IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 7, 2019, accepted October 27, 2019, date of publication November 4, 2019, date of current version November 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2951219

Deep-Q-Network-Based Multimedia Multi-Service QoS Optimization for Mobile Edge Computing Systems

BOREN GUO, XIN ZHANG, YAXIN WANG, AND HONGWEN YANG

School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Xin Zhang (zhangxin@bupt.edu.cn)

This work was supported by the National Science and Technology Major Project under Grant 2018ZX03001024-006.

ABSTRACT By offloading storage and computing resources to the edge of networks, mobile edge computing (MEC) is emerged as a promising architecture to reduce the transmission delay and bandwidth waste for mobile multimedia services. This paper focuses on a multi-service scenario in the MEC systems, where the MEC server can provide three multimedia services including live streaming, buffered streaming and low latency enhanced mobile broadband applications for edge users at the same time. In order to satisfy various quality of service (QoS) requirements for different multimedia applications, the 5G QoS model is applied. Notably, the packets from the multimedia applications with the same or similar requirements are mapped into the same QoS flow, and each QoS flow is processed individually. Therefore, how to effectively schedule the limited radio resource for QoS flows is an intractable problem. To address the problem above, a QoS evaluation model is designed, and a QoS maximization problem is formulated. Furthermore, a deep reinforcement learning method, deep-Q-network, is adopted to make decisions to allocate radio resource dynamically. Compared with round-robin and priority-based scheduling algorithms, the simulation results validate that the proposed algorithm outperforms other resource scheduling algorithms for multi-service scenario.

INDEX TERMS Deep-Q-network, deep reinforcement learning, mobile edge computing, multimedia, quality of service.

I. INTRODUCTION

Triggered by the rapid advance of diverse smart devices, the mobile data traffic is experiencing a tremendous growth. Cisco's forecasts [1] predicted that mobile data traffic will increase sevenfold between 2017 and 2022, which will bring significant challenges to wireless networks. Besides, with the enhancement of 5G technologies, e.g., network slice [2], device-to-device communication [3]–[5], edge caching [6], Internet of things [7], [8] and so forth, many streaming media services emerge [9], [10]. Specifically, various multimedia applications from massive mobile devices pose diverse requirements (e.g., ultra-low latency, high bitrate, etc.) on Radio Access Network (RAN).

Driven by the strict requirements of multimedia services, mobile edge computing (MEC) [11]–[14] has emerged as a crucial paradigm, where service environment and cloud computing capabilities are deployed to the edge within the RAN. Since storage, processing capacity and computing resource are moved closer to mobile users, MEC architecture can bring the possibility to enable multimedia services that require low latency and high bitrate, such as Augmented Reality (AR)/Virtual Reality (VR) services.

Attracted by the advantage of MEC, more and more researchers pay attention to MEC applications in recent years. MEC architecture is introduced and applied in various scenarios, such as edge caching [15]–[19], service deployment [20], [21], low latency data transmission [20]–[22], video on demand (VoD) [17], [23], high bitrate multimedia streaming [24], [25], security management [19], [26], and other multimedia services. In addition, to figure out complex issues in MEC systems, machine learning (ML) techniques are adopted [19], [27], [28].

From the previous works, researchers focus on a single service with specific quality of service (QoS) requirements. To the best of our knowledge, there are few researches

The associate editor coordinating the review of this manuscript and approving it for publication was Dapeng Wu.

on multimedia multi-service in MEC system, where the MEC server can provide multiple services with various QoS requirements for edge users. However, the QoS flow is the finest granularity of QoS differentiation in the 5th generation (5G) wireless systems [29], and the 5G QoS model is employed in this paper. In other words, data packets from various multimedia applications are mapped into different QoS flows, and those packets are classified by their QoS characteristics. Multimedia data packets in a QoS flow are marked with a distinct ID. In the gNodeB (gNB), processing resource of RAN infrastructure is allocated to each QoS flow to satisfy its distinct QoS requirements respectively. Hence, how to effectively allocated the limited RAN resource to those QoS flows effectively is an intractable problem.

This paper investigates multimedia multi-service in the 5G MEC system. Notably, the MEC server can provide three typical services, i.e., live streaming, buffered streaming and low latency enhanced mobile broadband (eMBB) applications. Besides, the packets scheduling mechanism and QoS model in MEC system are analyzed. Based the QoS evaluation model, the resource allocation problem above can be formulated as a multimedia multi-service QoS optimization problem. Inspired by the success of ML in settling complex problems, a deep reinforcement learning technique, deep-Q-learning (DQN) [30], is employed to solve the QoS optimization problem in MEC systems. To be more specific, four crucial ingredients, i.e., actions, states, observations and rewards are designed for DQN, and then a DQN based resource allocation algorithm is proposed. Adopting DQN technique, RAN resource is allocated dynamically to different QoS flows in the light of system states. Besides, compared with other resource allocation methods including round-robin scheduling and priority-based scheduling, the performance of the proposed DQN based algorithm is evaluated in terms of QoS evaluation value, packet delay, packet loss and throughput. The simulation results validate that the proposed DQN based algorithm can efficiently allocate RAN resource to different QoS flows to meet diverse QoS requirements, especially in high bitrate scenarios.

The main contributions of this paper can be summarized as follows.

- To the best of our knowledge, this paper is the first research which focuses on the 5G QoS model [29], where the packets from various applications are mapped to different QoS flows with different QoS characteristics. Notably, a multimedia multi-service scenario including live streaming, buffered streaming and eMBB applications is considered.
- A QoS evaluation model for multi-service is designed, and then a multi-service QoS optimization problem is formulated. To address the optimization problem, DQN framework is applied to allocate RAN resource to QoS flows dynamically.
- This paper evaluates the convergence of the proposed DQN based algorithm. Compared with round-robin and priority-based scheduling algorithms, the performance of the proposed algorithm is validated in terms of average QoS, average packet delay, average packet loss ratio and throughput.

The rest of this paper is arranged as follows. The relevant researches on multimedia applications in MEC systems are reviewed in the next section. Section III presents the QoS model of the multi-service MEC system, including QoS flows, RAN resource mapping and packets scheduling mechanism. Then, a multi-service QoS optimization problem is also formulated in Section III. To solve the optimization problem above, a DQN based dynamic resource allocation algorithm is proposed in IV. Section V evaluates the performance of the proposed resource allocation method. Finally, Section VI concludes this paper.

