Optimizing QoE and Cost in a 3D Immersive Media Platform: A Reinforcement Learning Approach

Panagiotis Athanasoulis*, Emmanouil Christakis†, Konstantinos Konstantoudakis†, Petros Drakoulis†, Stamatia Rizou*, Avi Weit‡, Alexandros Doumanoglou†, Nikolaos Zioulis† and Dimitrios Zarpalas†

*Singular Logic S.A., Athens, Greece, Email: pathanasoulis@ep.singularlogic.eu, srizou@singularlogic.eu

†Visual Computing Lab (VCL), Information Technologies Institute (ITI),
Centre for Research and Technology - Hellas (CERTH), Thessaloniki, Greece
Email: {manchr, k.konstantoudakis, petros.drakoulis, aldoum, nzioulis, zarpalas}@iti.gr

‡IBM Research, Haifa, Israel, Email: weit@il.ibm.com

Abstract*—Recent advances in media-related technologies, including capturing and processing, have facilitated novel forms of 3D media content, increasing the degree of user immersion. In order to ensure these technologies can readily support the rising demand for more captivating entertainment, both the production and delivery mechanisms should be transformed to support the application of media or network-related optimizations and refinements on-the-fly. Network peculiarities deriving from geographic and other factors make it difficult for a greedy or a supervised machine learning algorithm to successfully foresee the need for reconfiguration of the content production or delivery procedures. For these reasons, Reinforcement Learning (RL) approaches have lately gained popularity as partial information on the environment is enough for an algorithm to begin its training and converge to an optimal policy. The contribution of this work is a Cognitive Network Optimizer (CNO) in the form of an RL agent, designed to perform corrective actions on both the production and consumption ends of an immersive 3D media platform, depending on a collection of real-time monitoring parameters, including infrastructure, application-level and quality of experience (QoE) metrics. Our work demonstrates CNO approaches with different foci, i.e., a greedy maximization of the users' QoE, a QoE-focused RL approach and a combined QoE-and-Cost RL approach.

***Index Terms*—immersive media; cognitive network optimizer; reinforcement learning.**

I. Introduction

New forms of interactive 3D media applications have lately emerged as a result of advances in processing, 3D capturing and imaging technologies. These applications include mixed reality and tele-immersive platforms that allow the embedding of real world entities into the virtual world in a real time and interactive way, thus creating a more engaging user experience.

Media applications are justifiably known as some of the most demanding and computationally intensive services, imposing tough challenges on allocation and management procedures, concerning both computing and network resources. State-of-the-art forms of 3D media, ranging from virtual worlds to games, necessitate the rethinking of media production and distribution [1], with the user's Quality-of-Experience (QoE) playing an increasingly dominant role due to the increased level of user immersion. At the same time, it comes as no surprise that the production and delivery costs are rising in order to cater for these novel forms of 3D media content. Ergo, the composition of a real-time, QoE-and-Cost unified, optimization approach is considered essential for these services to remain both affordable and enjoyable. On this ground, recent advancements in the 5G field could provide such enhancements, unleashing the potential of 3D immersive media content, and the media industry in general, through dynamic, real-time refinements and/or reconfiguration of underlying services [2].

In this work, we simulate a 3D-immersive gaming session with all its required components, including simulated consumers of various processing and bandwidth constraints. Our aim is to develop a Cognitive Network Optimizer (CNO) system that manages to balance the consumers' QoE and the quality-transcoding costs. The most common approach would be to either design the CNO in a greedy manner, leveraging statistical knowledge of the overall performance of our targeted system, or apply a machine learning algorithm. Nevertheless, network peculiarities deriving from geographic and other factors, render the application of a supervised machine learning algorithm possibly untrustworthy. A supervised model would heavily depend on the diversity of the networking dataset that would be used, hence, the same model might not perform adequately well across all ranges of network characteristics. On this ground, a Reinforcement Learning (RL) approach was adopted for the design of the aforementioned CNO system, as this kinds of algorithms do not necessarily need prior knowledge in order to perform. Once deployed, a RL agent can learn from the environment through trial-and-error, by receiving positive or negative rewards upon its actions. With RL being the backbone of our envisioned CNO system, it would be possible to present a unified approach for this type of interactive application, possibly able to perform reasonably under any network or processing capacity circumstances.

