Rate-Splitting for IRS-Aided Multiuser VR Streaming: An Imitation Learning-based Approach

Rui Huang*, Vincent W.S. Wong*, and Robert Schober†
*Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
†Institute for Digital Communications, Friedrich-Alexander University of Erlangen-Nürnberg, Germany
email: {ruihuang, vincentw}@ece.ubc.ca, robert.schober@fau.de

Abstract—**Virtual reality (VR) applications require wireless systems to provide a high transmission rate to support 360-degree video streaming to multiple users simultaneously. In this paper, we propose an intelligent reflecting surface (IRS)-aided rate-splitting (RS) VR streaming system. In the proposed system, RS exploits the shared interests of the users in VR streaming, and the IRS creates reflected channels to facilitate a high transmission rate. The IRS also mitigates the performance bottleneck caused by the requirement that all RS users have to be able to decode the common message. We formulate an optimization problem for maximization of the achievable bitrate of the streamed 360-degree video subject to the quality-of-service (QoS) constraints of the users. We propose a deep reinforcement learning (DRL)-based algorithm, in which we leverage imitation learning and the hidden convexity of the formulated problem to optimize the IRS phase shifts, RS parameters, beamforming vectors, and bitrate selection of the 360-degree video tiles. Simulations based on a real-world dataset show that the proposed IRS-aided RS VR streaming system outperforms two baseline schemes in terms of system sum-rate and average runtime.**

I. Introduction

Virtual reality (VR) streaming provides the users with an immersive experience by rendering 360-degree videos using head-mounted devices (HMDs). The growing demand for VR streaming introduces new challenges for current wireless systems since the bitrate of a 360-degree video can be much higher than that of conventional multimedia applications. Hence, wireless systems have to be able to support a very high data transmission rate to meet the requirement of 360-degree video streaming.

In multiuser VR streaming, the same 360-degree video segment may be requested by multiple users due to their shared interests. As an example, for the streaming of a 360-degree soccer match video, the supporters of a particular soccer team may frequently request those video tiles that include the players of their team. Rate-splitting (RS) is a physical layer technique in which the information intended for the users is split into two parts, namely a common message and private messages [1], [2]. In RS, each user needs to decode the common message first. The user then subtracts the common message from the received signal using successive interference cancellation (SIC) and subsequently decodes its private message [1]. These features of RS make it a promising technique for multiuser VR streaming systems since (a) the data related to the shared interests of the users can be encoded into the common message to achieve multiplexing gains, and (b) the unique data requested by each user can be encoded in its private message to ensure that all the requested video tiles can be received. Most existing works studied RS systems where the data for different users are independent and uncorrelated, see, e.g., [2], [3], while the shared interests of the users have not been explored. The authors in [4] considered an RS non-orthogonal unicast and multicast (RS-NOUM) system, where a multicast message needs to be received by all the users in the system and each user's private message is being sent via unicast. The authors in [5] studied RS multigroup multicast systems, in which the same message is requested by the users of the same group. Although the RS-based multicasting schemes considered in [4], [5] exploit the multiplexing gain, they assumed that a part of the information is requested by every user in the system or in the same group. However, this assumption may not always hold in RS VR streaming systems when some users do not request the same video tile. Therefore, a new RS scheme needs to be designed for multiuser VR streaming systems to tackle this issue and exploit the shared interests of the users.

Besides RS, we propose to deploy an intelligent reflecting surface (IRS) to improve the performance of the RS system. IRSs are reconfigurable planar surfaces with a large number of passive reflecting elements. IRSs can effectively improve the minimum signal-to-noise-plus-interference ratio (SINR) in RS systems because users who suffer from a large path loss can benefit from the additional propagation channels created by the IRSs. IRSs also introduce additional degrees of freedom (DoF) that can be exploited to mitigate interference [6]. The authors in [3] designed an on-off phase shift control scheme for an IRS-aided rate-splitting multiple access (RSMA) system. The authors in [7] proposed an alternating optimization (AO) algorithm to maximize the minimum achievable rate of an IRS-aided RSMA system. However, multiuser VR streaming systems was not studied in [3], [7]. In IRS-aided RS VR streaming systems, the joint optimization of the IRS phase shifts, RS parameters, beamforming vectors, and bitrate selection of the video tiles based on the video tile requests and channel state information (CSI) of the VR users is crucial for achieving a high performance. Therefore, the results reported in [3], [7] are not applicable to the problem investigated in this paper.

In this paper, we propose an IRS-aided RS VR streaming system, where RS is applied to exploit the shared VR streaming interests of the users, and IRSs are used to improve the minimum SINR experienced by the common message across the users. We aim to maximize the achievable bitrate of the 360-degree video subject to the quality-of-service (QoS) requirements of the users. We propose a deep reinforcement learning (DRL)-based algorithm to solve the formulated problem in a computationally efficient manner. The contributions of this paper are as follows:

- We propose an IRS-aided RS VR streaming system, and formulate an optimization problem for maximization of the achievable bitrate of the 360-degree video. In particular, we aim to jointly optimize the beamforming vectors, IRS phase shifts, RS parameters, and bitrates of the 360-degree video tiles.
- We propose a DRL-based algorithm, in which imitation learning and actor-critic method are exploited to solve the formulated problem. The proposed algorithm can learn the policy by leveraging both exploration and solutions obtained with conventional optimization methods.
- We design a DNN module in which differentiable convex optimization (DCO) layers [8] is applied to tackle the convex constraints of the formulated problem.
- Simulations based on the VR streaming dataset from [9] show that the proposed algorithm outperforms two baseline schemes [4], [6] in terms of the system sum-rate and average runtime.