II. RELATED WORKS

Authors in [15] designed an Edge-Boost caching algorithm for multimedia delivery with MEC in 5G device-to-device network. Jointly minimizing the peak population of unserved clients and the number of replicas, the proposed caching algorithm obtained better delay reduction and cache utilization. Reference [16] studied how to minimize the traffic between content delivery network (CDN) and network edge, and support transparently and dynamically switch between multiple CDNs to keep QoS rates. Reference [17] solved a joint optimization problem of caching and transcoding for video on demand (VoD) services in MEC systems. MEC is also applied to adaptive video streaming in [18], and authors explored how to jointly maximize quality of experience (QoE) and minimize backhaul traffic for device-to-device communication.

The work presented in [20] studied the deployment of gNBs and MEC points to support low latency services, such as ultra-reliable and low latency communications (URLLC), AR/VR and so forth. Aiming to minimize the traffic load caused by services forwarding, authors in [21] proposed a decentralized algorithm to help base station make the decisions of services placement and user association. Taking into account data compression/decompression and data transmission delay, [22] addressed the multimedia compression strategy for MEC systems to minimize the end-to-end latency.

Considering the distortion rate of VoD and the coordination among MEC servers, [23] studied how to maximize QoE for VoD streaming in MEC. MEC is utilized in Olympic Stadium scenario [24], which can be seen as an eMBB use case. An NFV-based MEC is proposed to transmit ultra-high quality multimedia streaming [25], such as 4K and 8K video. As an application of MEC in the therapy field, [26] presented an in-home therapy management framework for on-demand data-sharing scenario.

The work in [19] investigated how to protect MEC systems from various attacks, and a reinforcement learning (RL) based security solutions is given. By using MEC, [27] achieved high recognition accuracy and low recognition time in video surveillance systems. To optimizing offloading decision and image compression parameter, a Q-learning approach is utilized in [27]. Addressing the trade-off between

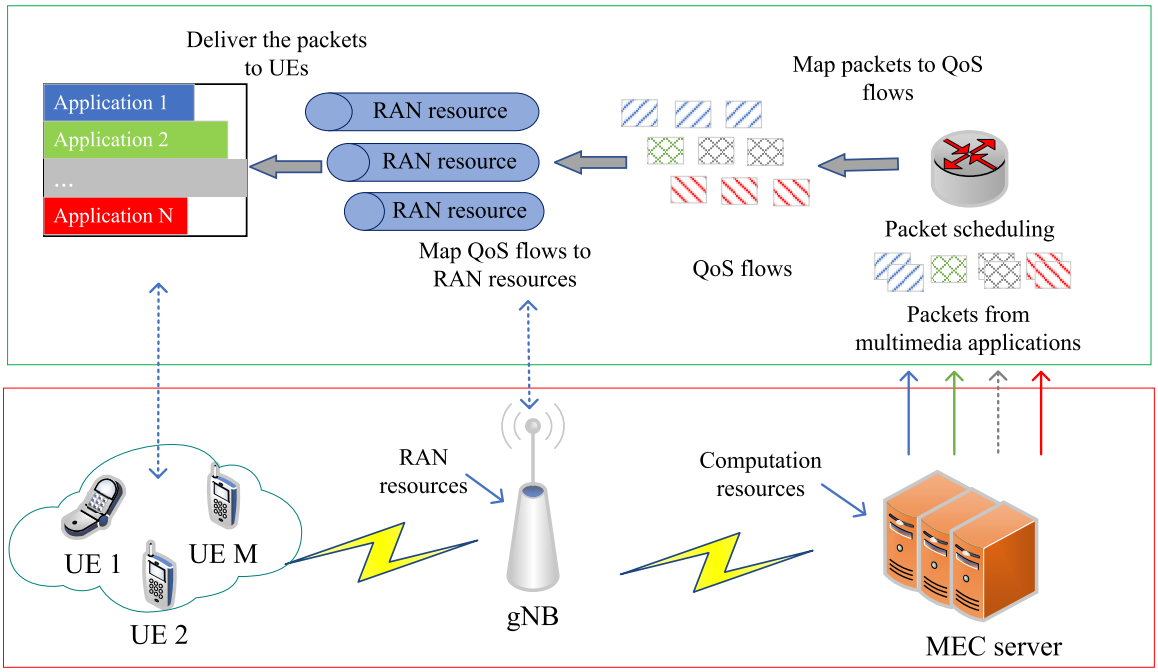


FIGURE 1. Multimedia multi-service in MEC system.

energy-efficiency and processing-throughput, authors in [28] proposed a ML based approach to make decisions (i.e., offloading data processing to an edge or at core). Besides, this work also presented that MEC can be adopted for energy conservation.

Unlike the works above, this paper considers a multi-service scenario in MEC system. Notably, multi-service consists of live streaming (e.g., live broadcast, interactive gaming), buffered streaming (e.g., www, ftp, p2p file sharing, progressive video) and low latency eMBB applications (e.g., AR/VR). Those services are characterized by different QoS requirements. How to allocate the shared RAN resource to those services are addressed in this paper.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, a multi-service scenario in MEC system is introduced first. Then, the QoS flow mapping and packets scheduling mechanism are presented. In addition, in order to quantify the bitrate and delay characteristics of QoS flows, the traffic model and delay model are given respectively. Finally, a multi-service QoS optimization problem is formulated.

A. MULTI-SERVICE SCENARIO IN MEC SYSTEM

As shown in Fig. 1, this paper considers a scenario where a MEC server can provide various multimedia applications for multiple user equipments (UEs). The data packets from multimedia applications are mapped into several QoS flows, according to their different QoS characteristics. Then, the data packets in different QoS flows are processed in the gNB and delivered to UEs. In other words, the QoS flows shared the same radio infrastructure and RAN resource. Following a specific resource allocation scheme, the QoS flows can be mapped to different RAN resource segments. Therefore, how to adjust resource allocation scheme to optimize the system performance is investigated.

Formally, this paper considers a scenario where M UEs can request N different multimedia applications from the MEC server at the network edge. Besides, the data packets from those applications can be mapped into three QoS flows with different QoS characteristics, and each QoS flow is marked by an ID (QFI, QoS Flow ID) q ($q \in 0, 1, 2$). In detail, $q = 0$ represents low latency eMBB applications, $q = 1$ represents live streaming, and $q = 2$ denotes buffered streaming.