The rest of this paper is organised as follows: Section II presents the related work while Section III provides insights on the overall system architecture. Section IV depicts the approach that was followed for the design and implementation of the CNO along with essential details. Sections V and VI present the experimental setup and the corresponding results respectively. Finally, Section VII concludes this paper.

II. Related Work

The perceived quality for images and videos has been heavily investigated in the literature and standardized by the International Telecommunication Union (ITU) [3]. However, these recommendations do not readily apply in the case of interactive gaming and 3D reconstruction content. Various works have attempted to model gaming QoE [4]–[7]. We chose to adopt the model described in [7] because the metrics required in order to determine the perceived QoE are readily available in our platform. Although this work examines gaming in its traditional video form, rather than 3D media content, we decided to ignore this discrepancy for two reasons. Firstly, the literature for 3D media quality assessment is still very limited [8]–[10]. It is widely acceptable that formulating a comprehensive QoE model for this kind of media is quite a challenging task. Secondly, the main focus of our work is to demonstrate the capabilities of our CNO system to balance the customers' QoE and the cost of the production platform. This implies that, in principal, any legitimate QoE model could be adopted by our system, thus it is not our focus.

A substantial amount of work has been conducted on adaptive video streaming. The most established technique for this task is MPEG-Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [11]. Recent works have exploited RL to train systems that can accomplish adaptation and have been able to surpass traditional methods in terms of users' QoE [12]–[15], yet, these efforts have solely focused on maximizing the client's satisfaction and have not attempted to balance other factors like the cost of production, which is of great importance for video service providers. Most closely related to our work are [16] and [17], both of which aim to find a balance between cost and QoE. However, contrary to them, we utilize RL to optimize our system, and the content on which our optimizations are performed is 3D reconstructed.

III. System Architecture

The envisioned use-case incorporates a tele-immersive interactive multiplayer video game named "Space Wars", which is developed by [18]. In this application, two players physical appearance is embedded inside a common virtual environment where they interact with each other to play a capture-the-flag style VR game. Players silhouette and texture coverage is being captured using a set of four color-depth (RGB-D) cameras per player, arranged in a square around them. In addition to the players, the system can also accommodate a potentially large number of remotely allocated live spectators, which can be arbitrarily distributed across a wide geographic area. This pushes further the envelope of the underlying infrastructure to include the need for very high bandwidth and near real-time latency, along with highly reliable large-scale delivery. Different parties may have different visual quality requirements, driven by devices capabilities, network conditions or subscription privileges. Thus, the production data streams need to be transcoded into various quality levels in real-time speed, and be concurrently available for consumption in an Adaptive-Bit-Rate (ABR) manner [19].

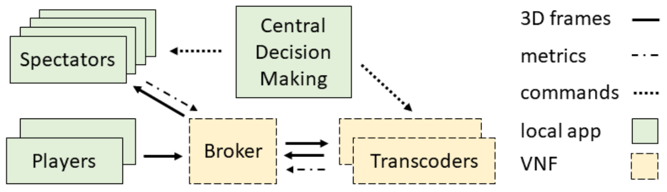


Figure 1. The flow of 3D video frames from the players to the spectators, along with the central decision commands and the metrics that drive them.

