Cost-Aware Task Offloading and Migration for Wireless Virtual Reality Using Interactive A3C Approach

Peng Lin, *Member, IEEE,* Yan Liu, Zhizhong Zhang, F. Richard Yu, *Fellow, IEEE,* Victor C. M. Leung, *Life Fellow, IEEE*

***Abstract*—Wireless virtual reality (VR) is becoming a promising service to provide users with immersive experience from anywhere. To deal with the performance-cost negotiation for a MEC-enabled wireless VR, we propose a cost-aware task offloading and migration scheme. We formulate the viewport rendering offloading, task migration, and subchannel allocation as an optimization problem, taking into account a long-term MEC operational cost budget and fluctuating channel conditions. To solve the problem, the Lyapunov optimization method is used to transform the long-term optimization to be a real-time optimization problem. Additionally, we design an interactive Asynchronous Advantage Actor-Critic (IA3C) algorithm to solve the problem in an online fashion. Our results show that the proposed scheme and the IA3C algorithm can substantially reduce the computing load of VR terminals while maintaining a low MEC operational cost in a high convergence rate.**

***Index Terms*—Wireless VR, operational cost, computation offloading, task migration, Lyapunov optimization, A3C.**

I. Introduction

VIRTUAL reality (VR) technology has ushered in a new era with the emergence of the metaverse [1]. Prosperous VR applications will change people's way of life greatly [2]. However, the traditional wired and computer-assisted VR terminals (VTs) limit the scope of VR applications. To overcome this limitation, wireless connected VTs have been developed to offer users an immersive VR experience anytime and anywhere [3]. As a result, wireless VR has attracted wide attention in many fields.

The immersive VR experience relies on the projection of pixels from a viewing sphere onto a 2D plane. This process is called viewport rendering, which requires substantial computation and power consumption at VTs. However, the limitations of battery capacity and computing resources persist as bottlenecks for wireless VTs. Addressing this challenge, multi-access edge computing (MEC) plays a pivotal role in supporting wireless VR. By deploying MEC servers at edge networks, e.g., base stations (BSs), VTs can offload heavy tasks to nearby BSs, with the goal of reducing local computing load [4]. There have been several studies on MEC-enabled wireless VR, including the security and trust issues [5]–[7], the integration of caching and offloading for VR [8], and UAV-assisted VR task offloading [9]. Some works have addressed the data-correlation problem for content delivery and task computing. In [10], data-correlation-based task offloading and resource allocation were studied, while [11] focused on reducing uplink and downlink traffic with potential VR data-correlation. For smart resource management, reference [12] proposed a multi-agent decision-making algorithm, which can realize proactive link prediction and resource assignment. Furthermore, an asynchronous advantage actor-critic (A3C) algorithm was studied in [13] to address the dynamics of the wireless fading channel and the processing queues. Other studies concentrated on optimizing the delay and smoothness of VR videos [14]–[17].

Although MEC has become an effective solution for wireless VR, some problems still need to be addressed. Firstly, traditional nearby offloading policy prioritizes the VR rendering tasks onto the nearest MEC node. However, this can lead to overloading on certain MEC nodes, resulting in a deterioration of VR QoS. Indeed, computing tasks can be migrated between MEC nodes to achieve load balancing, implying that task offloading and migration could be jointly optimized for wireless VR. Secondly, balancing the computing load among MEC nodes necessitates frequent cross-edge task migrations, incurring significant costs such as the usage of expensive cross-BS bandwidth and energy consumption. Therefore, negotiating a performance-cost tradeoff is indispensable. Thirdly, due to the time-varying wireless channel conditions, ensuring satisfactory VR QoS requires timely and adaptable optimization. How to realize an efficient and adaptive decision-making algorithm for task offloading and migration remains to be addressed

In this paper, we focus on the negotiation of performance and cost for wireless VR. We propose a cost-aware task offloading and migration scheme, where the viewport rendering tasks of VR can be offloaded and migrated among the MEC-enabled access points (MAPs) under a tolerant operational cost. The main contributions are outlined as follows:

- We formulate the task offloading, migration, and subchannel allocation as an optimization problem, considering fluctuating channel conditions and a limited MEC cost budget. Our objective is to minimize VTs' power con-

This work was supported in part by the National Nature Science Foundation of China under Grant 62201271 and Grant 62303232, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20220441, in part by the China Postdoctoral Science Foundation under Grant 2023M741781, in part by the National Key R&D Program of China under Grant 2022YFB2902100, and in part by the the Basic Science Research Project of Jiangsu Colleges and Universities of China under Grant 22KJB510003 and Grant 22KJB120006. *(Corresponding author: Yan Liu.)*

Peng Lin, Yan Liu, and Zhizhong Zhang are with Nanjing University of Information Science and Technology, Nanjing 210044, China. (e-mail: linpeng@nuist.edu,cn, liuyan@nuist.edu.cn, zhangzz@nuist.edu.cn).