B. QoS FLOW MAPPING AND PACKETS SCHEDULING

The association of data packets to QoS flows can be identified by QFI, priority level and other QoS characteristics. Therefore, when a packet is mapped into a QoS flow, the scheduler only focuses on its QoS characteristics. The packet scheduler doesn't care which application the packet comes from and which UE to deliver. In other words, the packets from multiple applications can be mapped to the same QoS flow. Similarly, the packets in the same QoS flow can be delivered to different UEs. In brief, the packets with the same or similar QoS characteristics will be mapped to the same QoS flow. Formally, a packet in the q^{th} QoS flow is denoted by p_q^t, where t means that the gNB receives the packet at time slot t.

The packet scheduling mechanism is depicted in Fig. 2. For each QoS flow q, there is a buffer B_q in the gNB. Packet

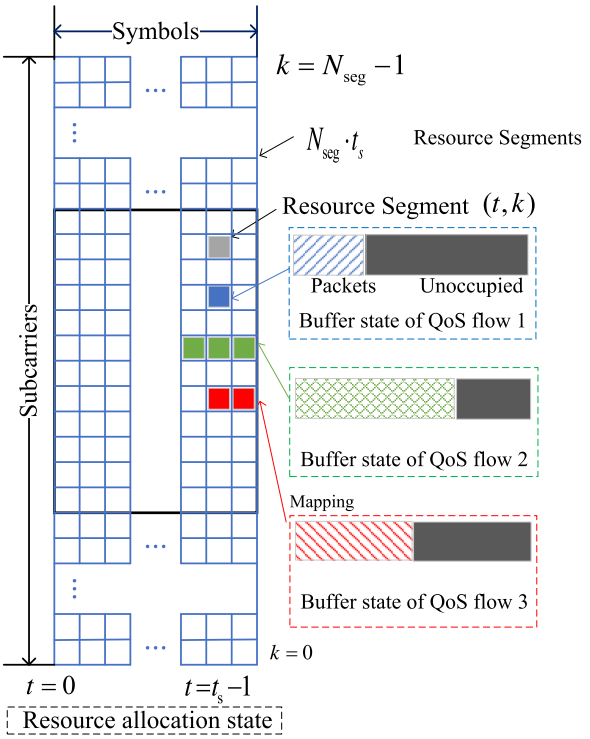


FIGURE 2. RAN resource allocation.

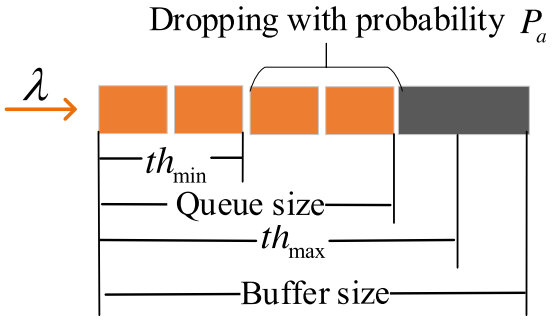


FIGURE 3. RED packet dropping scheme.

processing and delivery in each buffer are independent. For each buffer B_q, if the transmitting bitrate (i.e., service rate) μ_q is lower than the arriving bitrate λ_q, the untreated data packets will accumulate in the buffer, which leads to extra delay. Each packet is marked with a delay tolerance threshold, which can be considered as one of its QoS characteristics. If the packet delivery delay is greater than the threshold, it will be deemed as timeout loss by the UE. When $\mu_q < \lambda_q$, packet congestion occurs and a packet dropping scheme will be adopted. To be more specific, if the untreated packets in the buffer are too many to be delivered in time, some of those packets should be dropped to ensure other packets can be served timely. Otherwise, most of the untreated packets will be lost due to severe congestion.

To reduce packet loss when packet congestion occurs, random early detection (RED) algorithm [31] is employed as a packet dropping scheme to improve average packet delay. As shown in Fig. 3, minimum threshold th_{min} and maximum threshold th_{max} is designed in [31] to control congestion. For each buffer B_q, RED gateway calculates the packet queue length L_q first. Then, the packet queue length is compared with th_{min} and th_{max}. When L_q is less than th_{min}, no packets will be dropped. When L_q is greater than th_{max}, every arriving packet is dropped. When L_q is between th_{min} and th_{max}, each arriving packets are dropped with probability P_a, where P_a is a function of L_q, th_{min} and th_{max}. The packets dropping processing of each buffer is independent, and the parameters th_{max}, th_{min} and P_a of RED schemes for three buffers are the same.

C. TRAFFIC MODEL

For various multimedia applications, the bitrates may be different. For instance, the bitrate of 4K ultra high definition video streaming is much higher than that of standard definition video streaming. In brief, different QoS characteristics of multimedia applications will lead to different packets arriving bitrate. In general, average packet arriving bitrate of low latency eMBB flow (λ_0) is higher than that of the other two flows (λ_1 and λ_2). For QoS flow q, the number of arriving packets during each interval N_q^t follows a Poisson distribution with expected value $\lambda_q = \nu_q \cdot \lambda$, where ν_q is a bitrate factor. The factor ν_q can be considered as a bitrate characteristic of QoS flow q. To simulate the fluctuations in network traffic, λ follows a uniform distribution ranging from lower value λ_{min} to value λ_{max}.

Due to the discreteness of the radio resource in 5G networks, the RAN resource in the gNB is sliced into N_{seg} small segments at time t, and the time interval between each resource scheduling event is denoted as $t_{interval}$. Let $\beta_{q,k}^t \in \{0, 1\}$ denote the association between the k^{th} resource segment and the q^{th} QoS flow at time t. When $\beta_{q,k}^t = 1$, it means that the k^{th} resource segment is allocated to QoS flow q. Then, the service rate μ_q of QoS flow q at time t can be obtained by

$$\mu_q^t = \sum_k (\beta_{q,k}^t \cdot E/s_p), \tag{1}$$

where E is the size of each resource segment, and s_p denotes the packet size.