The remainder of this paper is organized as follows. The system model and problem formulation for IRS-aided RS VR streaming systems are presented in Section II. In Section III, we develop the proposed DRL-based algorithm. Simulation results are presented in Section IV. Conclusions are drawn in Section V.

Notations: We use upper-case and lower-case boldface letters to denote matrices and column vectors, respectively. $\mathbb{C}^{M \times N}$ denotes the set of $M \times N$ complex-valued matrices. $\boldsymbol{A}^H$ denotes the conjugate transpose of matrix $\boldsymbol{A}$. $\text{vec}(\boldsymbol{A})$ returns a vector obtained by stacking the columns of matrix $\boldsymbol{A}$. $\text{diag}(\boldsymbol{x})$ returns a diagonal matrix where the diagonal elements are given by the elements of vector $\boldsymbol{x}$. $\sim$ means "distributed as". $\mathbb{E}[\,\cdot\,]$ represents statistical expectation. $\mathbb{1}(\cdot)$ denotes the indicator function, which is equal to 1 if its argument is true and equal to 0 otherwise.

II. IRS-Aided RS VR Streaming System and Problem Formulation

As shown in Fig. 1, one base station and an IRS are deployed in an indoor facility to provide VR streaming service to N users. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of users. The base station has N_t antennas, while the IRS has L reflecting elements. The HMD of each user has one antenna. We denote $\phi_l \in [0, 2\pi)$, $l \in \{1, \ldots, L\}$, as the phase shift of the l-th reflecting element of the IRS. Time is slotted into intervals of equal duration. Let $\mathcal{T} = \{1, 2, \ldots\}$ denote the set of time slots. The time interval $[t, t+1)$ is referred to as time slot $t \in \mathcal{T}$. The channel gain between the base station and user $n \in \mathcal{N}$ in time slot t is denoted by $\boldsymbol{h}_{n,D}(t) \in \mathbb{C}^{N_t}$. The channel gain between the base station and the IRS in time slot t is denoted by $\boldsymbol{G}(t) \in \mathbb{C}^{L \times N_t}$. The phase shift matrix of the IRS in time slot t is denoted by $\boldsymbol{\Psi}(t) = \text{diag}(e^{j\phi_1(t)}, \ldots, e^{j\phi_L(t)})$. The channel gain between the IRS and user $n \in \mathcal{N}$ in time slot t is denoted as $\boldsymbol{h}_{n,R}(t) \in \mathbb{C}^L$. In order to investigate the performance upper bound of the considered system, we assume that perfect CSI can be obtained by the base station.

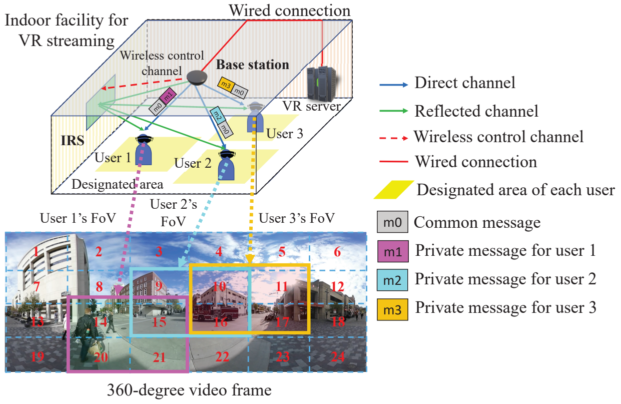

Fig. 1. An IRS-aided RS VR streaming system. The upper part of the figure shows an indoor facility for VR streaming. The lower part of the figure illustrates a 360-degree video frame.

Each 360-degree video frame is divided into $I_{\max}$ video tiles. We define $\mathcal{I} = \{1, 2, \ldots, I_{\max}\}$. The indices of tiles requested by user n in time slot t are collected in set $\mathcal{I}_n(t)$. We define $\mathcal{I}(t) = \bigcup_{n \in \mathcal{N}} \mathcal{I}_n(t)$, $t \in \mathcal{T}$. At the beginning of time slot $t \in \mathcal{T}$, user $n \in \mathcal{N}$ informs the base station about $\mathcal{I}_n(t)$. After receiving the video tile requests from the users, the base station needs to determine the bitrate of each requested video tile. We assume there are M available bitrate selections of the video tiles, i.e., $v_1 < \cdots < v_M$. Let $v_{n,i}(t)$ denote the bitrate of tile $i \in \mathcal{I}_n(t)$ requested by user n in time slot t. We have

$$\text{C1:} \quad v_{n,i}(t) \in \mathcal{V} = \{v_1, \ldots, v_M\},\ i \in \mathcal{I}_n(t),\ n \in \mathcal{N}, \quad (1)$$

and define $\boldsymbol{v}_n(t) = (v_{n,i}(t),\ i \in \mathcal{I}_n(t))$.

The utility obtained by user n in time slot t is given by

$$u_n(t) = \sum\nolimits_{i \in \mathcal{I}_n(t)} v_{n,i}(t) - \kappa^{\text{intra}} \ell_n^{\text{intra}}(t),\ n \in \mathcal{N}, \quad (2)$$

where $\ell_n^{\text{intra}}(t)$ is the intra-frame quality switch loss of user n in time slot t, and $\kappa^{\text{intra}} > 0$ is a scaling factor for $\ell_n^{\text{intra}}(t)$. $\ell_n^{\text{intra}}(t)$ is determined by the variance of the elements of vector $\boldsymbol{v}_n(t)$. The utility function in (2) is motivated by the current standardization of video streaming in wireless systems [10]. It takes into account two important metrics for measuring video streaming quality, namely, the achievable bitrate of the video and the bitrate switch during video streaming.