F. Richard Yu is with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: richard.yu@carleton.ca).

V. C. M. Leung is with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: vleung@ieee.org).

sumption while ensuring smooth VR streaming within a long-term cost budget. To our knowledge, this is the first study that explores the integration issues of task offloading, migration, and MEC cost-performance negotiation for wireless VR.

- To solve this problem, we use the Lyapunov optimization technique to incorporate the long-term cost budget into real-time optimizations. Additionally, we design a novel interactive A3C (IA3C) algorithm for the optimization problem to facilitate efficient decision-making.
- The simulation results demonstrate that the IA3C algorithm exhibits good convergence, and the proposed scheme effectively reduces the computing load of VTs while maintaining strict cost control.

The remainder is organized as follows. Section II presents the system model and problem formulation, followed by the problem transformation and the IA3C algorithm in Section III. In Section IV, the convergence and the system performance are evaluated. Section V draws conclusions.

II. System Model and Problem Formulation

We consider that M MAPs serve a set of on-ground mobile VTs. The time horizon is divided into time slots $t \in 1, 2, ..., T$. The set of VTs associated with the i-th MAP is denoted by $\mathcal{V}_i = \{1, 2, ..., K_i\}$, where K_i is the number of VTs connected to MAP i. To facilitate efficient content delivery, the content provider (CP) divides VR content into equal-sized low-resolution video chunks. When VT k requests a VR chunk, the requested chunk (chunk k) is initially delivered from the CP to MAP i. Subsequently, MAP i can either directly transmit chunk k to VT k or assist VT k in rendering the video chunks into full-resolution 360° viewports, depending on the load status of both the MAP and the VT. Additionally, each MAP actively explores migration opportunities with other MAPs to enhance the computation utilization efficiency.

A. VR Videos Request Model

We assume that each VT continually requests VR content from a finite library denoted as $\mathcal{F}$ during each time slot. The Orthogonal Frequency Division Multiple Access (OFDMA)-based networks are used for VR video streaming. The network allocates N downlink subcarriers for data downloads by VTs. We define $\gamma_{ik}^n(t)$ as the Signal-to-Interference-Plus-Noise Ratio (SINR) between VT k and MAP i on subcarrier n. Subsequently, the data transmission rate from MAP i to VT k on subcarrier n is calculated as:

$$R_{ik}^n(t) = B_n \log_2(1 + \gamma_{ik}^n(t)), \tag{1}$$

where B_n is the bandwidth allocated to subchannel n. $\gamma_{ik}^n(t)$ is a time-varying variable, and it conforms to the Jakes model [18], which characterizes the channel correlation between time slots t and $t-1$ as:

$$\gamma_{ik}^n(t) = \rho_{\mathrm{DL},k}\gamma_{ik}^n(t-1) + \sqrt{1-\rho_{\mathrm{DL},k}^2}\sigma_{\mathrm{DL},k}, \tag{2}$$

where $\rho_{\mathrm{DL},k}$ is the correlation coefficient of VT k, and $\sigma_{\mathrm{DL},k} \sim \mathcal{CN}(0,1)$ is a Gaussian random variable. Then, the content transmit rate from MAP i to VT k can be calculated as:

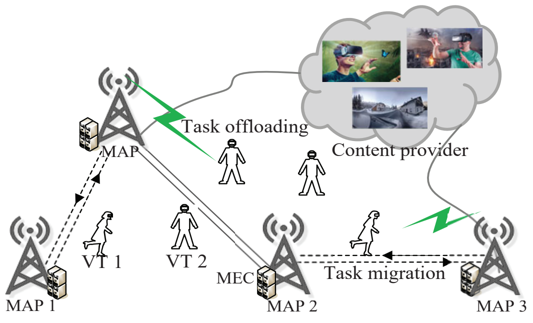

Fig. 1. Illustration of task offloading and migration for wireless VR.

$$R_{ik}(t) = \sum_{n=1}^{N} \iota_{ik}^n(t) R_{ik}^n(t), \tag{3}$$

where $\iota_{ik}^n(t) = 1$ indicates that subcarrier n is allocated to VT k served by MAP i at time slot t, otherwise $\iota_{ik}^n(t) = 0$.

B. Task Offloading and Migration Model

In the system, each VT generates independent content requests with time slots $t = 1, 2, \ldots, \infty$. The video chunk requested by VT k has a content size o_k and a data conversion factor η_k. Parameter η_k denotes the ratio of the size of output data size to the input data for viewport rendering, and it varies for different VTs, depending on video quality required by VT k. Use the binary variable $x_{ik}(t)$ to indicate whether chunk (task) k is computed at MAP i. Then, the task addressing process at the VT and the MAP are analyzed as below.