Hence, the packet queue length at time t can be represented as

$$L_q^t = \begin{cases} L_q^{t-1} + \lambda_q^t - \mu_q^t, & L_q^{t-1} < th_{min}; \\ L_q^{t-1} + \lambda_q^t \cdot (1 - P_a) - \mu_q^t, & th_{min} \le L_q^{t-1} \le th_{max}; \\ L_q^{t-1} - \mu_q^t, & th_{max} < L_q^{t-1}. \end{cases} \tag{2}$$

D. DELAY MODEL

In this paper, the packet delay means that the total transmission time of a packet from the MEC server to UEs. The packet delay can be defined as

$$d_{pkt} = d_{prp} + d_{prc} + d_{radio}, \tag{3}$$

where d_{prp} is the propagation delay of signals in optical fiber, d_{prc} denotes the processing delay in the gNB, and d_{radio} represents the radio delay from remote radio head to UEs. d_{prp} can be calculated by l/c, where l is the distance between MEC server and the gNB, and c is the propagation velocity of signal in the fiber. For a packet $p_q^{t,n}(n \in \{0, 1, \ldots, N_q^t - 1\})$, the packet queue length at time t can be obtained by (2). Then, the processing delay of the packet can be calculated by

$$d_{prc}(p_q^{t,n}) = \begin{cases} \sigma_q^t, & p_q^{t,n} \text{ is served;} \\ \infty, & p_q^{t,n} \text{ is dropped,} \end{cases} \tag{4}$$

where σ_q^t can be obtained by

$$\sigma_q^t = \arg\min_{\sigma}(L_q^t - \mu_q^{t+1} \ldots - \mu_q^{t+\sigma})^2. \tag{5}$$

It should be noticed that the processing delay will be infinite, if the packet is dropped. Substituting (4) for d_{prc} in (3), it can be rewritten as

$$d_{pkt}(p_q^{t,n}) = \begin{cases} d_{prp} + \sigma_q^t + d_{radio}, & p_q^{t,n} \text{ is served;} \\ \infty, & p_q^{t,n} \text{ is dropped.} \end{cases} \tag{6}$$

E. PROBLEM FORMULATION

In order to quantify the performance of multimedia multi-service, a QoS evaluation function is designed as

$$R_q^{t,n} = \begin{cases} \dfrac{\tau_q - d_{pkt}(p_q^{t,n})}{\tau_q}, & d_{pkt}(p_q^{t,n}) < \tau_q; \\ 0, & d_{pkt}(p_q^{t,n}) \geq \tau_q, \end{cases} \tag{7}$$

where τ_q is the delay tolerance threshold of QoS flow q. When a packet stays in the buffer too long to be served, the packet will be dropped and the evaluation value is set to 0. To satisfy the diverse requirements of QoS flows, the resource scheduler needs to map the RAN resource segments to the QoS flows properly. Mathematically, a QoS value maximization problem can be formulated as follows.

$$\begin{aligned} &\max \sum_t \sum_q \sum_n R_q^{t,n}, \\ &s.t. \left\{ \sum_q \sum_k \beta_{q,k}^t \leq N_{seg}, \right. \end{aligned} \tag{8}$$

where the constraint means that the resource segments which can be allocated to QoS flows are limited. During each time slot t, there are only N_{seg} resource segments.

IV. DEEP-Q-NETWORK BASED OPTIMIZATION FRAMEWORK

To address the complex problem proposed in section III.E, DQN framework is adopted to look for resource allocation policies. Firstly, this section gives a brief introduction over DQN framework. Then, the key elements of DQN framework is designed. Finally, the DQN based resource allocation algorithm is proposed.

A. DEEP-Q-NETWORK

DQN algorithm is considered as a deep reinforcement learning (DRL) algorithm, which focuses on how to interact with the environment to attain maximum cumulative reward. The DQN algorithm is evolved from Q-learning algorithm by applying neural network (NN). Q-learning is proposed as a RL algorithm in [32]. In general, Q-learning is regarded as a temporal-difference update, model-free, off-policy RL algorithm for discrete state spaces.

In Q-learning, when RL agent chooses an action a^i under state s^i at iteration step i, a reward $R(s^i, a^i)$ will be produced from the environment. The Q-value $Q(s^i, a^i)$ is an estimate of expected long-term reward for a state-action pair (s^i, a^i). All the Q-values are stored in a table. Based on the Q-value table, the agent can decide the next action according to a sampling policy, generally ϵ-greedy policy. If ϵ-greedy policy is adopted, the agent chooses an action with the best Q-value with a probability of ϵ, or a random action with a probability of $1 - \epsilon$. After the agent learns the reward $r(a^i, s^i)$, the state will transfer to the next state s^{i+1}. Finally, the Q-value table is updated in a temporal-difference manner. The update policy is given as

$$\begin{aligned} Q(s^i, a^i) \leftarrow\ & Q(s^i, a^i) \\ & + \alpha(R(s^i, a^i) + \gamma \max_{a^{i+1}} Q(s^{i+1}, a^{i+1}) - Q(s^i, a^i)), \end{aligned} \tag{9}$$

where α is learning rate and γ denotes reward decay factor.

Note that the size of Q-value table will increase exponentially, as the state space and action space expand. For large state space and action space, it not only occupies vast storage resource and also increase convergence time. To improve the performance of Q-learning, DQN can approximate Q-values by employing NN. The Q-function in DQN is denoted as $Q(s^i, a^i, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the weights of NN. As shown in Fig. 4, experience replay and target network is applied in DQN to improve learning efficiency and accelerate convergence.

- **Experience Replay**. The agent records state, action, reward and next state as a transition $z^i = [s^i, a^i, R^i(s^i, a^i), s^{i+1}]$. The transitions are stored in a memory buffer. Then, the agent selects some transitions as a mini-batch from the memory buffer to update the weights of NN.
- **Target Network**. A separate target NN Q^* is used to make the algorithm more stable. The agent clones the evaluation NN Q to obtain the target NN $\widehat{Q}$ every C steps, and uses Q^* to generate target values to calculate the loss $L(\boldsymbol{\theta})$. The loss $L(\boldsymbol{\theta})$ is utilized to update the weights of the evaluation NN Q.

In the nature DQN algorithm, the evaluation NN can be trained by function approximation. In other wards, Q-value $Q(s, a, \boldsymbol{\theta})$ should get closer and closer to the target value $Q^*(s, a, \bar{\boldsymbol{\theta}})$ by learning from enough iterations. The target value can be obtained by

$$Q^*(s, a, \bar{\boldsymbol{\theta}}) = R(s, a) + \gamma \max_{a'} Q^*(s', a', \bar{\boldsymbol{\theta}}). \tag{10}$$