After receiving the video tile requests, the base station needs to determine (a) the data of which tiles should be included in the common message, and (b) what is the proportion of the data of each tile in the common message. After construction, the common message is encoded into a data stream $s_0(t) \in \mathbb{C}$, where $\mathbb{E}\left[|s_0(t)|^2\right] = 1$. The beamforming vector for the common message is denoted as $\boldsymbol{b}_0(t) \in \mathbb{C}^{N_t}$. A private message $s_n(t) \in \mathbb{C}$ is constructed for user $n \in \mathcal{N}$ to encode the private part of the data requested by user n in time slot t. We have $\mathbb{E}\left[|s_n(t)|^2\right] = 1$. The beamforming vector for the private message of user n is denoted as $\boldsymbol{b}_n(t) \in \mathbb{C}^{N_t}$. We define $\boldsymbol{b}(t) = (\boldsymbol{b}_0(t), \boldsymbol{b}_1(t), \ldots, \boldsymbol{b}_N(t)) \in \mathbb{C}^{(N+1)N_t}$. We denote the maximum transmit power of the base station as P_{max}. We have the following constraint:

$$\text{C2:} \quad ||\boldsymbol{b}(t)||_2^2 \leq P_{\text{max}}. \tag{3}$$

We define $\boldsymbol{h}_n^H(t) = \boldsymbol{h}_{n,D}^H(t) + \boldsymbol{h}_{n,R}^H(t)\,\boldsymbol{\Psi}(t)\,\boldsymbol{G}(t)$. The signal received by user $n \in \mathcal{N}$ in time slot $t \in \mathcal{T}$ is given by:

$$y_n(t) = \boldsymbol{h}_n^H(t)\,\boldsymbol{b}_0(t) s_0(t) + \sum_{m \in \mathcal{N}} \boldsymbol{h}_n^H(t)\,\boldsymbol{b}_m(t) s_m(t) + z_n(t),$$

where $z_n(t)$ is the additive white Gaussian noise (AWGN) with zero mean and variance σ^2 at user n in time slot t. User n first decodes the common message by treating the private messages of all users as interference. The SINR of the common message at user n in time slot t is given by:

$$\gamma_n^{\text{c}}(t) = \frac{\left|\boldsymbol{h}_n^H(t)\boldsymbol{b}_0(t)\right|^2}{\sum_{m \in \mathcal{N}} \left|\boldsymbol{h}_n^H(t)\boldsymbol{b}_m(t)\right|^2 + \sigma^2}, \; n \in \mathcal{N}. \tag{4}$$

The achievable rate for the common message at user n in time slot t is $R_n^{\text{c}}(t) = \log_2(1 + \gamma_n^{\text{c}}(t))$. Let $R^{\text{c}}(t)$ denote the transmission rate of the common message in time slot t. To ensure successful decoding of the common message at all users, we have the following constraint on $R^{\text{c}}(t)$:

$$\text{C3:} \quad R^{\text{c}}(t) = \min\{R_1^{\text{c}}(t), \ldots, R_N^{\text{c}}(t)\}. \tag{5}$$

We denote the proportion of $R^{\text{c}}(t)$ that is dedicated to the data transmission of video tile $i \in \mathcal{I}(t)$ in time slot t as $c_i(t)$. We have

$$\text{C4:} \quad \sum\nolimits_{i \in \mathcal{I}(t)} c_i(t) \leq 1, \tag{6}$$

$$\text{C5:} \quad c_i(t) \geq 0, \; i \in \mathcal{I}(t). \tag{7}$$

After successful decoding of the common message, user n removes the corresponding signal from $y_n(t)$ using SIC, and decodes its private message. The SINR of the private message at user $n \in \mathcal{N}$ in time slot $t \in \mathcal{T}$ is given by [2]:

$$\gamma_n^{\text{p}}(t) = \frac{\left|\boldsymbol{h}_n^H(t)\boldsymbol{b}_n(t)\right|^2}{\sum_{m \in \mathcal{N} \setminus \{n\}} \left|\boldsymbol{h}_n^H(t)\boldsymbol{b}_m(t)\right|^2 + \sigma^2}, \; n \in \mathcal{N}. \tag{8}$$

The achievable rate of the private message of user n is given by $R_n^{\text{p}}(t) = \log_2(1+\gamma_n^{\text{p}}(t))$. Let $p_{n,i}(t)$ denote the proportion of $R_n^{\text{p}}(t)$ that is used to transmit the data of tile $i \in \mathcal{I}_n(t)$ in time slot t. We have

$$\text{C6:} \quad \sum\nolimits_{i \in \mathcal{I}_n(t)} p_{n,i}(t) \leq 1, \; n \in \mathcal{N}, \tag{9}$$

$$\text{C7:} \quad p_{n,i}(t) \geq 0, \; i \in \mathcal{I}_n(t), \; n \in \mathcal{N}. \tag{10}$$