1) Task Computing at VTs: Let c_k (CPU cycles/s) denote the computing capacity of VT k. Denote ζ_k (bits/cycle) as the data volume that one CPU cycle of VT k can process. Since task k is computed at VT k locally, i.e., $x_{ik}(t) = 0$, the data volume to be obtained per unit time for VT k is calculated as

$$v_k(t) = x_{ik}(t)R_{ik}(t) + (1 - x_{ik}(t))c_k\zeta_k, \tag{4}$$

where the first term is the rendered data received from MAP i, and the second term is the rendered data at VT k locally. Assume that VT k has a video playback buffer $b_k(t)$, then its status at time slot t depends on its buffer throughput at time slot $t-1$ as:

$$b_k(t) = b_k(t-1) + \Delta\tau v_k(t) - \Delta\tau v_0, \tag{5}$$

where $\Delta\tau$ is the playback time window, v_0 is the required data rate for VR video playing. To ensure a smooth VR video streaming, we have:

$$b_k(t) \leq b_0. \tag{6}$$

2) Task Offloading and Migration: The computing load of MAP depends on the task offloading quantity and the task migration status between MAPs. Denote $z_{irk}(t)$ as the task migration decision that is 1 if task k is migrated from MAP i to r at time slot t, and 0 otherwise. The total number of tasks $N_i(t)$ to be computed at MAP i can be calculated as:

$$N_i(t) = \sum_{k=1}^{K_i} x_{ik}(t) - \sum_{r=1}^{M}\sum_{k=1}^{K_i} z_{irk}(t) + \sum_{j=1}^{M}\sum_{k=1}^{K_i} z_{jik}(t), \tag{7}$$

where the second term is the output tasks migrated to other MAPs, and the third term is the input tasks coming from other MAPs. Each MAP migrates its task to just one other MAP when its computing power is not enough. Then, we have:

$$x_{ik}(t) + \sum_{r=1}^{M} z_{irk}(t) \leq 1, \quad \forall i, \forall k. \tag{8}$$

We assume that each MAP has the same CPU configuration, i.e., C (CPU cycles/s) and ζ (bits/cycle). The computing capacity of MAP i for task k can be obtained as $c_{ik} = \zeta C/N_i(t)$. Then, the computing delay for task k at MAP i can be calculated as $T^c_{ik}(t) = o_k/c_{ik}$. Correspondingly, the migration delay of task k between MAP i and MAP r is calculated as $T^b_{irk}(t) = o_k/R^{\text{bh}}_{ir}(t) + o_k\eta_k/R^{\text{bh}}_{ir}(t)$, where $R^{\text{bh}}_{ir}(t)$ is the backhaul transmit rate between MAP i and r, and η_k is the data conversion factor for viewport rendering.

C. Long-Term Optimization with Performance-Cost Tradeoff

We use the power consumption to assess the computing load of VT. The power consumption can be computed by $\varrho = \kappa f^2$, where κ is switched capacitance parameter and f is the CPU frequency [19]. Then, for VT k, the power consumption of local computing can be calculated as

$$E^{loc}_{ik}(t) = \kappa c_k^2 o_k \zeta_k^{-1} + \frac{P_k o_k}{R_{ik}(t)}, \tag{9}$$

where P_k is the circuit power for receiving data. When the viewport rendering task is offloaded to MAP i, the power consumption of VT k is calculated as:

$$E^{edg}_{ik}(t) = \frac{P_k o_k \eta_k}{R_{ik}(t)}. \tag{10}$$

Therefore, based on the task offloading decision $x_{ik}(t)$, the total power consumption of VT k can be calculated as:

$$E^{tot}_{ik}(t) = x_{ik}(t)E^{edg}_{ik}(t) + (1 - x_{ik}(t))E^{loc}_{ik}(t). \tag{11}$$

Frequent task offloading and migration incur operational costs. The operational cost for MAP i comprises two parts: task computing cost and task migration cost. The former is denoted by g_0, which represents the expense of utilizing the MEC server per unit time. The latter encompasses the task computing expense g_0 on the migrated MAP and the cross-edge service cost g_s. When task k is migrated from MAP i to r, the cross-edge service cost g_s is calculated as the expense of utilizing the cross-MAP bandwidth per unit time. Given the decision variables $x_{ik}(t)$ and $z_{irk}(t)$, the total operational cost of MAP i for addressing task k can be calculated as:

$$\begin{aligned} \mathcal{C}_{ik}(t) = x_{ik}(t)\Big[&\sum_{r=1}^{M} z_{irk}(t)\left(T^c_{rk}(t)g_0 + T^b_{irk}(t)g_s\right) \\ &+ \Big(1 - \sum_{r=1}^{M} z_{irk}(t)\Big)T^c_{ik}(t)g_0\Big]. \end{aligned} \tag{12}$$

