Improved video QoE in wireless networks using deep reinforcement learning

Improved Video QoE in Wireless Networks using Deep Reinforcement Learning

Henrique D. Moura*, Júnia Maísa Oliveira†, Daniel Soares†, Daniel F. Macedo†, Marcos A. M. Vieira†,
*IDLab - imec, University of Antwerp, Belgium
†Universidade Federal de Minas Gerais - Computer Science Department, Brazil

***Abstract*—Millions of videos are watched per minute on the Internet. Due to real-time performance demands, such as high-quality video streaming, network administrators face new challenges to control the network and cope with the expected quality of experience (QoE). Automatic control is a necessity to reduce the OPEX, because it could reduce the need for resource overprovisioning, as well as the number of human administrators. Dynamic rate in video streaming alleviates the resource usage, but it worsens the video quality when a network bottleneck occurs, lowering the QoE. This paper dynamically adjusts the IEEE 802.11 parameters to improve the network condition and hence maintain a higher QoE. While traditional networks are not aware of the application, in our proposal the controller learns the configuration of the access points (APs) (in terms of transmission power and channel number) that provide the best QoE, using double deep Q-learning (DDQL). The proposal improves video QoE by 91% in the best case, when compared to three baselines. It also balances the QoE among clients, improving the fairness up to 115% when compared to the baselines.**

***Index Terms*—Deep Reinforcement Learning, Adaptive Control, Wireless Networks, Quality of Experience**

I. Introduction

A recent study shows that an adult in the UK spends an average of 3 hours 52 minutes a day watching videos online (June 2020) [1], before the pandemic it was 30 minutes less. Real-time and resource demanding applications, such as high-quality video streaming, brought new challenges to network control, as users expect high levels of quality to be upheld. However, improving user satisfaction and minimizing customer turnover while still maintaining a competitive advantage can be a significant challenge for service providers. Quality of Experience (QoE) is hard to measure because it is perceived subjectively by users, as a result of the user's feelings and personality (e.g., predispositions, expectations, motivation, mood), the characteristics of the application (e.g., complexity, purpose, usability, functionality), and the context within which the application is experienced.

Wireless Local Area Networks (WLAN) have become commonplace in office and campus sites, and the number of public Wi-Fi hotspots is growing significantly year over year (e.g. from 169 million in 2018 to 628 million by 2023 [2] and much effort exists to estimate QoE using known parameters obtained from the network devices and applications [3].

Automatic Wireless local area network (WLAN) management is quite challenging. First, the wireless medium is subject to performance problems, and the communication deteriorates due to the dynamics of the medium. Second, the radio spectrum is often crowded, which requires frequent interventions to maintain the communication quality. Third, WLAN are more prone to packet loss, delay and low connection speeds than wired networks. State-of-the-art (SOTA) IEEE 802.11 networks have yet to exhibit dynamic QoE. This is due to the fact that devices still implement QoE statically, while centrally controlled APs remain unaware of the specific application. Moreover, these platforms are closed source, leaving clients dependent on vendors for new features [4]. Thus, automatic management approaches in WLAN should be studied for the following reasons: (i) access providers offload WLAN management to the end-users for legal reasons or because of technical difficulties due to the large number of users; (ii) WLAN users are non-technical, hence they cannot undertake complex configurations by themselves; (iii) problems in the network take a long time to be identified, as users only notice that there is a problem when their QoE degrades significantly or applications freeze [5]; (iv) manual operation of networking services is costly. For example, automation can provide network operation gains up to 2% and 13%, for ISPs and mobile network operators, respectively [6].

Numerous studies have examined QoE on wireless networks, with a subset of these studies focusing on IEEE 802.11 networks. Some of these studies investigate client parameters [7], [8], while others explore video server parameters [9]. For instance, dynamic rate in video streaming, such as dynamic adaptive streaming over HTTP dynamic adaptive streaming over HTTP (DASH)), enables the adaptation of video quality to the instantaneous link quality. Further optimizations are possible if the WLAN automatically optimizes the media access control and physical layers. However, the studies do not improve the Access Point (AP) channel control algorithms.

We propose a WLAN manager with a reinforcement learning control loop that optimizes the network and, consequently, improves video QoE. Our proposal employs a Software-defined Networking (SDN) architecture. We evaluate our proposal with a prototype over a realistic testbed, where the control loop automatically assigns the wireless channel and the transmission power to each AP. Results show that the control loop improves video QoE by up to 91% and the regret by up to 84% when compared to the baselines. It also improves the fairness index up to 115% and increases the mean opinion score (MOS) when compared to the baselines. Our main contributions are an automatic control loop, based

on double deep Q-learning, that improves QoE via dynamic channel assignment and transmission power control, and an experimental analysis of the proposed control loop.