Let $J_{n,i}^{\text{c}}(t)$ denote the number of bits that user n obtains from the common message for tile $i \in \mathcal{I}_n(t)$ in time slot t. We have $J_{n,i}^{\text{c}}(t) = W T_{\text{DL}} c_i(t) R^{\text{c}}(t)$, $i \in \mathcal{I}_n(t)$, $n \in \mathcal{N}$, where W and T_{DL} are the downlink transmission bandwidth and time duration, respectively. Let $J_{n,i}^{\text{p}}(t)$ denote the number of bits that user n obtains from its private message for tile $i \in \mathcal{I}_n(t)$ in time slot t. We have $J_{n,i}^{\text{p}}(t) = W T_{\text{DL}} p_{n,i}(t) R_n^{\text{p}}(t)$, $i \in \mathcal{I}_n(t)$, $n \in \mathcal{N}$. The total number of bits required by user n to retrieve tile i with bitrate $v_{n,i}(t)$ is given by $J_{n,i}^{\min}(t) = T_v\, v_{n,i}(t)$, $i \in \mathcal{I}_n(t)$, $n \in \mathcal{N}$, where T_v denotes the time duration of a video tile. In order to ensure that all data requested by user n can be received within T_{DL}, we have the following per-user per-tile QoS constraint:

$$\text{C8:} \quad J_{n,i}^{\text{c}}(t) + J_{n,i}^{\text{p}}(t) \geq J_{n,i}^{\min}(t), \; i \in \mathcal{I}_n(t), \; n \in \mathcal{N}. \tag{11}$$

In time slot t, we tackle the following utility maximization problem for an IRS-aided RS VR system:

$$\begin{aligned} \underset{\substack{\boldsymbol{b}(t),\, \boldsymbol{\Psi}(t),\, \boldsymbol{v}_n(t),\, n \in \mathcal{N}, \\ c_i(t),\, i \in \mathcal{I}(t), \\ p_{n,i}(t),\, i \in \mathcal{I}_n(t),\, n \in \mathcal{N}}}{\text{maximize}} \quad & u(t) \triangleq \sum_{n \in \mathcal{N}} u_n(t) \\ \text{subject to} \quad & \text{constraints C1}-\text{C8}, \\ & \text{C9: } \phi_l(t) \in [0,\, 2\pi), \; l \in \{1, 2, \ldots, L\}. \end{aligned} \tag{12}$$

Problem (12) is a mixed-integer nonconvex optimization problem. We present an AO algorithm for solving problem (12) in the journal version of this paper [11]. Although a suboptimal solution can be obtained with the AO algorithm, the AO algorithm can be computationally expensive for VR streaming applications. In the next section, we propose a DRL-based algorithm to efficiently solve problem (12).

III. The Proposed DRL-based Algorithm

In this section, we present the Markov decision process (MDP) formulation and the proposed DRL-based algorithm.

A. MDP Formulation

We model the sequential decision process for solving problem (12) in time slot $t \in \mathcal{T}$ as an MDP with τ^{max} decision epochs. We drop time index t for notational simplicity. The state in the τ-th decision epoch of the MDP is defined as

$$\begin{aligned} \boldsymbol{s}(\tau) = \big[&\boldsymbol{h}_{n,D},\, \text{vec}(\text{diag}(\boldsymbol{h}_{n,R}^H)\boldsymbol{G}),\, n \in \mathcal{N}, \\ &\boldsymbol{b}(\tau-1),\, \boldsymbol{c}(\tau-1),\, R_n^{\text{c}}(\tau-1),\, R_n^{\text{p}}(\tau-1), \\ &\boldsymbol{p}_n(\tau-1),\, \boldsymbol{v}_n(\tau-1),\, \text{vec}(\boldsymbol{\Psi}(\tau))\,\boldsymbol{o}_n,\, n \in \mathcal{N}\big], \end{aligned} \tag{13}$$

where $\boldsymbol{p}_n(\tau) = (p_{n,i}(\tau),\, i \in \mathcal{I})$, $\boldsymbol{c}(\tau) = (c_i(\tau),\, i \in \mathcal{I})$, and $\boldsymbol{o}_n = \left(\mathbb{1}\left(i \in \mathcal{I}_n\right), i \in \mathcal{I}\right) \in \{0,1\}^{I_{\text{max}}}$.

The action vector in the τ-th decision epoch is defined as

$$\boldsymbol{a}(\tau) = \left(\boldsymbol{b}(\tau),\, \text{vec}(\boldsymbol{\Psi}(\tau)), \boldsymbol{c}(\tau), \boldsymbol{p}_n(\tau), \boldsymbol{v}_n(\tau), n \in \mathcal{N}\right). \tag{14}$$

We use the objective function in problem (12) as the reward function $r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$. That is, $r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) = u(\tau)$.

B. Algorithm Design

We use an actor network with learnable parameters $\boldsymbol{\Phi}_{\text{act}}$ to learn a policy for solving problem (12). The policy learned by the actor network, i.e., $\pi_{\boldsymbol{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau))$, defines a mapping from a state to an action. The critic network learns a state-action value function $Q_{\boldsymbol{\Phi}_{\text{crt}}}$, which is parameterized by $\boldsymbol{\Phi}_{\text{crt}}$.