Since the MEC providers generally operate within a long-term cost budget, e.g., daily, the total operational cost incurred by utilizing MEC service resources should be within an acceptable bound during a period of time. We use $\mathcal{C}_{avg}$ to denote the long-term cost budget over time T, which satisfies

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{M} \sum_{k=1}^{K_i} \mathcal{C}_{ik}(t) \leq \mathcal{C}_{avg}. \tag{13}$$

Then, we formulate a long-term optimization problem under the constraint of long-term cost budget as follows:

$$\begin{aligned} \text{P1:} \quad &\min_{x_{ik}, z_{irk}, \iota^n_{ik}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K_i} E^{tot}_{ik}(t), \quad \forall i, \\ &\text{s.t.} \quad (6), (8), (13). \end{aligned} \tag{14}$$

III. Interactive A3C Enabled Task Offloading and Migration

A. Problem Transformation via Lyapunov Optimization

In this part, the Lyapunov optimization method is used to decouple the long-term optimization problem. To address the long-term constraint (13), we introduce a virtual queue $q(t)$ as the accumulated MEC operational cost, and the initial queue backlog is $q(0) = 0$,

$$q(t+1) = \max[q(t) + \sum_{i=1}^{M} \sum_{k=1}^{K_i} \mathcal{C}_{ik}(t) - \mathcal{C}_{avg}, 0], \tag{15}$$

where the value of $q(t)$ is the queue length, indicating the excess cost over the budget by the end of time slot t. To meet the long-term constraint (13), the queue $q(t)$ should be stable, i.e., $\lim_{T\to\infty} \mathbb{E}\{q(T)\}/T = 0$. Define a Lyapunov drift function $L(\Theta_t) = q^2(t)/2$. The stability of queue q_t can be achieved by minimizing the one-step conditional Lyapunov drift function $\Delta(\Theta_t)$, where

$$\Delta(\Theta_t) \triangleq \mathbb{E}\left[L(\Theta_{t+1}) - L(\Theta_t)|\Theta_t\right]. \tag{16}$$

The drift function $\Delta(\Theta_t)$ describes the dynamics of the virtual queue $q(t)$ in the Lyapunov function over one time slot.

Lemma 1: For any decision $x_{ik}(t)$ and $z_{irk}(t)$, the value of $\Delta(\Theta_t)$ holds the following statements:

$$\Delta(\Theta_t) \leq \frac{1}{2}\digamma_{max} + q(t)\mathbb{E}[\digamma_t], \tag{17}$$

where

$$\digamma_{max} = \mathcal{C}^2_{max} + \mathcal{C}^2_{avg}, \ \digamma_t = \sum_{i=1}^{M} \sum_{k=1}^{K_i} \mathcal{C}_{ik}(t) - \mathcal{C}_{avg}. \tag{18}$$

Proof: According to equation (15), we can obtain the boundary of Lyapunov drift as follows:

$$\begin{aligned} q^2(t+1) &= \left(\max\left[q(t) + \sum_{i=1}^{M} \sum_{k=1}^{K_i} \mathcal{C}_{ik}(t) - \mathcal{C}_{avg}, 0\right]\right)^2 \\ &\leq q^2(t) + \left(\sum_{i=1}^{M} \sum_{k=1}^{K_i} \mathcal{C}_{ik}(t) - \mathcal{C}_{avg}\right)^2 \\ &\quad + 2q(t)\left(\sum_{i=1}^{M} \sum_{k=1}^{K_i} \mathcal{C}_{ik}(t) - \mathcal{C}_{avg}\right) \\ &= q^2(t) + \digamma_t^2 + 2q(t)\digamma_t, \end{aligned} \tag{19}$$

where $\digamma_t = \sum_{i=1}^{M}\sum_{k=1}^{K_i}\mathcal{C}_{ik}(t) - \mathcal{C}_{avg}$. Rearranging $q^2(t)$ to the left of the equation (19), we can obtain:

$$\mathbb{E}\Big[q^2(t+1) - q^2(t)|\Theta_t\Big] \le \mathbb{E}\Big[\digamma_t^2 + 2q(t)\digamma_t|\Theta_t\Big]. \quad (20)$$

For function $\digamma_t$, there exist a constant value $\digamma_{max} > 0$ and satisfying the following inequation:

$$\Delta(\Theta_t) \le \frac{1}{2}\digamma_{max} + q(t)\mathbb{E}[\digamma_t|\Theta_t], \quad (21)$$

where $\digamma_{max} = \mathcal{C}_{max}^2 + \mathcal{C}_{avg}^2$, and $\mathcal{C}_{max} = \max_{t\in T}\sum_{i=1}^{M}\sum_{k=1}^{K_i}\mathcal{C}_{ik}(t)$. ■