In contrast to a typical 3D content delivery pipeline, the examined prototype utilizes many of the forthcoming 5G mannerisms to mitigate the conflicting nature of certain requirements in a resource-efficient, business-friendly way. The opted architecture steps on the model described in [20], where transcoding components are built as Virtual Networking Functions (VNF) and deployed on the 5G-compliant cloud-computing infrastructure (NFVI) opportunistically, in a quality / cost optimized manner. Optimizations may apply in both infrastructure and software level, with only the latter being in scope of this work in the form of management of the quality-levels production and distribution. Optimization strategy and all of the decision-making mechanism is embodied in the CNO component, which translates various network and application-level input measurements to parametrization actions that steer virtual topology towards maximizing the global QoE / cost ratio. A schematic representation of the architecture can be found in Figure 1.

IV. Cognitive Network Optimizer

The role of the CNO is to decide about essential optimizations and corrective actions that should be applied upon both the production and consumption ends of the targeted system, i.e, spectators and transcoders. The CNO, implemented as a RL agent, receives a combination of infrastructure, application-level and QoE metrics, and executes a list of optimization actions. This section provides insights on the CNO's design and implementation, including decisions regarding the implemented algorithm, the selected monitoring parameters and the supported actions.

A. *Reinforcement Learning: State-space, Modelling & Reward*

Reinforcement Learning is a process of learning in which a situation is mapped to an action in order to maximize either a specific numerical or abstract reward. No advice is given to the entity following the learning procedure, the learner, regarding which actions to take. Instead of this, the learner should discover the rewards that yield from the available actions [21]. The foundation behind most RL algorithms is a Markov Decision Process, composed of:

- a finite number of states S;
- a finite number of actions A;
- a transition function $T : S \times S \times A \rightarrow [0, 1]$ assigning a probability distribution over states;
- a reward function $R : S \times A \times S \rightarrow \mathbb{R}$ giving the immediate reward received after each transition.

The goal of a RL algorithm is to learn a mapping from states to actions, or policy, π. The optimal value of an action a taken from state s, denoted by $V^*(s)$, expresses the expected sum of rewards discounted by a factor γ, that an agent would receive when, starting from state s, the agent performs an action a and follows the optimal policy. The aforementioned values are connected through the following equations:

$$Q^*(s,a) = \sum_{\forall s' \in S} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \quad (1)$$

$$V^*(s) = \max_{a \in A(s)} \sum_{\forall s' \in S} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \quad (2)$$

$$\pi^*(s) = arg \max_{a \in A(s)} \sum_{\forall s' \in S} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \quad (3)$$

The optimal values of the states are the solutions of equation (2). Given the optimal values of the states, the optimal policy is then defined as shown in equation (3).

State-space. In order to capture the complexity of the system in detail, a fine-grained segmentation of the state-space would be required, making the RL agent impractical in terms of required computational resources and training time. Instead of this, a more coarse-grained segmentation of the state-space was preferred. Having gained a thorough understanding of the overall system's performance, leveraging statistical characteristics, and having captured the full value range of the selected monitoring parameters, levels were statically defined for each of these parameters, thus forming a compact, computational resource and training time efficient, state-space.

Modelling Approach. The optimal policy can be calculated following either a model-based or a model-free approach. The model-free approach focuses on the effectiveness of the executed actions, with Q-learning algorithm [22] being its most common representative, an efficient algorithm in terms of required computational and memory resources. Nevertheless, Q-learning performs only local updates to the values, hence only the policy of the initial state can be updated at each step and a large amount of experiences is required to converge to an optimal policy. In the model-based approach, the RL agent tries to model the exact behavior of the environment, thus this approach with Prioritized Sweeping was preferred. Prioritized Sweeping uses careful bookkeeping to concentrate the computational effort on the most interesting parts of the system [23]. After a transition, its probability estimate is updated along with the transition probabilities of previously observed successors.