The Q-value $Q_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ estimates the discounted total reward of selecting action $\boldsymbol{a}(\tau)$ in state $\boldsymbol{s}(\tau)$. The goal of the actor-critic method is to learn a policy that maximizes the discounted total reward [12]. We have

$$\begin{aligned}&\underset{\mathbf{\Phi}_{\text{act}}}{\text{maximize}}\ \mathcal{L}(\mathbf{\Phi}_{\text{act}})\\ &\triangleq \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\mathbf{\Phi}_{\text{act}}}}, \boldsymbol{a}\sim\pi_{\mathbf{\Phi}_{\text{act}}}}\left[\sum\nolimits_{\tau'=1}^{\tau^{\max}} \gamma^{\tau'-1} r(\boldsymbol{s}(\tau'), \boldsymbol{a}(\tau'))\right],\end{aligned} \tag{15}$$

where $p_{\pi_{\mathbf{\Phi}_{\text{act}}}}$ denotes the distribution of the state transition as the result of taking actions based on policy $\pi_{\mathbf{\Phi}_{\text{act}}}$, and $\gamma \in [0,1]$ is the discount factor. The deterministic policy gradient for solving problem (15) is given by [12]:

$$\nabla\mathcal{L}_{\text{DPG}} = \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\mathbf{\Phi}_{\text{act}}}}}\left[\nabla Q_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s},\boldsymbol{a})\,|_{\,\boldsymbol{a}=\pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s})} \nabla \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s})\right]. \tag{16}$$

The critic network is updated to minimizing the error of Q-value approximation using the following gradient [13, Ch. 6]:

$$\begin{aligned}\nabla\mathcal{L}(\mathbf{\Phi}_{\text{crt}}) = \mathbb{E}_{\boldsymbol{s}\sim p_{\pi_{\mathbf{\Phi}_{\text{act}}}}, \boldsymbol{a}\sim\pi_{\mathbf{\Phi}_{\text{act}}}}\Big[&2\nabla Q_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}))\\ &\left(Q_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s})) - \widehat{Q}_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}))\right)\Big],\end{aligned} \tag{17}$$

where $\widehat{Q}_{\mathbf{\Phi}_{\text{crt}}}(\boldsymbol{s}, \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}))$ is the target of the Q-value approximation. The learning agent maintains an experience replay that stores the system transition history due to past decisions as a *system transition tuple* $(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), r(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1))$.

Apart from the actor-critic method, we use imitation learning [14] to facilitate the convergence and improve the quality of the learned policy by learning from the AO algorithm in [11]. To this end, we first invoke the AO algorithm to solve problem (12) for D time slots. Note that one time slot corresponds to one episode comprising $\tau^{\max}$ decision epochs in the MDP. In the τ-th decision epoch of the d-th time slot, where $d \in \{1, \ldots, D\}$, we first initialize the AO algorithm with the variables in $\boldsymbol{s}^{(d)}(\tau)$, which is the state vector in the τ-th decision epoch of the d-th time slot. We then execute the AO algorithm for one iteration and obtain the solution. We denote this solution as $\boldsymbol{a}_{\text{AO}}^{(d)}(\tau)$. With $\boldsymbol{a}_{\text{AO}}^{(d)}(\tau)$, we determine the reward and the next state of the MDP as $r(\boldsymbol{s}^{(d)}(\tau), \boldsymbol{a}_{\text{AO}}^{(d)}(\tau))$ and $\boldsymbol{s}^{(d)}(\tau+1)$, respectively. Then, the system transition obtained from the execution of the AO algorithm in the τ-th decision epoch of the d-th time slot is denoted as the system transition tuple $(\boldsymbol{s}^{(d)}(\tau), \boldsymbol{a}_{\text{AO}}^{(d)}(\tau), r(\boldsymbol{s}^{(d)}(\tau), \boldsymbol{a}_{\text{AO}}^{(d)}(\tau)), \boldsymbol{s}^{(d)}(\tau+1))$. We index this system transition tuple with the tuple (d, τ). We use a *demonstration replay* to store the system transition tuples obtained from the execution of the AO algorithm.

For imitation learning, we first sample a minibatch of M_D transition tuples from the demonstration replay. The indices of these tuples are collected in set $\mathcal{M}_D$. For each state $\boldsymbol{s}^{(d)}(\tau), (d,\tau) \in \mathcal{M}_D$, an *imitation loss* $\mathcal{L}_{\text{IMI}}$ is determined based on the mean squared error between the action chosen by the actor network, i.e., $\pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{(d)}(\tau))$, and the solution obtained by the AO algorithm, i.e., $\boldsymbol{a}_{\text{AO}}^{(d)}(\tau)$. We have

$$\mathcal{L}_{\text{IMI}} = \frac{1}{M_D}\sum\nolimits_{(d,\tau)\in\mathcal{M}_D} ||\pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}^{(d)}(\tau)) - \boldsymbol{a}_{\text{AO}}^{(d)}(\tau)||^2. \tag{18}$$

Algorithm 1 Training Algorithm

```
Set episode counter $t \leftarrow 0$.
Initialize $\mathbf{\Phi}_{\text{act}}$ and $\mathbf{\Phi}_{\text{crt}}$.
Execute the AO algorithm for $D$ episodes and store the system transition tuples in the demonstration replay.
Perform random exploration and store the system transition tuples in the experience replay.
while $t \leq T_{\max}$ do
    Set $c_i(0) = \mathbb{1}(i \in \mathcal{I}(t))\frac{1}{|\mathcal{I}(t)|}$, $p_{n,i}(0) = \mathbb{1}(i \in \mathcal{I}_n(t))\frac{1}{I_n(t)}$, and $v_{n,i}(0) = \mathbb{1}(i \in \mathcal{I}_n(t))\, v_1$, $i \in \mathcal{I}, n \in \mathcal{N}$.
    Initialize $\mathbf{\Psi}(0)$ and $\boldsymbol{b}(0)$ based on random initialization.
    Initialize $\tau \leftarrow 1$.
    while $\tau \leq \tau^{\max}$ do
        Determine the action $\boldsymbol{a}(\tau) \leftarrow \pi_{\mathbf{\Phi}_{\text{act}}}(\boldsymbol{s}(\tau)) + \boldsymbol{\varrho}_{\text{epl}}$.
        Store the transition tuple in the experience replay.
        Determine the gradients in (17) and (19).
        Update $\mathbf{\Phi}_{\text{act}}$ and $\mathbf{\Phi}_{\text{crt}}$ using the Adam optimizer.
        $\tau \leftarrow \tau + 1$.
    end while
    $t \leftarrow t + 1$.
end while
```