The remainder of this paper is organized as follows. Section II shows the proposed control loop. Section III describes the scenario used to test our prototype. The QoE predictor is evaluated in Section IV. Section V discusses the results of our prototype. Related work is discussed in Section VI. Finally, Section VII concludes and discusses future work.

II. Network Architecture of the QoE-aware control loop

This section introduces the network architecture of the QoE-aware control loop proposed in this work. In this paper, the Deep RL control loop optimises the QoE of DASH video flows by automatically adjusting the transmission power and the channel of Wi-Fi APs. The controller runs the Deep RL control loop, and the control loop can manage several Wi-Fi APs in a network, as shown in Figure 1. The controller interacts with the APs, requesting information to serve as input for the Deep RL algorithm, and the output of the algorithm defines changes on the Wi-Fi parameters of the APs. This interaction can employ SDN protocols, or protocols such as REST. The presented network architecture is flexible, being able to treat other use cases, such as in previous works that controlled the QoE of web flows (employing traditional reinforcement learning) using Ethanol as SDN substrate [10].

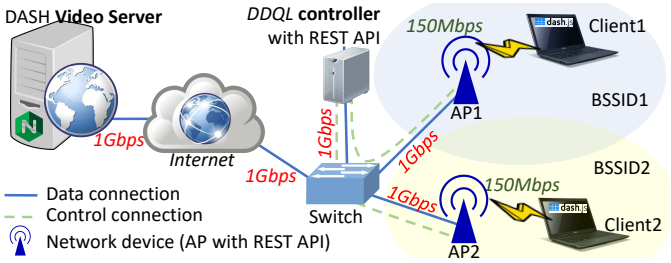

Figure 1. Network architecture of the control system.

The components of the Deep Reinforcement learning controller for Video QoE are shown in Figure 2. The control relies on a southbound SDN interface to interact with the APs, which in this paper is a simple REST API. The Deep RL controller has four components: a flow identification component, the Reinforcement Learning Control Loop, a MOS Predictor, and the Reward Function.

The control loop cycle operates as follows. First, flow identification (§II-A) helps the controller identify the video flows. Next, the controller collects data from the APs to compose the system states and determine the suitable actions (§II-B). The controller also collects information to calculate the mean opinion score (MOS – §II-C) of the video flows. Finally, the Deep reinforcement learning (RL) control loop decides the next action, i.e., the AP configuration that maximize the video flows QoE (§II-D). This configuration is transmitted using the southbound SDN interface. Finally, the cycle restarts. Below we present details about the flow identification, Deep Reinforcement Learning Control Loop, and reward function.

A. Flow identification

The flow classification module identifies that a certain flow is a video session. Our control loop relies on the state of the art for flow classification, since this is already a well-studied topic, and many effective classifiers already exist in the literature (e.g. [11], [12]).

Thus, for simplicity reasons, the implementation described in Section V identifies whether the flow is directed to our video server or not, *i.e.*, it is based on the server address and port. Also, we assume that all video traffic is DASH video. We chose DASH over other streaming video protocols because it is used in the most popular video streaming systems today (e.g. YouTube, VIMEO, and Netflix).

B. Deep Reinforcement Learning Control Loop

The control loop uses a double deep Q-learning (DDQL) model to select the best transmission power and the wireless channel for each controlled AP. The components of the Deep RL agent are explained below.

Deep RL algorithm: The agent employs DDQL, this is an actor-based algorithm, and in future work we plan to explore actor-critic approaches in order to improve the convergence time. The neural networks were implemented using temporal convolutional neural network (TCN), in order to reduce its computation time. Further, our agent uses an ϵ–greedy [13] exploration strategy.

In very broad terms, the agent decides which action A maximizes the reward when it is on a certain state S. The following paragraphs detail how our agent models states and actions, while §II-D details the reward function.

States: The state is represented by a tuple (s_{t-1}, s_t), that contains the features of the current and previous timesteps. We used two periods because the MOS predictor considers two periods (see Section IV). The number of periods can be increased provided that the predictor employs the same number of periods. The length of the period is a fixed number of seconds (configurable by the administrator) that represents the time between two consecutive actions.

For the system to learn a Q-function that can be generalized, the state must be independent of which AP is providing information to the learner. We chose information related to the state of the wireless medium as well as the usage of the Wi-Fi network, as listed below. We show in brackets if the feature is continuous *[C]* or discrete *[D]*.