Reward Function. The reward function is one of the most vital elements of a RL algorithm. It should be designed carefully as it is capable of either assisting the fast convergence of the algorithm or leading to false optimal policies [24]. In our CNO system, the reward function is composed of five reward components aiming for optimizations on different parts of the system: *a) GPU usage*, positive if the application is deployed on a CPU-only node, due to the substantially low cost of such a node, or a GPU-node is used by the majority of spectators, and negative otherwise, *b) QoE of single spectator*, defined as the percentage of increment or decrement of QoE of a spectator after the execution of a certain action, *c) Combination of QoE & transcoding cost*, positive in case the QoE sum of all spectators is greater than the transcoding cost, or negative otherwise, *d) Monitoring parameters*, depending on the percentage of increment or decrement of selected parameters, namely, a spectator's bit rate and frame rate, following the execution of a certain action, and *e) Number of produced profiles*, which gives a positive reward if the number of produced profiles is reduced and a negative reward if it is increased. Of the aforementioned reward components *(b)* and *(d)* are focused on the experience of a single spectator, while *(a)*, *(c)* and *(e)* are focused on the performance and cost of the overall system. Adjustments of the reward components' weights will force the CNO to focus on specific aspects of the system while performing the necessary optimizations.

B. Monitoring Parameters & Optimization Actions

This subsection presents the monitoring parameters that were selected for collection and input to the CNO, along with the optimization actions that are supported by the infrastructure. For the metrics related to containers running on Kubernetes, Prometheus [25] has been employed and two derived metrics expressing the packet loss of the transmitted and received network packets were exported. Besides the Prometheus-exported metrics, application-level metrics are exported from both the spectators and the transcoder VNFs. Moreover, the Mean Opinion Score (MOS) value is computed based on the frame rate and recorded PSNR as described in [7]. Table I presents a list of all metrics used in the optimization process, along with their origin.

The available actions that are selected and executed by the implemented RL algorithm are summarized in Table II. These actions can command *a)* a spectator to start consuming from a specific produced profile (*set_vtranscoder_client_profile*), *b)* a transcoder to start or end production of specific profiles (*set_vtranscoder_profile*), or *c)* a transcoder to migrate from a CPU-only to a GPU-node, or vice-versa (*set_vtranscoder_processing_unit*). Additionally, it is possible that the CNO will detect no need for optimization (*no_operation*). It should be clear that the CNO makes de-

TABLE I
UTILIZED MONITORING METRICS

Origin of Metric	Metric Description
Prometheus (derived metrics)	Transmitted Network Packet Loss Received Network Packet Loss
Spectator	Bit Rate Bit Rate (Aggregated) Frame Rate Frame Rate (Aggregated) Consumed Profile
vTranscoder	Number of Produced Profiles Output Data Bytes Working Frames per Second Theoretic Load Percentage
QoE	Mean Opinion Score

TABLE II
OPTIMIZATION ACTIONS AND SUPPORTED VALUES

Action	Target	Values
set_vtranscoder_client_profile	Spectator	$[0, 1, 2, 3, 4, 5]$
set_vtranscoder_profile	vTranscoder	$[1, 2, 3, 4, 5]$
set_vtranscoder_processing_unit	vTranscoder	[cpu, gpu]
no_operation	-	-

cisions for an action that must be executed on a spectator, yet it is responsible for executing all the essential prerequisite actions as well.