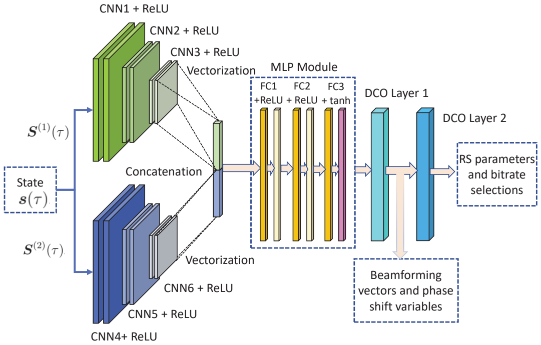

Fig. 2. The DNN architecture of the actor network. The actor network takes state vector $\boldsymbol{s}(\tau)$ as input and determines action $\boldsymbol{a}(\tau)$.

By combining with the deterministic policy gradient, the gradient for updating the actor network is given by:

$$\nabla\mathcal{L}_{\text{act}} = \omega_1\nabla\mathcal{L}_{\text{DPG}} + \omega_2\nabla\mathcal{L}_{\text{IMI}}, \tag{19}$$

where ω_1 and ω_2 are positive parameters representing the weights of the deterministic policy gradient and the gradient of imitation loss, respectively.

The proposed training algorithm is illustrated in **Algorithm 1**. We train the learning agent for $T_{\max}$ episodes. In each training iteration, we sample one minibatch from the demonstration replay and one from the experience replay. We determine the losses based on (17) and (19), and update the learnable parameters. We use the Adam optimizer [15] with a learning rate of α to update the learnable parameters of the actor and critic networks. We add exploration noise $\boldsymbol{\varrho}_{\text{epl}}$ during the training phase to facilitate exploration.

The proposed DNN structure of the actor network is shown in Fig. 2. We construct two matrices, namely $\boldsymbol{S}^{(1)}(\tau)$ and $\boldsymbol{S}^{(2)}(\tau)$, from $\boldsymbol{s}(\tau)$. $\boldsymbol{S}^{(1)}(\tau)$ collects the information about the channel, beamforming vectors, and IRS phase shifts. $\boldsymbol{S}^{(2)}(\tau)$ collects the information about the video tile requests, RS parameters, and bitrate selections. Then, $\boldsymbol{S}^{(1)}(\tau)$ is passed through three convolutional neural network (CNN) layers.

Each CNN layer is followed by a rectified linear unit (ReLU) activation layer. We employ another three CNN layers to process $\boldsymbol{S}^{(2)}(\tau)$. The outputs of the CNN layers are vectorized and concatenated to obtain a new vector. We feed this vector into a multilayer perceptron (MLP) module with three fully-connected (FC) layers to obtain the beamforming variables $\boldsymbol{b}'(\tau)$ and IRS phase shifts $\boldsymbol{\psi}'(\tau)$. We define $\boldsymbol{a}'(\tau) = (\boldsymbol{b}'(\tau), \boldsymbol{\psi}'(\tau))$.

Then, we determine the projection of $\boldsymbol{a}'(\tau)$ onto the feasible set of problem (12) by solving the following problem:

$$\begin{aligned} \underset{\boldsymbol{a}(\tau)}{\text{minimize}} \quad & ||\boldsymbol{a}(\tau) - \boldsymbol{a}'(\tau)||^2 \\ \text{subject to} \quad & \text{constraints C2, C8, C9.} \end{aligned} \tag{20}$$

We solve problem (20) using a DCO layer [8] since it can be transformed into a convex problem by applying a quadratic transform [11]. Compared with conventional convex solvers, the DCO layer can be integrated as a layer in the DNN module. In addition, it can solve problem (20) efficiently in a batch-wise manner, which significantly facilitates the training process. We denote the feasible beamforming and IRS phase shift solutions obtained by solving problem (20) as $\boldsymbol{b}(\tau)$ and $\boldsymbol{\psi}(\tau)$, respectively. We then feed $\boldsymbol{b}(\tau)$ and $\boldsymbol{\psi}(\tau)$ into a second DCO layer which solves the following optimization problem to obtain the RS parameters and bitrate selections:

$$\begin{aligned} \underset{\substack{\boldsymbol{v}_n,\, n\in\mathcal{N},\, c_i,\, i\in\mathcal{I}, \\ p_{n,i},\, i\in\mathcal{I}_n,\, n\in\mathcal{N}}}{\text{maximize}} \quad & \sum_{n\in\mathcal{N}} \sum_{i\in\mathcal{I}_n} v_{n,i} \\ \text{subject to} \quad & \text{constraints C1, C3−C8.} \end{aligned} \tag{21}$$

We relax constraint C1 as $v_1 \leq v_{n,i} \leq v_M$, $i \in \mathcal{I}_n$, $n \in \mathcal{N}$. The relaxed problem can be solved using the DCO layer. We round down the solution of $v_{n,i}, i \in \mathcal{I}_n$, $n \in \mathcal{N}$, to the nearest feasible solution. Note that rounding down the bitrate solutions only makes the right-hand side of constraint C8 smaller, while the left-hand side remains unchanged. Therefore, the obtained solutions satisfy constraint C8.