- $\#stations$ *[D]*: Represents the number of wireless clients connected to the AP at a given moment;
- $ch1, ch2, \ldots, ch11$ *[D]*: Identifies the transmission channel;
- tx_power *[D]*: The current transmission power of the AP. It measures the configured power in dBm;
- $\#num\_neighbors$ *[D]*: It indicates how many controlled APs are in the vicinity of the current AP;

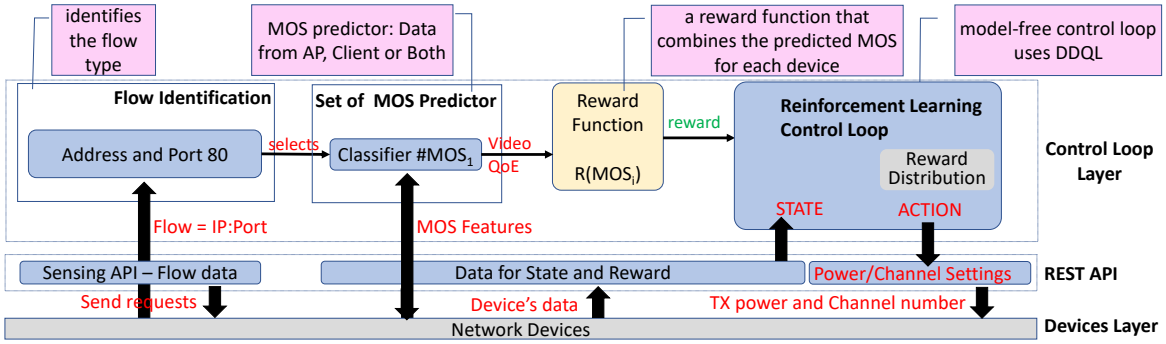

Figure 2. Architecture of the Deep RL controller.

- *ch_noise_max [C]*: indicates the maximum noise detected in the current channel by the AP;
- *perc_phy_busy_time [C]*: measures the occupation of the channel as the percentage of PHY busy time;
- Some features are related to the clients of the AP:
  - *sta_signal_min [C]*: is the minimum signal received by the AP from the connected client;
  - *rec_bitrate_min [C]*: is the minimum received bitrate from the clients;
  - *tx_byte_avg [C]*: the number of bytes transmitted by the clients in the period;
  - *rx_byte_avg [C]*: is the number of bytes received from the clients in the period.

Actions: any discrete action is suitable for the control loop in DDQL. Examples are turn on/off the use of RTS/CTS or adjust the contention window parameters. This paper's control loop alters two parameters of the APs:

- tx_power — the transmission power is in the range of 1 dBm up to 15 dBm, in our devices; and
- ch_number — the channel used in the wireless network.

Example – One network with three APs. Suppose we have one controller automatically configuring three APs. Each AP will have its own DDQL agent, with its own states. Hence, the first controller will have the states $(s_{t-1,1}, s_{t,1})$, the second will have the states $(s_{t-1,2}, s_{t,2})$, and so on. Agents periodically calculate tx_power_i, ch_number_i, which is the adequate transmission power and channel number for AP i.

The choice of having one agent per AP, instead of one single agent controlling the whole network, reduces the convergence time of the algorithm. Studies with one single agent have shown similar performance, however with a more complex state space [10].

C. MOS Prediction

The reward of the Deep RL algorithm relies on an estimation of the QoE of each video flow. Our architecture employs an objective metric, calculated as a Mean Opinion Score (MOS), a five-point scale where 1 represents bad quality and 5 represents excellent quality. The MOS estimator eliminates the need to inquire the user about his or her perception, thus being more scalable and easier to implement on a closed control loop. The predictor estimates the MOS experienced by the user based only on QoS metrics.

We have based our QoE metric on a common metric used in video quality evaluations, called Peak signal to noise ratio (PSNR) [14]. PSNR was used because it is a simple and well accepted metric in the literature. PSNR identifies the differences between the frames from both videos, and determines how much the received video has changed in the transmission. Other more complex metrics could be used as well, as long as they output a numeric value, which then could be used to calculate the reward of the control loop.

PSNR compares the original video stored on the server with the video received by the video client. Our estimator, on the other hand, eliminates interactions with the video client. This occurs because we trained a supervised learning model, trained off-line using real samples of video playback over wireless links. This is detailed in Section IV.

D. Reward Function

RL methods use a scalar reward to drive learning. The proposed reward was modeled using two premises: a) the agent should maximize the average MOS perceived by the video clients; and b) clients should experience similar QoE levels, i.e. the overall fairness of the system should be maximized. Fairness is a very broad concept [15]. We consider in our work that a system is fair when all users receive the same minimum allocation, and unused resources can be reassigned to users that demand more than this minimum.