C. Algorithmic Flow

With a certain level of abstraction, the operation of the CNO is described by the algorithm presented in Figure 2. To begin with, the necessary monitoring metrics, as listed in Table I, are collected and the MOS value is computed. This collection of metrics forms a feature vector and is fed to the algorithm. This vector is unequivocally mapped to a state. The determination of the overall system's current state is followed by the establishment of the corrective actions to be executed. A little while after the optimal action's execution, a new set of monitoring metrics will be collected, a new MOS value will be computed and the newly formed feature vector will once again be mapped to a state. Consequently, the reward is computed according to the pre-action and post-action measurements. The latest experience in the form *(pre_action_measurements, action, post_action_measurements, reward)* is recorded and the model values are updated using Prioritized Sweeping.

```
1: $produced\_profiles \leftarrow [p_0]$
2: $GPU\_profiles \leftarrow [p_k, ..., p_{k+n}]$
3: $pu \leftarrow CPU$
4: $initial\_metrics \leftarrow get\_collected\_measurements()$
5: while $True$ do
6:     $s \leftarrow get\_state(initial\_metrics)$
7:     $a, p \leftarrow get\_suggested\_action(s)$
8:     if $p \in produced\_profiles$ then
9:         $set\_vtranscoder\_client\_profile(p)$
10:    else
11:        if $p \in GPU\_profiles$ and $pu = CPU$ then
12:            $set\_vtranscoder\_processing\_unit(GPU)$
13:            $pu \leftarrow GPU$
14:        $produced\_profiles.append(p)$
15:        $set\_vtranscoder\_profiles(produced\_profiles)$
16:        $set\_vtranscoder\_client\_profile(p)$
17:    $produced\_profiles \leftarrow get\_consumed\_profiles()$
18:    $set\_vtranscoder\_profiles(produced\_profiles)$
19:    if $pu = GPU$ and $GPU\_not\_needed()$ then
20:        $set\_vtranscoder\_processing\_unit(CPU)$
21:        $pu \leftarrow CPU$
22:    $sleep()$
23:    $after\_metrics \leftarrow get\_collected\_measurements()$
24:    $r \leftarrow get\_reward(initial\_metrics, after\_metrics)$
25:    $update\_model()$
26:    $initial\_metrics \leftarrow after\_metrics$
```

Figure 2. CNO Algorithm

V. EXPERIMENTAL SETUP

To develop our CNO, we simulate the various components required for a "Space Wars" two-players-with-spectators game scenario. For the players, we used an actual game recording between two people. The 3D models and textures of the players are recorded as two streams, one for each player. The resulting streams contain a number of consecutive frames with each frame consisting of four RGB images of a player captured from different angles and a 3D mesh. During the simulation, these two streams are published on a Kafka broker. Then, two transcoders connect on the same broker and each one receives a different stream to produce the extra qualities required to be consumed by the spectators.

The non-transcoded streams deriving directly from the players, namely "passthrough", are produced using jpeg compression for the textures and Google's Draco compression for the meshes. The transcoders can potentially produce five extra qualities. Two of them use jpeg to compress as still images down-scaled versions of the original textures along with more heavily quantized versions of the original meshes, and can be produced solely using the CPU. Another three profiles utilize the HEVC algorithm to encode textures as video sequences, also together with more pronouncedly quantized versions of the meshes, and deliver much higher compression ratios but can only be produced using GPU infrastructure.

All transcoded streams, together with the "passthrough", are published on the broker. The last piece, spectators, also connect to the broker and receive one of the produced qualities. For our experiments, we simulate the spectators so that we can easily modify their bandwidth and processing capabilities, creating various profiles that correspond to real viewing scenarios. The bandwidth of our spectators is set to three different levels: 20, 40 and 60 Mbps, and they can have two different processing capabilities; one corresponding to a low-end mobile user and one to a high-powered desktop one. The processing power of a spectator determines how fast he can decode the incoming frames and in return the frames-per-second (fps) with which he can watch the game. The spectators publish live metrics (downloading bit-rate, viewing fps) on the broker.

Overall, the CNO system analyzes the spectators' metrics and issues a) a command to the transcoders, dictating which qualities they have to produce and b) a command to each spectator to indicate which quality they must consume from every transcoder. As mentioned before, the aim of the CNO is to balance users' QoE with production cost. The production cost is determined by which and how many qualities a transcoder has to produce, directly affecting the underlying CPU or GPU infrastructure needed. A quality that requires GPU to be produced is much more expensive, as is the case on actual commercial services like the Amazon Web Services (AWS), where the leasing price of nodes equipped with GPUs is up to 15 times higher than of nodes without.