By incorporating the queue stability into the objective of P1, our strategy becomes to minimize the upper bound of the Lyapunov drift-plus-penalty function as below:

$$\begin{aligned}&\Delta(\Theta_t) + \mu\sum_{i=1}^{M}\sum_{k=1}^{K_i}E_{ik}^{tot}(t)\\ &\quad\le \frac{1}{2}\digamma_{max} + q(t)\mathbb{E}[\digamma_t] + \mu\sum_{i=1}^{M}\sum_{k=1}^{K_i}E_{ik}^{tot}(t)\end{aligned} \quad (22)$$

where μ is a non-negative parameter that controls the tradeoff between power consumption and MEC operational cost.

B. Interactive A3C Algorithm

To solve the formulated problem efficiently, we resort to an improved A3C algorithm, which named *Interactive A3C* (IA3C) algorithm. We first transform the optimization problem into a Markov Decision Process (MDP):

a) State:

$$s_{i,t} = \{N_i(t), \gamma_{ik}^n(t)\}, \quad \forall s_{i,t}\in\mathcal{S}, \quad (23)$$

where $\mathcal{S}$ is the state space including all the combinations of channel conditions $\gamma_{ik}^n(t)$ and computation load $N_i(t)$.

b) Action:

$$a_{i,t} = \{x_{ik}(t), z_{irk}(t), \iota_{ik}^n(t)\}, \quad \forall a_{i,t}\in\mathcal{A}, \quad (24)$$

where $\mathcal{A}$ is the action space including all the combinations of task offloading decision $x_{ik}(t)$, task migration decision $z_{irk}(t)$, and subcarrier allocation decision $\iota_{ik}^n(t)$.

c) Reward: The system reward should correspond with P1' objective to ensure convergence in a desired direction. Since P1 has been transformed to minimize the upper bound of the Lyapunov drift-plus-penalty function. Therefore, the system reward is $\mathcal{R}(s_{i,t}, a_{i,t}) = -U_t^\pi(x_{ik}, z_{irk}, \iota_{ik}^n)$, where

$$U_t^\pi(x_{ik}, z_{irk}, \iota_{ik}^n) = q(t)\digamma_t + \mu\sum_{i=1}^{M}\sum_{k=1}^{K_i}E_{ik}^{tot}(t). \quad (25)$$

$\mathcal{R}(s_{i,t}, a_{i,t})$ indicates the tradeoff between power consumption and MEC cost for taking $a_{i,t}$ at $s_{i,t}$ under policy π.

To train the MDP effectively, the A3C algorithm approximates the policy function $\pi(s_{i,t}, a_{i,t})$ and the value function $V(s_{i,t})$ using two parameterized neural networks as follows:

$$\pi(s_{i,t}, a_{i,t}) \approx \pi(a_{i,t}|s_{i,t};\theta) \quad (26)$$

$$V(s_{i,t}) \approx V(s_{i,t};\theta_v) \quad (27)$$

Algorithm 1 Interactive A3C algorithm

```
1: Initialize: Global shared parameters $\theta$ and $\theta_v$, thread-specific
   parameters $\theta^{'}$ and $\theta_v^{'}$, thread step counter $t \leftarrow 1$, and $T = 0$.
2: repeat
3:    Reset gradients:$d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
4:    Synchronize thread-specific parameters $\theta^{'} = \theta$ and $\theta_v^{'} = \theta_v$.
5:    $t_{start} = t$, get state $s_t$.
6:    for $t - t_{start} \le t_{max}$ do
7:       Take action $a_t$ according to thread policy $\pi(a_t|s_t;\theta^{'})$.
8:       Update punishment factor $\mu$:
              $\mu^{[t+1]} = \mu^{[t]} - \varrho^{[t+1]}\frac{\partial U_\pi(x_{ik}, z_{irk}, \iota_{irk})}{\partial\mu}$.
9:       Receive reward $\mathcal{R}(s_t, a_t)$ and state $s_{t+1}$.
10:      Reconstruct reward $\overline{\mathcal{R}}(s_t, a_t)$ according to (29).
11:      $t \leftarrow t + 1$.
12:   end for
13:   $\mathcal{R}_{cum} = V(s_t;\theta_v^{'})$.
14:   for $t_x \in \{t-1, ..., t_{start}\}$ do
15:      $\mathcal{R}_{cum} = \overline{\mathcal{R}}(s_{t_x}, a_{t_x}) + \alpha\mathcal{R}_{cum}$
16:      Accumulate gradients wrt. $\theta^{'}$ according to (31).
17:      Accumulate gradients wrt. $\theta_v^{'}$ according to (30).
18:   end for
19:   Perform asynchronous update of $\theta$ and $\theta_v$ using $d\theta$ and $d\theta_v$.
20: until $T \ge T_{max}$
```