For the reward to achieve the first objective, we first define $\bar{m}_{t,a}$, the mean MOS perceived by all the clients in a certain AP in time t, and $\bar{m}_t$, the mean of all $\bar{m}_{t,a}$. Then, the first objective is achieved when the reward increases with $\bar{m}_t$:

$$r_t = \bar{m}_t \tag{1}$$

Next, we refine the equation above to add fairness into the objective. We adopted the fairness index defined by [16], which is bounded in the closed interval $[0, 1]$, with 1 indicating perfect fairness. The reward is changed into Equation 2:

$$r_t = 1 + (\bar{m}_t - 1) \times F_t^C \tag{2}$$

The term F_t^C is a fairness multiplier, and the constant C adjusts the impact of the fairness index F_t in the reward. The index F_t proposed by [16] is the following:

$$F_t = 1 - \frac{2 \times se(S)}{H - L} \tag{3}$$

where $se(S)$ is the standard deviation of the set S, and H and L are the upper and lower MOS bounds, respectively. F_t will be zero in the most unbalanced situation, and $F_t = 1$ when all MOS values are equal.

Finally, we penalize the clients with a higher than average MOS ($\bar{r}_t$) in order to free network resources, so clients with a lower than average MOS have room to improve. To do so, we adopt a reward r_t^a for each AP a as follows:

$$r_t^a = \begin{cases} \bar{r}_t & if\ \bar{m}_{t,a} \leq \bar{m}_t \\ 1 + (\bar{m}_t - 1) \times F_t^C & otherwise \end{cases} \tag{4}$$

where $\bar{r}_t$ is the average reward among the interfering APs. Only APs that are on the same channel compete for capacity, because they are co-interfering[1]. Hence, F_t is calculated only over the set of co-interfering APs.

III. Experiment setup

This section describes the scenario and testbed. The scenario is used for collection of video for the MOS predictor (Section IV) as well as for the evaluation of the Deep RL control loop (Section V).

The scenario simulates a site with multiple APs managed by a single administrator, such as in a company or campus. The controller manages two APs, which receive commands via a Representational State Transfer (REST) API. Each AP has one wireless client. Data collected from clients relies only on standard IEEE 802.11 messages. The video server, controller, and APs are connected to a gigabit Ethernet network, as shown in Figure 1. The controller uses Ethernet to send and receive control data. The APs use Ethernet to receive control data from the controller, and also to transmit the video flow to the Internet. Each AP provides access to wireless clients. The APs are started on the same channel and with the same power.

The video file is split into several chunks of the same duration. In each experiment, the server streams a video to wireless clients, and the video loops during the whole experiment. The streamed video is Big Buck Bunny[2], which is available in our video server with several resolutions in two-second chunks. Each video chunk is encoded with the bitrates shown in Table I. The video buffers from both client and server were configured to one minute, so each loop starts fresh. The video server and the clients use the DASH protocol.

The video player used is *Firefox* with the Dash.js video player. It is an open source player, and it can log video performance metrics. Dash.js has three adaptive bitrate algorithms [17]: (1) throughput, (2) Buffer Occupancy based Lyapunov Algorithm (BOLA), and (3) a heuristic that switches between BOLA and the throughput strategies. We used the heuristic, since [17] claim that it shows the strengths of both strategies.

[1]APs co-interfere if they are on overlapping channels, and at least one of them detects the other using IEEE 802.11 "Neighbor Report" messages.

[2]https://peach.blender.org/download/

Table I
Experiment's video parameters

Parameter	Value
Codec	avc3.640032; hev1.1.6.L60.90; vp09.00.40.08; av1.experimental
Resolutions	256x144; 320x180; 384x216; 512x277;604x360; 768x432; 1024x576; 1280x720; 1920x1080; 2560x1440; 3840x2160
Frame rate	30 fps
Chunk size	2 s

The *southbound API* was implemented using a REST API, so the controller can access and manipulate textual representations of the agent resources using a uniform and predefined set of stateless operations. The control model proposed in our paper can be used with several *southbound APIs*. In previous work [10], we implemented this three-tier control using software defined networking (SDN) instead of REST in the *southbound API*. Therefore, similar results could be obtained on other platforms (e.g. 5G-emPOWER).

The experiments run on an open RF environment to perform experiments that are closer to real life, at the expense of limited reproducibility of the radio frequency (RF) interference profile. The wireless devices are aligned: the APs are 45 cm apart, the clients are 90 cm apart, and the APs and clients are separated by 3.5 m. The experiments are performed in an environment with more than 72 interfering APs. The average signal of the testbed APs perceived by the clients is -45.5 dB.