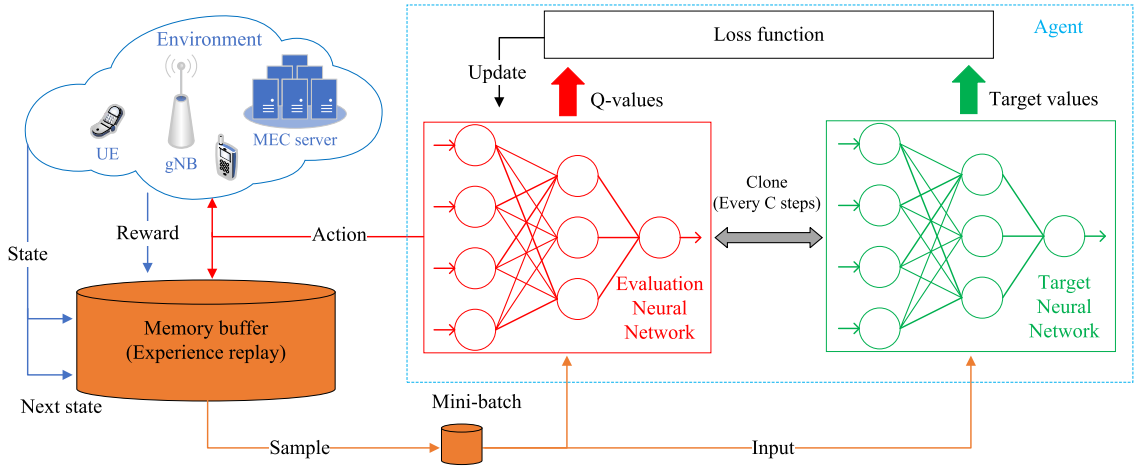


FIGURE 4. An illustration of Deep-Q-Network.

Then, the loss function is defined as

$$L(\boldsymbol{\theta}) = (Q^*(s, a, \bar{\boldsymbol{\theta}}) - Q(s, a, \boldsymbol{\theta}))^2. \tag{11}$$

By minimizing $L(\boldsymbol{\theta})$, the NN weights can be obtained by a gradient descent approach.

B. DRL FRAMEWORK DESIGN

Generally, DQN framework can be modeled as a Markov decision process (MDP) [33], which is a typical model for decision making problems. The aim of MDP is to find a policy to choose a proper action a under state s at each iteration i, so that the agent can achieve the best accumulated reward.

The DRL framework consists of four key elements, which are state space S, action space A, observation of each state $o(s)$ and reward function $R(s, a)$.

- **State space**. The system state s^i represents the system features including the waiting time of the packets in the buffers, the association between packets and QoS flows, the buffer states and so forth.
- **Action space**. In order to maximize the QoS value, the agent decides how to allocate the k^{th} resource segment at time t. Let $A^i = \{0, 1, 2\}$ denotes all the available actions which can be selected by the agent at step i. It is noticed that there are three available actions at each step, which corresponds to three QoS flows in the gNB. Specifically, action $a^i = \upsilon(\upsilon \in \{0, 1, 2\})$ means that the k^{th} segment at time t is allocated to the υ^{th} QoS flow.
- **Observation**. The observations are samples of the system states for the agent to make decisions. To reduce the complexity of NN, the agent just has to focus on two vectors, $\mathbf{X}^i$ and $\mathbf{D}^i$, at each step. The observation at step i is represented as

$$\begin{aligned} o^i(s^i) &= [\mathbf{X}^i, \mathbf{D}^i] \\ &= [x_0^i, x_1^i, x_2^i, d_0^i, d_1^i, d_2^i], \end{aligned} \tag{12}$$

  where x_q^i is the number of untreated packets in the buffer B_q, and d_q^i is the packet delay of the first untreated packet in the buffer B_q. In brief, observation $o^i(s^i)$ collects the information of each QoS flow, which includes how many packets are waiting in each buffer and how long the unprocessed packets have waited.
- **Reward function**. To optimize the overall QoS performance of multimedia multi-service, a QoS evaluation function is proposed in (7), which can evaluate the packet delay and packet loss at the same time. When the agent chooses an action a^i under state s^i, it will receive a reward from the environment. According to the reward, the agent learns to how to allocate the k^{th} segment at each iteration, so that the transmitting bandwidth will be adjusted dynamically. The reward for the action a^i under the state s^i can be defined as

$$R^i(s^i, a^i) = \eta \cdot \sum_q R_q^i + \sum_q \mu_q^i \cdot s_p/E, \tag{13}$$

  where η is a weighting coefficient for the QoS evaluation value, $\sum_q \mu_q^i \cdot s_p/E$ is resource utilization ratio. The resource utilization ratio in the hybrid reward can achieve a fast convergence.

Noted that the index t indicates the serial number of time slot, and the index i denotes the serial number of iteration step. The relationship between t and i is given by

$$i = t \cdot N_{seg} + k, \tag{14}$$

where k is the serial number of RAN resource segments.

C. DEEP-Q-NETWORK BASED RESOURCE ALLOCATION ALGORITHM

The DQN based resource allocation algorithm is described in Algorithm 1. The DRL agent can collect the information including state, action, reward and be trained in the background. After enough iterations, the learnt NN are stored. Then, the agent makes decisions to allocate the proper resource segments to QoS flows dynamically.

Algorithm 1 Deep-Q-Network Based Resource Allocation Algorithm

Initialize an evaluation NN Q with weights $\boldsymbol{\theta}$, and a target NN Q^* with weights $\bar{\boldsymbol{\theta}}$, where $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}$.
Initialize a replay memory buffer.
for episode $ep = 0$ to $N_{ep} - 1$ **do**
  for iteration $i = 0$ to $t_{total} \cdot N_{segment} - 1$ **do**
    Receive the incoming packets.
    Obtain the observation $o^i(s^i)$ of system state s^i.
    Choose an action satisfying $a^i = \arg\max_{a^i} Q^i(s^i, a^i, \boldsymbol{\theta})$ with probability ϵ, or randomly select an action from action space A.
    Execute the selected action, and observe the reward $R^i(s^i, a^i)$.
    Store the transition $z^i = [o^i(s^i), a^i, R^i(s^i, a^i), o^{i+1}(s^{i+1})]$ in the memory buffer.
    Samples random mini batch of transitions from the memory buffer as the input of NN.
    if episode terminates at step i **then**
      Set $y^i = R(s^i, a^i) + \gamma \max_{\bar{a}^i} Q^*(\bar{s}^i, \bar{a}^i, \bar{\boldsymbol{\theta}})$;
    else
      Set $y^i = R(s^i, a^i)$.
    end if
    Calculate the loss $L^i(\boldsymbol{\theta}) = (y^i - Q^i(s^i, a^i, \boldsymbol{\theta}))^2$.
    Update the weights $\boldsymbol{\theta}$ of evaluation NN by minimizing the loss $L^i(\boldsymbol{\theta})$ via a gradient decent algorithm.
    if $i \,\%\, C == 0$ **then**
      Update the target NN by resetting weights $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}$.
    end if
  end for
end for

V. SIMULATION RESULTS

A. SCENARIO CONFIGURATION

To evaluate the performance of the proposed DQN based resource allocation algorithm, simulations are performed on the Python platform. The simulation parameters are depicted in Table 1.