For simplicity, the index i of MAP is omitted in the following presentation. Then, for each training thread, construct a k-step reward. The k-step reward of the n-th thread is written as:

$$\mathcal{R}_{n,k} = \sum_{l=0}^{k-1}\alpha^l\mathcal{R}(s_{t+l}, a_{t+l}) + \alpha^k V(s_{t+k};\theta_v^{'}), \quad (28)$$

where α is the attenuation factor, and $\theta_v^{'}$ is the thread-specific parameter. Step k varies with different states and its maximum value is t_{max}. Reward $\mathcal{R}_{n,k}$ is used to construct the accumulate gradients for updating the main networks. Different from traditional A3C, in this paper, we propose an IA3C algorithm where multiple training threads can interact with their learning experience during the learning process. Specifically, the reward $\mathcal{R}(s_t, a_t)$ in (28) is reconstructed with the *Exploration* and *Extension* incentives as:

$$\begin{aligned}\overline{\mathcal{R}}(s_t, a_t) = \mathcal{R}(s_t, a_t) + \underbrace{\epsilon_{ep}\mathbb{E}\{D_{KL}(\pi_{act}||\pi(a|s;\theta^{'})\}}_{\text{Exploration}}\\ - \underbrace{\epsilon_{et}\mathbb{E}\{D_{KL}(\pi_{act}||\widetilde{\pi}(a|s;\theta^{'})\}}_{\text{Extension}},\end{aligned} \quad (29)$$

where π_{act} is the policy that makes the thread take actual actions. In equation (29), the exploration and extension items enable a thread to interact reward with other thread. The average KL-divergence of the parameterized policy $\pi(a|s;\theta^{'})$ from policy π_{act} is used to construct the exploration incentive. It encourages the a thread to select actions that lead to states unfamiliar from its past experiences. For instance, when $\pi(a|s;\theta^{'})$ significantly differs from π_{act} in states that the thread has not previously visited, the reward function $R(s_t, a_t)$ will augment an additional reward. Similarly, the extension incentive mitigates the inconsistency between different thread's decisions by applying KL divergence between its policy and the policies of other threads to the system reward.

Then, we the actor and the critic networks can be updated through the accumulated gradient, which are expressed as:construct the k-step accumulate gradients:

$$d\theta_v = d\theta_v + \frac{\partial(\mathcal{R}_{n,k} - V(s_t;\theta_v^{'}))^2}{\partial\theta_v^{'}}. \tag{30}$$

$$d\theta = d\theta + \nabla_{\theta'} \log \pi(a_t|s_t;\theta^{'})(\mathcal{R}_{n,k} - V(s_t;\theta_v^{'})). \tag{31}$$

Algorithm 1 shows the detailed IA3C algorithm. At the beginning, parameters θ' in different training threads are initialized. In the training process, the agent in one thread first takes action a_t under policy $\pi(a_t|s_t;\theta')$. Then, the subgradient method is used to update the punishment factor μ by iterating with decision variables $x_{ik}(t)$, $z_{irk}(t)$, and $\iota_{irk}(t)$ (steps 8-9). Based on the obtained immediate reward $R(s_t, a_t)$, the system reward $\overline{\mathcal{R}}(s_t, a_t)$ is reconstructed according to (29). After a continuous time duration of $t_{\max}$, the cumulated reward $\mathcal{R}_{n,k}$ can be obtained according to (28). Afterward, the main parameter θ and the thread-specific parameter θ_v' are updated successively according to (30) and (31). Ultimately, the policy and parameter μ will converge to be stable as the system continues to run.

IV. Simulation Results

The simulation environment is implemented using Python Anaconda 3.7. The simulation parameters are set with common configurations inspired from [20]. The main network is created with 3 fully connected feed-forward hidden layers. The neurons in three layers are set as 256, 256, and 512, respectively. The action space consists of 1256 effective actions as the output nodes of the actor network. The rectified linear units is used as the activation function. The user environment is a 500×500 m^2 geographic area with 10 MAPs. The CPU frequency of the VT and the MAP are set as c_k=0.5 GHz and C = 3.5 GHz, respectively. The computing capacity of the CPUs for the VT and the MAP are set as ζ_k = 5 Kb/cycle and ζ = 30 Kb/cycle, respectively. The circuit power of VT is set to P_k = 100 mW. The VR chunk size is set at 300 Mb uniformly. The switched capacitance is set as $\kappa = 10^{24}$. The downlink bandwidth of each subcarrier is 15 MHz. The dynamics of wireless channel conditions follows the Jakes model, with the correlation coefficient ρ set to 0.7. The data conversion factor η_k is uniformly set as 2. The MEC cost is inversely proportional to the available computing and transmit capacities of MAPs, and the values are perturbed by a random parameter in the range [1, 1.35] of the minimal power consumption parameter [21].