The controller is an Intel i5 computer with 16GB of RAM. The APs have an Intel i7 CPU @ 1.80GHz with 8GB of RAM and Atheros AR9485 2.4 GHz IEEE 802.11n and gigabit Ethernet. The clients are notebooks with Atheros ath9k IEEE 802.11n and one gigabit Ethernet NICs. The clients run Firefox version 69.0.2. The video server is a VM with 2GB RAM and an Intel 2.3GHz processor, running nginx with the MPEG-TS Live Module.

IV. MOS predictor

This section presents the data collection, training and evaluations of the MOS predictor.

A. Dataset

The dataset consists of network parameters as features, and PSNR as outputs. PSNR compares the original video stored on the server with the video received by the client. This was calculated using Video Quality Measurement Tool (https://mmspg.epfl.ch/downloads/vqmt/). The dataset was collected over a testbed. Traffic generation between clients and AP uses *Firefox*. Each round takes 12 hours. Data was collected in a 1-second period for 15 days, providing over 1 million samples.

The label of each instance, called MOS_PSNR, converts PSNR into a MOS, which from now on we call MOS_PSNR. MOS_PSNR incorporates video stalls, which lower the MOS.

Table II
MAPPING PSNR TO MOS SCALE

PSNR in dB	**MOS**	**Quality**
> 37	5	Excellent
31 – 77	4	Good
25 – 31	3	Fair
20 – 25	2	Poor
< 20	1	Bad

The reference resolution is the highest resolution in our dataset (4K). Table II, obtained in [18] presents MOS_PSNR.

The MOS_PSNR metric at time t is the weighted mean of the perceived MOS in the interval and the stalled MOS: $MOS\_PSNR_t = \frac{\sum_{j=1}^{m} \mathcal{F}(PSNR_t^{(j)}) \times p_t^{(j)} + \iota_t}{T_t}$, where $T_t = \sum_{j=1}^{m} p_t^{(j)} + \iota_t$ is the total sampling time in the interval t, $p_t^{(j)}$ is the time that the chunk j ran in the video player, ι_t is how much time the video stopped during the interval t, and $\mathcal{F}(PSNR_t)$ is the MOS value for frame *j* using Table II.

B. Model Training

We trained the classifier for the videos with the dataset presented previously. In a production environment, there must be a MOS predictor that generalizes to a wide variety of videos available for streaming (*e.g.* [19], [20]). Previous works show that it is possible to train QoE models using a few videos, and this model is able to generalize to other videos as long as they use the same video parameters and codecs [21]. The MOS predictor provides 100% accuracy in our train and test set. The model return the PSNR of the videos we are streaming, as desired. The frame sent by the server and the frame received by the client do not need to be compared in real time and, therefore, overfitting does not influence the result.

V. CONTROL LOOP EVALUATION

This section describes the performance evaluation of the complete control loop. The experiment aims to show that the proposed control loop improves the QoE of DASH videos.

A. Baselines

There are no QoE-aware techniques in the literature that adjust wireless parameters. We consider three baselines:

• ***overlapping*** **channel baseline:** The APs run in the same channel (channel 1, the channel with the largest number of interfering APs in the neighborhood of our testbed), and with the highest transmission power. Due to the physical configuration of our testbed, this generates the highest level of co-interference generated by the clients.

• ***non-overlapping*** **channel baseline:** The APs are configured in opposite channels (channels 1 and 11), and the highest transmission power. The channels do not overlap, so one client's transmissions do not affect the other's. The full power ensures that the client can achieve the highest possible bitrate and, minus the interference of third party networks.

• ***ACS*** **baseline:** This algorithm is implemented in all Linux-based AP. It selects the best channel based on the automatic channel selection (ACS) function implemented in *hostapd*[3]. It is calculated for each channel, and the algorithm selects the channel with the lowest interference factor. The transmission power is always set to the maximum value.

Finally, the results present the performance of the proposed control loop, which are marked as **Learning**.

B. Control loop hyperparameters

The experiments begin with the system having zero knowledge. We empirically set the hyperparameters as shown in Table V. The value of γ was selected to favor an immediate reward. We wanted the control loop to guarantee a high reward quickly, mainly because an improvement in network performance can be amplified as a result of the DASH adaptability, which thus improves the result for the client. The control loop runtime is 30 seconds, which is 25% of the video playtime, and recent players (e.g. *DASH.js*) can switch to a higher quality segment immediately (even in the middle of a chunk). The control loop is executed every thirty seconds. Therefore, state and reward information is aggregated with this granularity. We will analyze the effect of changing parameters in future work.