The critic network has a similar structure as the actor network with the following two modifications: (a) the DCO layers are not present in the critic network, and (b) the tanh activation layer is replaced by the ReLU activation layer.

IV. Performance Evaluation

We consider a 10 m × 10 m × 3.5 m indoor facility for VR streaming as illustrated in Fig. 1. Each user is designated a 2.7 m × 2.7 m area. We assume all channels are line-of-sight. We assume a carrier frequency of 60 GHz. Let $d_{n,D}$, $d_{n,R}$, and d_0 denote the distance between the base station and user n, the distance between the IRS and user n, and the distance between the base station and the IRS, respectively. We determine the CSI of the direct and reflected channels by $\boldsymbol{h}_{n,D} = (\frac{\nu}{4\pi d_{n,D}})^{\zeta} \widehat{\boldsymbol{h}}_{n,D}$, $\boldsymbol{h}_{n,R} = (\frac{\nu}{4\pi d_{n,R}})^{\zeta} \widehat{\boldsymbol{h}}_{n,R}$, and $\boldsymbol{G} = (\frac{\nu}{4\pi d_0})^{\zeta} \widehat{\boldsymbol{G}}$, where ν is the wavelength of the carrier signal and ζ is the path loss exponent. The elements in $\widehat{\boldsymbol{h}}_{n,D}$, $\widehat{\boldsymbol{h}}_{n,R}$, and $\widehat{\boldsymbol{G}}$ are complex Gaussian distributed with zero mean and unit variance. We use the real-world dataset from [9] to generate the video tile requests in our simulation. The available bitrate selections are given by set $\mathcal{V} = \{2, \ldots, 11\}$ Mbps. The other simulation parameter settings are given in Table I. We compare the performance of the following baseline systems and algorithms:

TABLE I
Simulation Parameters for Performance Evaluation

Parameter	Value
Bandwidth for downlink W	1 GHz
Path loss exponent ζ	2.29
Maximum transmit power $P^{\max}$	1 Watt
Noise power	-174 dBm/Hz
Time duration of each video tile T_v	1 sec
Downlink transmission window T_{DL}	10 ms
Coefficient κ^{intra}	10
Number of decision epochs per episode $\tau^{\max}$	50
Learning rate α	5×10^{-4}
Minibatch size M_D	512
Value of q for q-step return	5
Kernel size of the CNN layers	2×2
Number of channels of the CNN layers	16, 16, 32, 16, 16, 32
Coefficients for training loss ω_1, ω_2	10^{-3} , 1
Discount factor γ	0.95

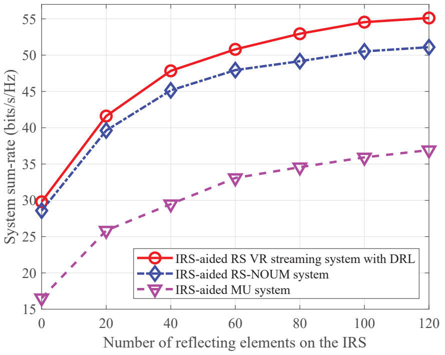

Fig. 3. System sum-rate versus the number of reflecting elements L. We set $N_t = N = 6$. Note that $L = 0$ represents the system without an IRS.

- **IRS-aided RS-NOUM system [4]**: We extend the RS-NOUM system proposed in [4] by including an IRS. We solve the sum-rate maximization problem for the resulting system with the constraints of problem (12) using an AO algorithm.
- **IRS-aided multiuser system without RS (IRS-aided MU system) [6]**: In this system, the video tiles are sent to the users via unicast without RS. We solve the sum-rate maximization problem for the resulting system with the constraints of problem (12) using an AO algorithm.

In Fig. 3, we vary the number of reflecting elements L and investigate the system sum-rate. When L is equal to 120, the IRS-aided RS VR streaming system with the proposed DRL-based algorithm achieves a system sum-rate that is 6.8% and 49.3% higher than that of the IRS-aided RS-NOUM system and IRS-aided MU system, respectively. In addition, $L = 0$ implies a system without IRS. For the IRS-aided RS VR streaming system with DRL-based algorithm, deploying an IRS with $L = 120$ reflecting elements results in a system sum-rate improvement of 84.9% compared to the same system without IRS. This is due to the SINR improvement achieved

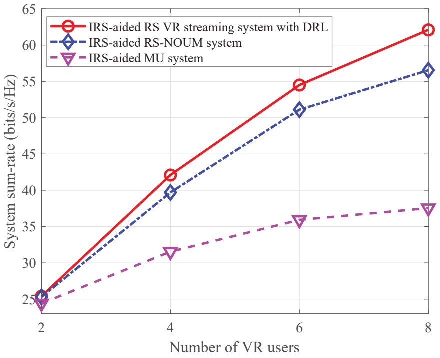

Fig. 4. System sum-rate versus the number of VR users N. We set $N_t = 6$ and $L = 100$.