This paper compares the simulation results with the following two resource allocation algorithms, so as to validate the advantages of the proposed algorithm.

- **Round-robin (RR)** A RR scheduler just allocates the resource segments in turn to QoS flows. In other words, a QoS flow can't be served in two successive scheduling periods. Therefore, each QoS flow can be mapped to one third of the total RAN resource.
- **Priority-based (PB)** As mentioned above, each QoS flow has distinct QoS characteristics. As one of the QoS characteristics, priority level is configured when the packets are mapped into QoS flows [29]. Priority level indicates the importance of a resource request. More exactly, the QoS flow with higher priority level shall be processed first. When all QoS requirements has been fulfilled for the QoS flow with higher priority level, the QoS flow with lower priority level will be processed. According to [29], the priority order is set to $q_0 > q_1 > q_2$.

TABLE 1. Simulation parameters.

Parameter	Value
Number of QoS flows	3
Packet size (S_p)	1024 bytes
Size of single segment unit (E)	61440 bit
RAN resource	300 Mbps
Number of segment (N_{seg})	5
Delay tolerance thresholds (τ_0, τ_1, τ_2)	10,100,300 ms
Priority level	$q_0 > q_1 > q_2$
Radio delay d_{radio}	1 ms
Transmission distance (l)	100 km
Propagation velocity in the fiber (c)	$2 \times 10^8 m/s$
Bitrate factor (ν_0, ν_1, ν_2)	3,1,2
Minimum Poisson factor (λ_{min})	2
Maximum Poisson factor (λ_{max})	7
Number of time slots per episode (t_{total})	3000
Number of episodes (N_{ep})	150
Interval time unit $(t_{interval})$	1 ms
QoS reward weighting factor (η)	0.2
RED threshold (th_{min}, th_{max})	15,30 Mbit
Maximum packet drop ratio (P_a)	0.015
Learning rate (α)	0.00001
Reward decay(γ)	0.9
Maximum value of (ϵ)	0.9
Size of memory buffer	80000
Size of mini batch	32
Memory replay period (C)	4 iteration steps
Number of hidden layers	2
Number of nodes per layer	90

B. CONVERGENCE ANALYSIS

Fig. 5-7 display the convergence curves of the proposed DQN based algorithm for loss $L(\boldsymbol{\theta})$, reward $R(s, a)$ and resource utilization ratio, respectively.

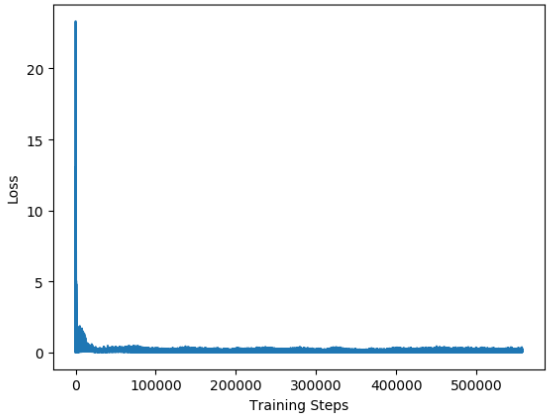


FIGURE 5. Loss between target values and Q-values.

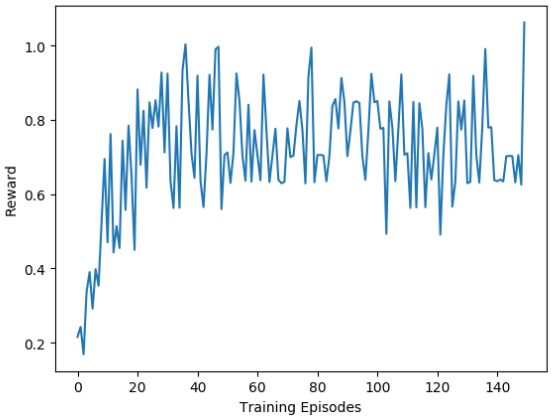


FIGURE 6. Average reward for varying episodes.

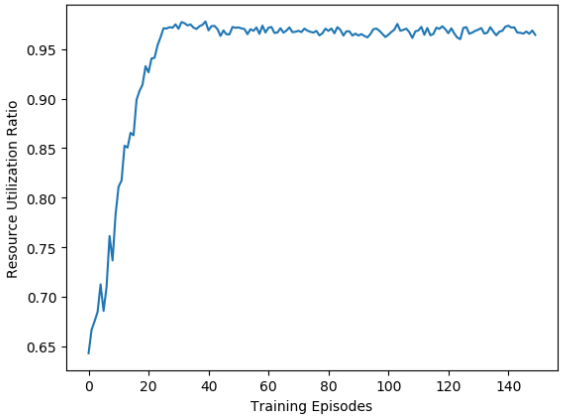


FIGURE 7. Average resource utilization ratio for varying episodes.

As shown in Fig. 5, the loss between the target values from target NN and the Q-values from evaluation NN declines sharply, as the training steps increase. The loss converges to a stable state, when the agent has been trained enough steps. Fig. 6 illustrates the average reward of iterations in each episode. the reward gradually increases and starts to fluctuate when the number of episodes N_{ep} is about 30. It is seen from Fig. 7 that the average resource utilization ratio raises rapidly at the beginning. When N_{ep} is about 30, the average resource utilization ratio in each episode is always better than 95%. In a word, the convergence curves prove that the agent has trained well, and is able to work as a resource scheduler in the gNB.

It is noticed that there are some fluctuations in Fig. 6 and Fig. 7, although the loss curve has leveled off. This is because the max value of greedy factor ϵ in training phase is set to 0.9, which means that the agent may choose a suboptimal or even bad action with a probability of 10% at each iteration.

