOV-aware edge caching for adaptive 360° video streaming. In: 2018 ACM Multimedia Conference (2018)
5. Maniotis, P., Thomos, N.: Viewport-aware deep reinforcement learning approach for 360° video caching. IEEE Trans. Multimedia **24**, 386–399 (2022). https://doi.org/10.1109/TMM.2021.3052339
6. Ji, S., Lee, S., Park, G., Song, H.: Head movement-aware mpeg-dash SRD-based 360° video VR streaming system over wireless network. In: 2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 281–287 (2022). https://doi.org/10.1109/WoWMoM54355.2022.00054
7. Xiao, H., et al.: A transcoding-enabled 360° VR video caching and delivery framework for edge-enhanced next-generation wireless networks. IEEE J. Sel. Areas Commun. **40**(5), 1615–1631 (2022). https://doi.org/10.1109/JSAC.2022.3145813
8. Rondon, M., Sassatelli, L., Aparicio-Pardo, R., Precioso, F.: TRACK: a new method from a re-examination of deep architectures for head motion prediction in 360-degree videos. IEEE Trans. Pattern Anal. Mach. Intell. **44**(9), 5681–5699 (2021)
9. Zhang, R., et al.: Buffer-aware virtual reality video streaming with personalized and private viewport prediction. IEEE J. Sel. Areas Commun. **40**(2), 694–709 (2022). https://doi.org/10.1109/JSAC.2021.3119144
10. Cheng, Q., Shan, H., Zhuang, W., Yu, L., Zhang, Z., Quek, T.Q.: Design and analysis of MEC- and proactive caching-based 360° mobile VR video streaming. IEEE Trans. Multimedia **24**, 1529–1544 (2022). https://doi.org/10.1109/TMM.2021.3067205
11. Liu, X., Deng, Y., Han, C., Renzo, M.D.: Learning-based prediction, rendering and transmission for interactive virtual reality in RIS-assisted terahertz networks. IEEE J. Sel. Areas Commun. **40**(2), 710–724 (2022). https://doi.org/10.1109/JSAC.2021.3118405
12. Zhu, Y., Zhai, G., Yang, Y., Duan, H., Min, X., Yang, X.: Viewing behavior supported visual saliency predictor for 360 degree videos. IEEE Trans. Circuits Syst. Video Technol. **32**(7), 4188–4201 (2022). https://doi.org/10.1109/TCSVT.2021.3126590
13. Song, Y., Wo, T., Yang, R., Shen, Q., Xu, J.: Joint optimization of cache placement and request routing in unreliable networks. J. Parallel Distrib. Comput. **157**, 168–178 (2021)
14. Lo, W.C., Fan, C.L., Lee, J., Huang, C.Y., Chen, K.T., Hsu, C.H.: 360° video viewing dataset in head-mounted virtual reality. In: The 8th ACM, pp. 211–216 (2017)