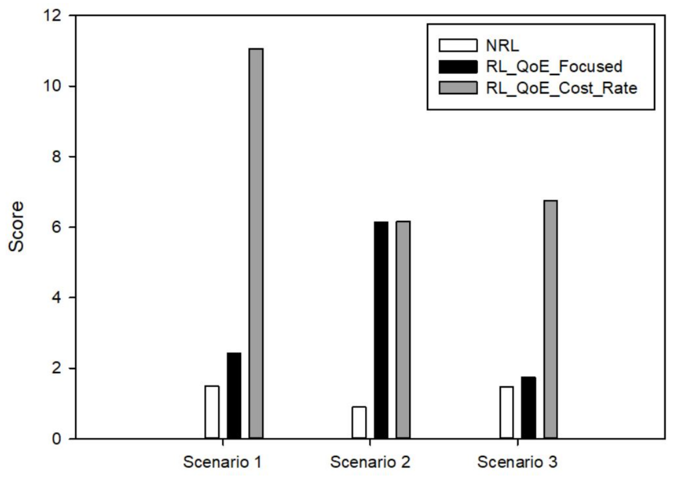


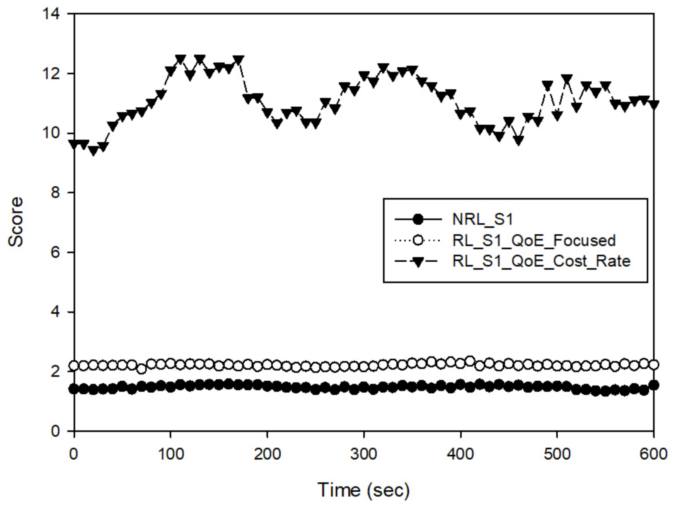
Figure 3. Comparison of the average score achieved at each execution scenario for each CNO configuration

Finally, three CNO versions were examined, namely the *NRL*, implemented as a greedy algorithm targeting on the maximization of the a spectator's QoE, the *RL_QoE_Focused*, implemented as a RL agent with increased weights on its QoE-based reward component, and a combined *RL_QoE_Cost_Rate* version performing joint optimizations between the spectators' QoE and the overall production costs. These CNO versions were executed upon three spectator scenarios. Scenario 1 included spectators of high bandwidth and processing power, Scenario 2 included spectators on the other side of the spectrum, i.e., with low bandwidth and processing power, while Scenario 3 was a mixture of both, also including spectators with moderate bandwidth. For these experiments, we defined the score as the QoE sum of all running spectators divided by the production cost.