C. MOS results

Figure 3 presents the cumulative distribution function (CDF) of MOS_PSNR. The average MOS using our proposal is approximately 4.58 out of 5, while the best result using the baselines is 3.8. The *non-overlapping* baseline obtains better results than the *overlapping* baseline, and yet the control loop gives better results than both. This is true even despite the prediction errors, because there is still a good correlation between the prediction value and the actual MOS_PSNR value. We observed that about 50% of the control loop results reach the maximum MOS, while the *non-overlapping* and *overlapping* baselines only reach this value for 30% and 10% of their results, respectively.

Table III shows the regret obtained during the whole experiments. Regret measures the absolute difference between the sum of the rewards obtained by the strategy adopted by the agent, and the optimal strategy, *i.e.*, a perfect MOS of 5. In other words, lower regrets indicate more effective control loops. Note that these results consider the entire experiment, including the initial learning phase, because the control loop reaches the maximum MOS in about 3 intervals. The table also shows the confidence interval with 95% confidence. The proposed control loop has the lowest regret value. The control loop improves the regret by 65%, 84%, and 82% if compared to the *non-overlapping*, *overlapping*, and *ACS* baselines, respectively. Non-overlapping has lower regret than overlapping, because in the former, the APs are on non-interfering channels. However, we expected *ACS* to behave similar to *non-overlapping*, but its results are close to the *overlapping* configuration. This occurs because in many cases ACS selects the same channel for both APs.

As mentioned previously, one of the objectives of the proposed control loop is to provide a more homogeneous QoE

[3]https://wireless.wiki.kernel.org/en/users/documentation/acs

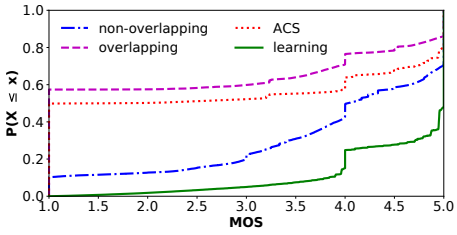

Figure 3. Comparison among the baselines using MOS_PSNR.

Table III
COMPARISON OF THE REGRET OBTAINED USING THE CONTROL LOOP.

Stats	Mean	Impr.
Non-overlapping	1.167 ± 0.024	64.5%
Overlapping	2.602 ± 0.019	84.1%
ACS	2.286 ± 0.029	81.9%
Learning	0.414 ± 0.004	—

Table IV
COMPARISON OF THE FAIRNESS INDEX OBTAINED DURING THE EXPERIMENT.

Stats	Mean	Impr.
Non-overlapping	0.667 ± 0.008	49.0%
Overlapping	0.518 ± 0.006	91.9%
ACS	0.461 ± 0.010	115.6%
Learning	**0.994** ± 0.001	—

Table V
HYPERPARAMETERS

Where	Hyperparameter	Value
TCN	Number of filters	24
	Padding	causal
	Kernel size	2
	Kernel initialization	He normal [22]
	Activation function	relu[1]
	Dilation levels	4
	Dropout	5 %
Optimization	Optimizer	Adam
	Learning rate (α)	0.002
	Epochs	50
Replay	Episodes	10
	Batch size	32
	Capacity	2000
Q-Learning	Initial epsilon (ϵ_0)	0.1
	Epsilon decay (ϵ_δ)	0.995
	Discount factor (γ)	0.95

Notes: [1]The last layer uses a linear activation function.

among the video users. We adopt the Hossfeld fairness index as our measure of homogeneity among clients. Table IV shows the Hossfeld fairness index values for the baselines and the proposed control loop. The row with the mean shows the confidence interval with 95% confidence. For the mean values, the *overlapping* baseline obtained the worst value, while the best value is obtained with the proposed control loop. The control loop improves the fairness index by 49%, 92%, and 116% when compared, respectively, to the *non-overlapping*, *overlapping*, and *ACS* baselines.

Another important measure in control loops in general is how long the control loop takes to converge to its best solution. Figure 4 shows the MOS convergence time. The X-axis shows the timesteps required for convergence (120 timesteps equals one-hour, which is the test maximum execution time). The proposed control loop shows better results than the others, converging in 50% of cases at least twice as fast. Table VI shows the average, the maximum and the percentage improvement in convergence time observed for each case. These results reinforce the fact that our control loop achieves better average MOS than the others during all the experiment.