TABLE II
AVERAGE EXECUTION RUNTIME OF DIFFERENT ALGORITHMS

Parameter Settings	$N_t = 6$, $N = 4$, $L = 60$	$N_t = 6$, $N = 6$, $L = 60$	$N_t = 6$, $N = 6$, $L = 100$
IRS-aided RS VR streaming system with DRL	7.61 sec	8.42 sec	10.17 sec
IRS-aided RS-NOUM system	3.87 min	4.83 min	14.65 min
IRS-aided MU system	3.06 min	3.30 min	10.37 min

by the additional propagation channels created by the IRS. Furthermore, since the common rate is determined by the user experiencing the minimum SINR (as shown in (5)), the additional DoF introduced by the IRS are implicitly exploited to increase the rate of the common message.

In Fig. 4, we show the system sum-rate versus the number of users N. We observe that the performance gains of the IRS-aided RS VR streaming system over the IRS-aided RS-NOUM and IRS-aided MU systems become more pronounced for more users. For more users, a particular tile is more likely to be requested by multiple users, and therefore there are more shared tile requests of the users to be exploited by the IRS-aided RS VR streaming system. When N is equal to 8, the IRS-aided RS VR streaming system with the proposed DRL-based algorithm achieves a system sum-rate that is 26.7% and 90.8% higher than that of the IRS-aided RS-NOUM system and IRS-aided MU system, respectively.

In Table II, we compare the average online execution runtime of different algorithms per time slot. We observe that the average runtime of the proposed DRL-based algorithm is lower than that of the considered AO algorithms. This is because the computationally expensive processes required for solving the beamforming and IRS phase shift subproblems using AO are not needed in the proposed DRL-based algorithm. In particular, when $N_t = 6$, $N = 6$, and $L = 100$, the average runtime of the proposed DRL-based algorithm is only 1.16% and 1.63% of the average runtime of the IRS-aided RS-NOUM system and IRS-aided MU system, respectively.

V. CONCLUSION

In this paper, we proposed a novel IRS-aided RS VR streaming system, in which RS and IRS were exploited to improve the QoS of 360-degree video streaming. We jointly optimized the DoF of the system using the proposed DRL-based algorithm. We combined the DRL technique with imitation learning to improve the quality of the learned policy. Simulation results based on a real-world dataset showed that the DoF introduced by RS and IRS can be efficiently exploited by the proposed DRL-based algorithm to achieve a higher system sum-rate than the benchmark IRS-aided RS-NOUM and IRS-aided MU systems. The performance improvement of the proposed IRS-aided RS VR system becomes more pronounced as the number of shared video tile requests increases. Our simulation results also revealed the respective contribution of RS and IRS to the performance gain achieved with the proposed IRS-aided RS VR system. For future work, we will tackle the potential impact of the suboptimality of the AO algorithm on policy learning [11].

REFERENCES

[1] H. Joudeh and B. Clerckx, "Sum-rate maximization for linearly precoded downlink multiuser MISO systems with partial CSIT: A rate-splitting approach," *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4847–4861, Nov. 2016.

[2] Y. Mao, E. Piovano, and B. Clerckx, "Rate-splitting multiple access for overloaded cellular Internet of things," *IEEE Trans. Commun.*, vol. 69, no. 7, pp. 4504–4519, Jul. 2021.

[3] A. Bansal, K. Singh, B. Clerckx, C.-P. Li, and M.-S. Alouini, "Rate-splitting multiple access for intelligent reflecting surface aided multi-user communications," *IEEE Trans. Veh. Techno.*, vol. 70, no. 9, pp. 9217–9229, Sept. 2021.

[4] Y. Mao, B. Clerckx, and V. O. K. Li, "Rate-splitting for multi-antenna non-orthogonal unicast and multicast transmission: Spectral and energy efficiency analysis," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8754–8770, Dec. 2019.

[5] H. Joudeh and B. Clerckx, "Rate-splitting for max-min fair multigroup multicast beamforming in overloaded systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7276–7289, Nov. 2017.

[6] Q. Wu and R. Zhang, "Intelligent reflecting surface enhanced wireless network via joint active and passive beamforming," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5394–5409, Nov. 2019.

[7] H. Fu, S. Feng, and D. W. K. Ng, "Resource allocation design for IRS-aided downlink MU-MISO RSMA systems," in *Proc. of IEEE Int'l Conf. Commun. (ICC) Workshop*, Jun. 2021.

[8] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Proc. of Conf. Neural Inf. Proc. Syst. (NeurIPS)*, Vancouver, Canada, Dec. 2019.

[9] S. Knorr, C. Ozcinar, C. O. Fearghail, and A. Smolic, "Director's cut: A combined dataset for visual attention analysis in cinematic VR content," in *Proc. of ACM SIGGRAPH European Conf. Visual Media Production*, London, United Kingdom, Dec. 2018.

[10] 3GPP TS 26.247 V17.1.0, "Technical specification group services and system aspects; Transparent end-to-end packet-switched streaming service (PSS); Progressive download and dynamic adaptive streaming over HTTP (3GP-DASH) (Release 17)," Jun. 2022.

[11] R. Huang, V. W.S. Wong, and R. Schober, "Rate-splitting for intelligent reflecting surface-aided multiuser VR streaming," accepted for publication in *IEEE J. Sel. Area Commun.*, Dec. 2022.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. of Int'l Conf. Learning Representations, (ICLR)*, San Juan, Puerto Rico, May 2016.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[14] P. Rashidinejad, B. Zhu, C. Ma, J. Jiao, and S. Russell, "Bridging offline reinforcement learning and imitation learning: A tale of pessimism," in *Proc. of Conf. Neural Inf. Proc. Syst. (NeurIPS)*, Dec. 2021.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of Int'l Conf. Learning Representations, (ICLR)*, San Diego, CA, May 2015.