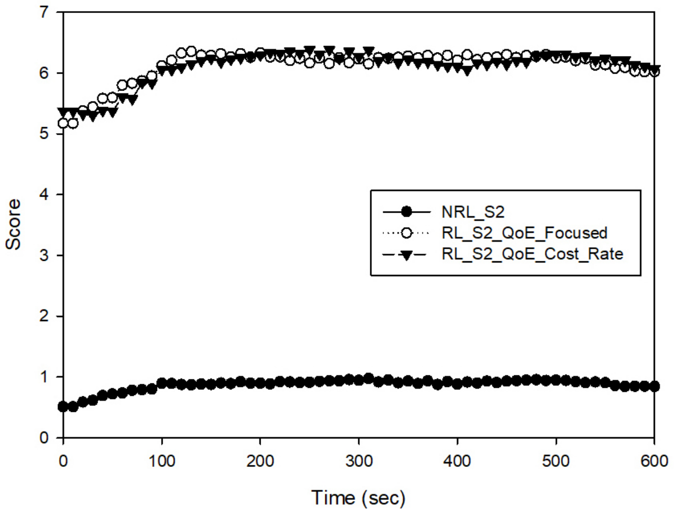
VI. Results

This section presents and discusses the results of the executed experiments. Figure 3 offers a performance comparison of each of the CNO configurations, for all three of the different spectator scenarios, while Figures 4a, 4b, and 4c present the progress of score through time, for each scenario. *NRL* receives the lowest average score in all scenarios, while *RL_QoE_Cost_Rate* achieves the highest score. At the same time *RL_QoE_Focused* version is performing well only in Scenario 2, while being marginally better than *NRL* in Scenarios 1 & 3.

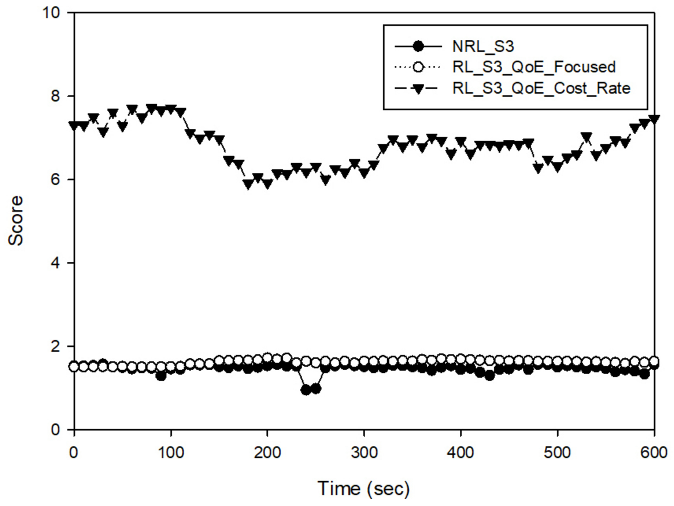
The *NRL's* bad performance is due to its sole focus on the maximization of the spectators' QoE, without consideration of the production cost. *NRL* constantly commands the transcoders and spectators to, respectively, produce and consume the most expensive profiles, i.e., profiles that are exclusively produced by GPUs, as these emit the maximum QoE, without considering the higher production costs induced by the GPU's usage, thus reducing its overall score. For this reason, *NRL* achieves its lowest score on Scenario 2, as the GPU's usage makes no difference for these - low bandwidth and processing power - spectators, which cannot keep up with the production under no



(a) Scenario 1: Spectators with high bandwidth and high processing power



(b) Scenario 2: Spectators with low bandwidth and low processing power



(c) Scenario 3: Combination of bandwidth levels and processing power

Figure 4. Observation of score through time

circumstances. Since these spectators have limited potentials, the best choice in this scenario would be to consume the low-cost CPU-produced profiles, and stop production of the more expensive ones.

Concerning the *RL_QoE_Focused* version, its performance appears almost as bad as the *NRL's* performance in Scenarios 1 & 3. It is thought that in both of these scenarios, the algorithm's decisions were dominated by the high-bandwidth and processing power spectators. Hence, in Scenario 3, the spectators of far inferior capabilities did not effectively affect the CNO's decisions. This statement is supported by the good performance of this version in Scenario 2. Upon detecting spectators with limited capabilities, the algorithm commanded that one of the low-cost CPU profiles should be consumed, thus reducing the overall production cost and increasing score.

Finally, *RL_QoE_Cost_Rate*, considering the spectators' QoE and production cost of as two factors of equal importance, is the most balanced approach and achieves the highest overall score in all scenarios, as originally expected.