We observe in our data that using the control loop, whenever the MOS falls below 5, the algorithm can return to the maximum MOS situation on average in 111.0±13.0 seconds as shown in Table VI, *i.e.*, the algorithm can recover to full MOS in up to four interactions (on average) after the disturbance. In our experiments, due to the simplicity of our testbed, the agent reached the maximum MOS, and only in about 20% of cases the MOS drops to less than 4.

VI. RELATED WORK

Reference [23] controls the transmission power using RL, however, their proposal applies to small long term evolution (LTE) cells, and only to voice transmission. Reference [24] also controls the transmission power and use QoE to guide the Q-learning (QL) algorithm, but their proposal alters the bitrate of the downloaded video and data, while ours selects the channel and the transmission power. Their proposal applies to secondary users in 5G cognitive networks, and limits the amount of interference over the primary users below a limit, while our proposal maximizes the QoE and copes with the co-interference from other APs.

In [25] the authors improve the quality of the video stream by allowing the client to request different chunks of one segment of the video using different links. They use policy iteration to find the best action, *i.e.*, select the resolution of chunk for each link. Thus, their proposal is for a multi-link client, which performs load balancing and rate adaptation among the links, but does not control the network.

Some approaches control the video transmission at the source or the destination, while ours uses the default DASH configuration and alters the users connection to the network. Reference [26] uses RL to improve the video quality at the wireless client, and the learner adapts the client requests to the DASH server. Further, it does not control the network. Meanwhile, [20] also uses RL to decide what is the best video quality segment, which is very similar to [26]. Also, the decision is different, the former proposal defines the value needed, while the latter decides if the quality should increase, decrease or remain the same. The authors in [27] alter the video sender bitrate (SBR) at the video server, according to users' QoE requirement, thus they do not control the network. Also, their proposal defines a function that predicts the SBR.

Finally, as far as we know there is only one proposal that modifies the WiFi parameters to improve video quality. EdgeDASH [28] adapts the airtime of clients to improve video quality, together with a client-assisted selection of the quality level being played. Their article assumes a perfect wireless medium, disregarding the interference of neighboring APs.

VII. CONCLUSION AND FUTURE WORK

This paper showed an intelligent control loop to improve QoE in video applications. The control loop employs Deep

Table VI
TIME (S) TAKEN BY THE SYSTEM TO CONVERGE TO MAXIMUM MOS

Stats	Mean	Max	Impr.
Non-overlapping	185.5	810.0	40.5%
Overlapping	252.8	1,260.0	56.3%
ACS	167.3	930.0	34.0%
learning	111.0	420.0	—

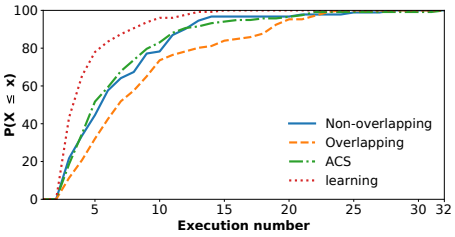

Figure 4. CDF of convergence time to MOS.

Reinforcement Learning to decide the best transmission power and wireless channel to be used on many APs managed by a controller. The video client's QoE is estimated using a MOS model running on the controller, eliminating the need for periodic user feedback.

The proposed control loop improves the regret by up to 84% when compared to the baselines. The control loop also improves the fairness by up to 115%. We simplified the state representation using interpretability techniques. The performance of the system with fewer features is similar to that obtained with the initial proposal. Future work will improve the practicality of the system as well as the models.

VIII. ACKNOWLEDGEMENTS

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq (funding agency from the Brazilian federal government), FAPEMIG (Minas Gerais State Funding Agency), and São Paulo Research Foundation (FAPESP) with Brazilian Internet Steering Committee (CGI.br), grants 2018/23097-3 and 2020/05182-3.

REFERENCES

[1] UKOM Ltd, "Digital Market Overview," June 2020. [Online]. Available: https://ukom.uk.net/uploads/files/news/ukom/206/Digital_Market_Overview_June_2020_v3_exc_May.pdf

[2] Cisco, "Cisco Annual Internet Report (2018–2023) White Paper," March 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] J. B. Ernst, S. C. Kremer, and J. J. Rodrigues, "A survey of QoS/QoE mechanisms in heterogeneous wireless networks," *Physical Communication*, vol. 13, pp. 61–72, 2014.

[4] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective," *Journal of Cloud Computing*, vol. 5, no. 1, p. 4, 2016.

[5] D. N. d. Hora, R. Teixeira, K. van Doorselaer, and K. van Oost, "Predicting the Effect of Home Wi-Fi Quality on Web QoE," in *InternetQoE '16*. New York, NY, USA: ACM, 2016, pp. 13–18.