C. PERFORMANCE ANALYSIS

In comparison of RR and PB resource allocation algorithms, the performance of the proposed algorithm is validated in terms of average QoS, average packet delay, average packet loss ratio and throughput. For each packet, its QoS value can be obtained by (7), and thus the average QoS for QoS flow q is defined as

$$R_{av,q} = \sum_t \sum_n R_q^{t,n} / \sum_t N_q^t. \tag{15}$$

Besides, for each packet, its packet delay can be obtained by (6). It is noticed that this paper only calculates the delay of the packets which have been processed. Then, the average packet delays of the processed packets in QoS flow q is defined as

$$d_{av,q} = \sum_t \sum_{n_{serv}} d_{pkt} / \sum_t N_q^t, \tag{16}$$

where $n_{serv} \in [0, N_{serv,q}^t - 1]$ and $N_{serv,q}^t$ is the number of the processed packets in QoS flow q during slot t. The average packet loss ratio is given by

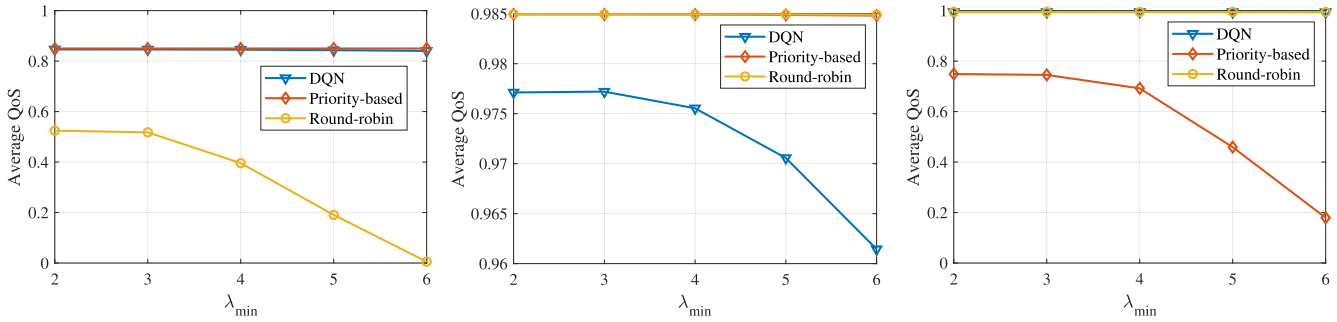
$$\zeta_{loss} = \sum_q \sum_t (N_{drop,q}^t + N_{timeout,q}^t) / \sum_t N_q^t, \tag{17}$$

where $N_{drop,q}^t$ is the number of packets which are dropped by RED scheme in the QoS flow q at time t, and $N_{timeout,q}^t$ is the number of packets which are deemed as timeout by UEs in the QoS flow q at time t. Furthermore, throughput can be calculated by

$$B = \frac{\sum_q \sum_t (N_q^t - N_{drop,q}^t - N_{timeout,q}^t) \cdot s_p}{t_{total} \cdot t_{interval}}. \tag{18}$$

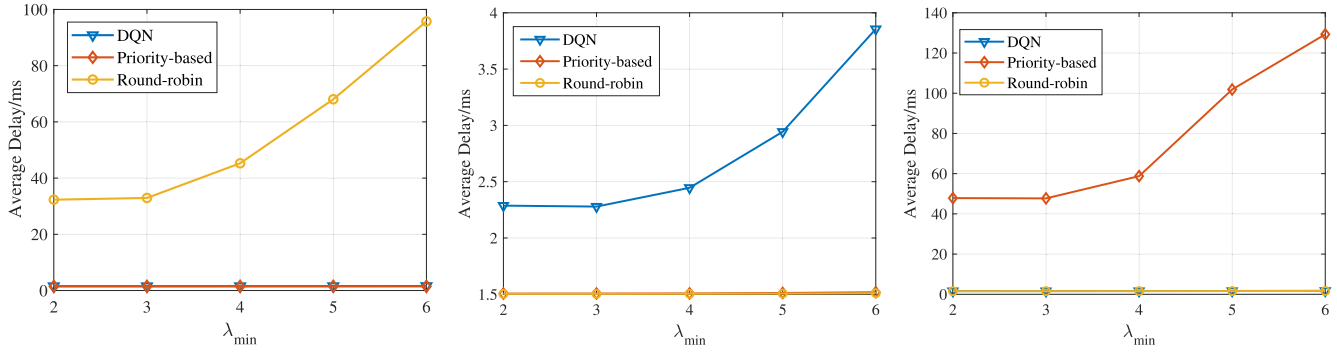
In the simulations, the minimum Poisson factor λ_{min} varies in a range [2, 6]. As mentioned in III.C, the arriving multimedia packets in each QoS flow q at each time slot t follows a Poisson distribution with expected value $\lambda_q = \nu_q \cdot \lambda$. Then, the factor λ will be selected from the set $[\lambda_{min}, \lambda_{max}]$ randomly. Therefore, as λ_{min} increases, the average packet arrival bitrate raises and the network congestion becomes more and more serious.

Fig. 8 illustrates the average QoS of each resource allocation schemes for data packets in different QoS flow. As for the packets from low-latency eMBB applications, the average QoS of RR scheduler declines gradually. When $\lambda_{min} = 6$, the average QoS is almost down to 0. In contrast, the average



(a) Low-latency eMBB applications. (b) Live streaming. (c) Buffered streaming.

FIGURE 8. Average QoS values for packets in different QoS flows.



(a) Low-latency eMBB applications. (b) Live streaming. (c) Buffered streaming.

FIGURE 9. Average delays for the packets which have been served in different QoS flows.

QoS of PB and DQN based schedulers are always greater than 0.8, although λ_{min} keeps growing. As for the packets from living streaming, the average QoS of RR scheduler and PB scheduler is always about 0.985. As the increase of λ_{min}, the average QoS of DQN based scheduler decreases slightly. Although $\lambda_{min} = 6$, its average QoS is still higher than 0.96. As for the packets from buffered streaming, the average QoS of RR and DQN based schedulers is nearly unchanged, when λ_{min} varies. As network congestion becomes severe, the average QoS of PB scheduler falls.

Fig. 9 shows the average packet delays of the packets which have been served for different QoS flows. For low-latency eMBB applications, only the average packet delay of RR scheduler soars to 95.7 ms, when $\lambda_{min} = 6$. Although $\lambda_{min} = 2$, its average packet delay exceeds the delay tolerance threshold ($\tau_0 = 10$ ms). For living streaming, only the average packet delay of DQN based scheduler rises slightly, as the increase of λ_{min}. Even though $\lambda_{min} = 6$, its average packet delay is lower than 4 ms, which is much less than the delay tolerance threshold($\tau_1 = 100$ ms). For buffered streaming, only the average packet delay of PB scheduler worsens, as λ_{min} ascends. When $\lambda_{min} = 6$, its average packet delay reaches about 130 ms, which is far greater than that of the others.