VII. Conclusion

The contribution of this work is the implementation and study on the performance of a CNO, implemented as a RL agent aiming for the joint optimization of users' QoE with respect to the production costs. The suggested CNO's operation was validated upon a 3D media platform, in the form of a tele-immersive interactive multiplayer video game with live spectators, adopting many of the forthcoming 5G mannerisms such as supporting dynamic network refinements and reconfiguration in real-time. Three optimization algorithms were examined, a greedy QoE-focused algorithm, a RL algorithm prioritizing the QoE maximization and a combined QoE-and-Cost RL approach. The latter one proved superior in all executed scenarios, in terms of users' QoE / production cost maximization, with QoE-focused RL version being marginally better than the non-RL greedy implementation as well.

Acknowledgment

This research has been partially supported by the European Commission funded program 5G-Media under H2020 Grant Agreement 761699.

References

[1] P. Daras and F. Alvarez, "A future perspective on the 3d media internet." in *Future Internet assembly*, 2009, pp. 303–312.

[2] S. Rizou *et al.*, "A service platform architecture enabling programmable edge-to-cloud virtualization for the 5g media industry," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2018, pp. 1–6.

[3] T. ITU, "Opinion model for video-telephony applications," *ITU-T Recommendation P. 1070*, 2007.

[4] I. Slivar, M. Suznjevic, and L. Skorin-Kapov, "The impact of video encoding parameters and game type on qoe for cloud gaming: A case study using the steam platform," in *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, May 2015, pp. 1–6.

[5] I. Slivar, L. Skorin-Kapov, and M. Suznjevic, "Cloud gaming qoe models for deriving video encoding adaptation strategies," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 18:1–18:12. [Online]. Available: http://doi.acm.org/10.1145/2910017.2910602

[6] S. Wang and S. Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, Nov 2009, pp. 1–7.

[7] S. Zadtootaghaj, S. Schmidt, and S. Mller, "Modeling gaming qoe: Towards the impact of frame rate and bit rate on cloud gaming," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, May 2018, pp. 1–6.

[8] A. Doumanoglou, N. Zioulis, E. Christakis, D. Zarpalas, and P. Daras, "Subjective quality assessment of textured human full-body 3d-reconstructions," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, May 2018, pp. 1–6.

[9] A. Doumanoglou *et al.*, "Quality of experience for 3-d immersive media streaming," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 379–391, June 2018.

[10] E. Alexiou and T. Ebrahimi, "On subjective and objective quality evaluation of point cloud geometry," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, May 2017, pp. 1–3.

[11] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, April 2011.

[12] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 197–210. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098843

[13] M. Claeys, S. Latr, 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, April 2014.

[14] M. Xing, S. Xiang, and L. Cai, "Rate adaptation strategy for video streaming over multiple wireless access networks," in *2012 IEEE Global Communications Conference (GLOBECOM)*, Dec 2012, pp. 5745–5750.

[15] V. Menkovski and A. Liotta, "Intelligent control for adaptive video streaming," in *2013 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2013, pp. 127–128.

[16] J. He, Y. Wen, J. Huang, and D. Wu, "On the costqoe tradeoff for cloud-based video streaming under amazon ec2's pricing models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 669–680, April 2014.

[17] Y. Zheng *et al.*, "Online cloud transcoding and distribution for crowdsourced live game video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1777–1789, Aug 2017.

[18] K. Christaki, K. Apostolakis, A. Doumanoglou, N. Zioulis, D. Zarpalas, and P. Daras, "Space wars: An augmentedvr game," 2018.

[19] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over http," *IEEE Communications Surveys Tutorials*, vol. 19, pp. 1842–1866, 2017.

[20] A. Doumanoglou *et al.*, "A system architecture for live immersive 3d-media transcoding over 5g networks," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2018, pp. 11–15.

[21] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[22] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[23] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine learning*, vol. 13, no. 1, pp. 103–130, 1993.

[24] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Reward function and initial values: better choices for accelerated goal-directed reinforcement learning," in *International Conference on Artificial Neural Networks*. Springer, 2006, pp. 840–849.

[25] Prometheus monitoring system & time series database. [Online]. Available: https://prometheus.io/