[6] N. Clinckx and Y. Baffalio, "Optimize network OPEX and CAPEX while enhancing the quality of service," EY, Tech. Rep., 2014.

[7] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-DASH: A Deep Q-Learning Framework for DASH Video Streaming," *IEEE TCCN*, vol. 3, no. 4, pp. 703–718, 2017.

[8] Y. Liu, W. He, Y. Wang, and H. Yang, "Network-Assisted Neural Adaptive Naked-Eye 3D Video Streaming Over Wireless Networks," *IEEE Access*, vol. 7, pp. 141 363–141 373, 2019.

[9] F. Z. Yousaf, O. Mämmelä, and P. Mannersalo, "Reinforcement learning method for QoE-aware optimization of content delivery," in *2014 IEEE WCNC*. IEEE, 2014, pp. 3390–3395.

[10] H. D. Moura, D. F. Macedo, and M. A. M. Vieira, "Wireless Control using Reinforcement Learning for Practical Web QoE," *Computer Communications*, 2020.

[11] Y. D. Goli and R. Ambika, "Network traffic classification techniques – a review," in *2018 CTEMS*, Dec 2018, pp. 219–222.

[12] R. Dubin, A. Dvir, O. Pele, and O. Hadar, "I Know What You Saw Last Minute—Encrypted HTTP Adaptive Video Streaming Title Classification," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 3039–3049, 2017.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 3rd ed. Cambridge, Massachusetts: MIT press, 2018.

[14] Q. Huynh-Thu and M. Ghanbari, "The accuracy of PSNR in predicting video quality for different video scenes and frame rates," *Telecommunication Systems*, vol. 49, no. 1, pp. 35–48, 2012.

[15] S. Huaizhou, R. V. Prasad, E. Onur, and I. Niemegeers, "Fairness in wireless networks: Issues, measures and challenges," *IEEE COMST*, vol. 16, no. 1, pp. 5–24, 2013.

[16] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela, "Definition of QoE fairness in shared systems," *IEEE COMML*, vol. 21, no. 1, pp. 184–187, 2016.

[17] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: Improving bitrate adaptation in the DASH reference player," *ACM TOMM*, vol. 15, no. 2s, p. 67, 2019.

[18] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid – a framework for video transmission and quality evaluation," in *TOOLS*, 2003, pp. 255–272.

[19] E. Demirbilek and J.-C. Grégoire, "Machine Learning–Based Parametric Audiovisual Quality Prediction Models for Real-Time Communications," *ACM TOMM*, vol. 13, no. 2, p. 16, 2017.

[20] L. Amour, M. S. Mushtaq, S. Souihi, and A. Mellouk, "QoE-based framework to optimize user perceived video quality," in *2017 IEEE 42nd LCN*, Oct 2017, pp. 599–602.

[21] G. Miranda, D. F. Macedo, and J. M. Marquez-Barja, "Estimating video on demand qoe from network qos through icmp probes," *IEEE Trans. on Netw. and Serv. Manag.*, vol. 19, no. 2, p. 1890–1902, jun 2022. [Online]. Available: https://doi.org/10.1109/TNSM.2021.3129610

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE ICCV*, 2015, pp. 1026–1034.

[23] F. B. Mismar and B. L. Evans, "Q-learning algorithm for volte closed loop power control in indoor small cells," in *2018 52nd ACSSC*. IEEE, 2018, pp. 1485–1489.

[24] F. S. Mohammadi and A. Kwasinski, "QoE-Driven Integrated Heterogeneous Traffic Resource Allocation Based on Cooperative Learning for 5G Cognitive Radio Networks," in *2018 IEEE 5GWF*. IEEE, 2018, pp. 244–249.

[25] M. Xing, S. Xiang, and L. Cai, "A real-time adaptive algorithm for video streaming over multiple wireless access networks," *IEEE J-SAC*, vol. 32, no. 4, pp. 795–805, 2014.

[26] M. Claeys, S. Latre, J. Famaey, and F. De Turck, "Design and evaluation of a self-learning HTTP adaptive video streaming client," *IEEE communications letters*, vol. 18, no. 4, pp. 716–719, 2014.

[27] A. Khan, L. Sun, E. Jammeh, and E. Ifeachor, "Quality of experience-driven adaptation scheme for video applications over wireless networks," *IET communications*, vol. 4, no. 11, pp. 1337–1347, 2010.

[28] S. Bayhan, S. Maghsudi, and A. Zubow, "Edgedash: Exploiting network-assisted adaptive video streaming for edge caching," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1732–1745, 2021.