Fig. 2 evaluates the convergence performance of the proposed IA3C algorithm compared with the traditional A3C algorithm used in [13] and the Actor-Critic algorithm. The Actor-Critic algorithm serves as the baseline model. In the A3C algorithm, the reinforcement learning is executed on three threads, and each thread-specific training performs independently. We observe that all the simulated algorithms can make the optimization problem converge to stability. Compared with the Actor-Critic algorithm, the A3C algorithm demonstrates the advantages of multi-threading in terms of convergence rate and accuracy. The proposed IA3C uses *Exploration* and *Extension* between different threads to interact learning experience, which exhibits faster convergence speed and yields closer decision results than the A3C algorithm.

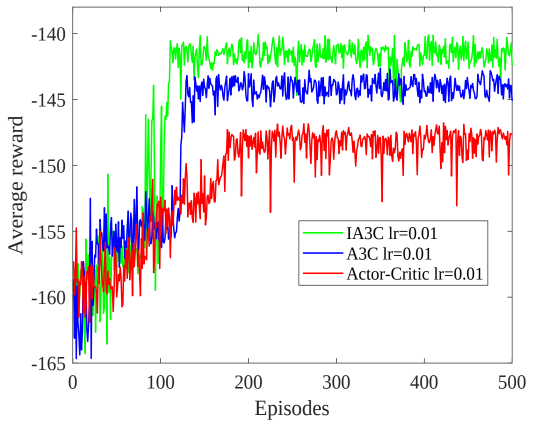

Fig. 2. Convergence performance compared with traditional RL.

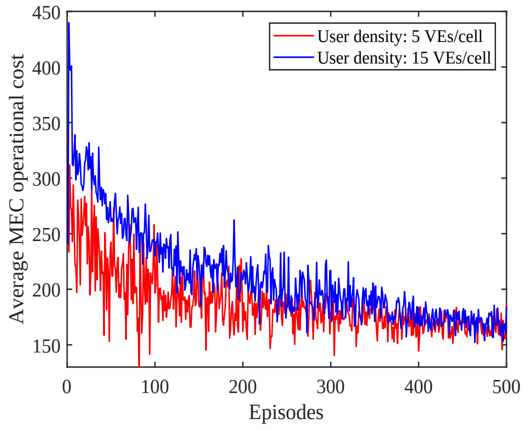

Fig. 3. Evaluation of the average MEC operational cost.

Fig. 3 illustrates the change in task addressing cost with the training episodes of the IA3C algorithm under different user densities. It can be observed that a larger user density induces more migration costs to stabilize system performance. Meanwhile, we observe that the average operational cost in both scenes decreases significantly and eventually converges to the long-term cost budget. Just as we have discussed that the operational cost will not exceed the budget when there is a stable cost queue: $\lim_{T\to\infty} \mathbb{E}q(T)/T = 0$.

Fig. 4 illustrates the average power consumption of VTs under the increase of the computing capacity of MAPs. The power consumption of VTs is an indicator of how task offloading and migration contribute to reducing the computing load on VEs. To demonstrate the experimental results sufficiently, four schemes are configured for comparison. The *local computing* scheme serves as a baseline model, in which VTs' tasks are computed locally, and the MAP only performs subchannel allocations. The *greedy offloading* scheme allows the MAP to compute viewport rendering for VTs based on their access sequences. The *offloading without migration* scheme is derived from references [10], [22], where the objective is to mini-

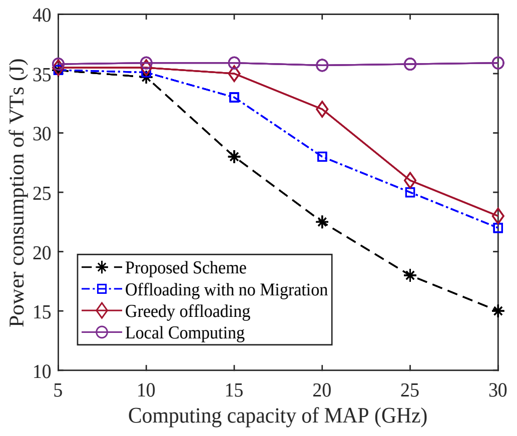

Fig. 4. The power consumption versus computing capacity compared with existing schemes

mize the system latency through task offloading optimization without considering task migrations. Compared with *greedy offloading*, the *offloading without migration* scheme holds better power consumption, which highlights the effectiveness of the dynamic optimization for task offloading. The proposed scheme is capable of maintaining minimal energy consumption under limited computing capacity, demonstrating that a well-regulated task offloading and migration scheme can effectively enhance the service capability of MEC for wireless VR.

V. Conclusion

In this paper, we proposed a cost-aware task offloading and migration scheme in which the viewport rendering tasks of VTs can be regularly offloaded and migrated among MAPs under an acceptable long-term operational cost. Considering dynamic wireless transmission environments, the Lyapunov optimization method was used to transform the original problem into a series of real-time decision problems. Then, an IA3C algorithm was proposed to realize online decision-making with a high convergence rate. Simulation results showed the advantages of our scheme in terms of convergence rate, VTs' computing load, and the operational cost of MEC systems.

References

[1] P. Lin, Q. Song, F. R. Yu, D. Wang, A. Jamalipour, and L. Guo, "Wireless virtual reality in beyond 5g systems with the internet of intelligence," *IEEE Wireless Communications*, vol. 28, no. 2, pp. 70–77, 2021.

[2] P. Rosedale, "Virtual reality: The next disruptor: A new kind of worldwide communication," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 48–50, 2017.

[3] E. Bastug, M. Bennis, M. Medard, and M. Debbah, "Toward interconnected virtual reality: Opportunities, challenges, and enablers," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 110–117, 2017.

[4] Y. Sun, J. Chen, Z. Wang, M. Peng, and S. Mao, "Enabling mobile virtual reality with open 5g, fog computing and reinforcement learning," *IEEE Network*, vol. 36, no. 6, pp. 142–149, 2022.

[5] Y. Xu, H. Zhang, X. Li, F. R. Yu, V. C. Leung, and H. Ji, "Trusted collaboration for mec-enabled vr video streaming: A multi-agent reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, pp. 1–14, 2023.

[6] P. Lin, Q. Song, F. R. Yu, D. Wang, and L. Guo, "Task offloading for wireless vr-enabled medical treatment with blockchain security using collective reinforcement learning," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15749–15761, 2021.

[7] Y. Xu, H. Zhang, X. Li, F. R. Yu, H. Ji, and V. C. Leung, "Blockchain-based edge collaboration with incentive mechanism for mec-enabled vr systems," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2023.

[8] Z. Gu, H. Lu, and C. Zou, "Horizontal and vertical collaboration for vr delivery in mec-enabled small-cell networks," *IEEE Communications Letters*, vol. 26, no. 3, pp. 627–631, 2022.

[9] L. Zhang and J. Chakareski, "Uav-assisted edge computing and streaming for wireless virtual reality: Analysis, algorithm design, and performance guarantees," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 3267–3275, 2022.

[10] P. Lin, Q. Song, D. Wang, F. R. Yu, L. Guo, and V. C. M. Leung, "Resource management for pervasive-edge-computing-assisted wireless vr streaming in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7607–7617, 2021.

[11] M. Chen, W. Saad, C. Yin, and M. Debbah, "Data correlation-aware resource management in wireless virtual reality (vr): An echo state transfer learning approach," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4267–4280, 2019.

[12] Z.-Y. Wu, M. Ismail, E. Serpedin, and J. Wang, "Artificial intelligence for smart resource management in multi-user mobile heterogeneous rf-light networks," *IEEE Wireless Communications*, vol. 28, no. 4, pp. 152–158, 2021.

[13] J. Du, W. Cheng, G. Lu, H. Cao, X. Chu, Z. Zhang, and J. Wang, "Resource pricing and allocation in mec enabled blockchain systems: An A3C deep reinforcement learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 33–44, 2022.

[14] P. Lin, Z. Ning, Z. Zhang, Y. Liu, F. R. Yu, and V. C. M. Leung, "Joint optimization of preference-aware caching and content migration in cost-efficient mobile edge networks," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2023.

[15] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "Mec-assisted immersive vr video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9517–9529, 2020.

[16] Y. Liu, J. Liu, A. Argyriou, L. Wang, and Z. Xu, "Rendering-aware vr video caching over multi-cell mec networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2728–2742, 2021.

[17] J. Du, Z. Kong, A. Sun, J. Kang, D. Niyato, X. Chu, and F. R. Yu, "Maddpg-based joint service placement and task offloading in mec empowered air-ground integrated networks," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[18] Y. Hu, A. Schmeink, and J. Gross, "Optimal scheduling of reliability-constrained relaying system under outdated csi in the finite blocklength regime," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6146–6155, 2018.

[19] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282.

[20] Y. Xu, H. Zhang, X. Li, F. R. Yu, V. C. Leung, and H. Ji, "Trusted collaboration for mec-enabled vr video streaming: A multi-agent reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 9, pp. 12167–12180, 2023.

[21] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[22] D. Wang, Y. Bai, G. Huang, B. Song, and F. R. Yu, "Cache-aided mec for iot: Resource allocation using deep graph reinforcement learning," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11486–11496, 2023.