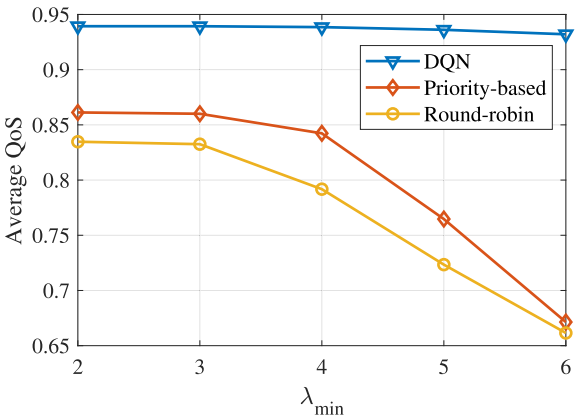


FIGURE 10. Average QoS values for all packets in the gNB.

The overall performance of the proposed DQN based algorithm for multi-service scenario is also evaluated. Fig. 10 gives the average QoS for all the packets which have been served by the gNB. According to the figure, the average QoS of DQN based scheduler keeps higher than 0.9. In contrast, the average QoS values of RR and PB schedulers drop gradually, as λ_{min} increases.

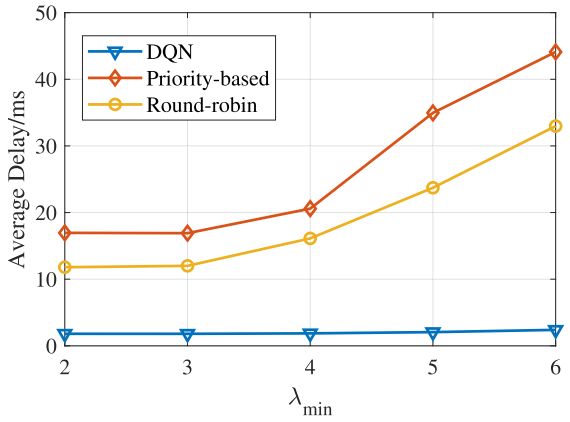


FIGURE 11. Average delays for all the packets which have been served in the gNB.

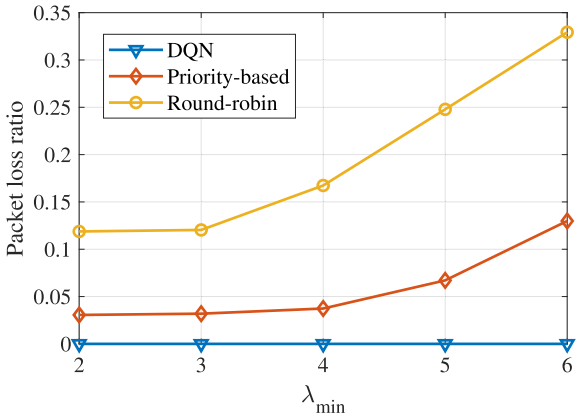


FIGURE 12. Average packet loss ratios for different resource allocation algorithms.

Fig. 11 shows the average delay of the packets which have been served in all three QoS flows. When $\lambda_{min} = 6$, the average packet delay of PB scheduler is up to 44.1 ms, and that of RR scheduler is about 33.0 ms. Unlike RR and PB schedulers, the average packet delay of DQN based scheduler increases slowly, and it is still lower than 2.5 ms when $\lambda_{min} = 6$.

The average packet loss ratios are depicted in Fig. 12. It can be seen that the average packet loss ratio of DQN based scheduler is always 0. Besides, the average packet loss ratios of RR and PB schedulers go up, as λ_{min} increases. When $\lambda_{min} = 6$, the average packet loss ratio of PB scheduler is up to 0.130, and that of PB scheduler is about 0.329.

Finally, the throughput curves are exhibited in Fig. 13. As the λ_{min} rises, the throughput of DQN based scheduler ascends from 200 Mbps to nearly 300 Mbps, and that of PB goes approximately from 193 Mbps to 253 Mbps. It is noticed that the throughput of PB grows more slowly, when $\lambda_{min} \geq 5$. As for RR scheduler, the throughput raises when $\lambda_{min} \leq 4$, and drops when $\lambda_{min} \geq 4$. The maximum throughput of RR is lower than 184 Mbps, when $\lambda_{min} = 4$.

However, the following conclusions can be drawn from the figures.

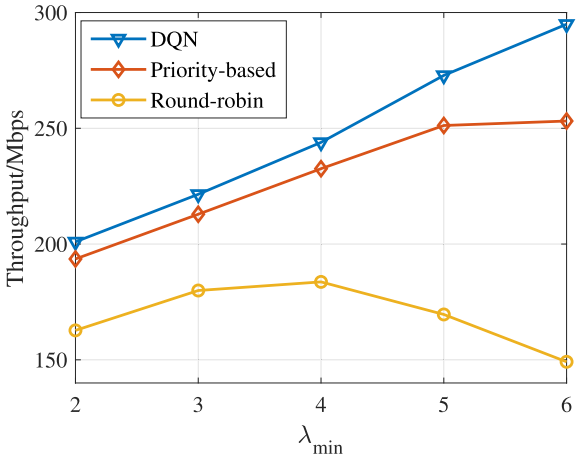


FIGURE 13. Throughputs for different resource allocation algorithms.

1) With slight degradation in the performance of live streaming, the proposed DQN based algorithm ensures the performance of the other two QoS flows.
2) In PB scheduling, the performance of buffered streaming is sacrificed to keep low-latency eMBB applications being served in time. This is because the priority level of low-latency eMBB applications is higher than that of the others.
3) In RR scheduling, low-latency eMBB application can't be served timely, especially when arriving bitrate is high. This is because the resource are allocated equally and its delay tolerance are lower than that of the others. Moreover, its arriving packets number is much more than that of the other QoS flows.
4) The proposed DQN based algorithm outperforms the others in terms of average QoS, average packet delay, average packet loss ratio and throughput.

VI. CONCLUSION AND FUTURE WORKS

This paper considers a multimedia multi-service scenario, where the MEC server can push various multimedia applications to edge users. Besides, the 5G QoS model is investigated, in which packets from different multimedia applications can be mapped into different QoS flows, according to their QoS requirements. However, matching the allocated resource to QoS flows with diverse QoS characteristics effectively will be the most critical challenge to the scheduler. To deal with the challenge, a QoS evaluation model is designed to formulate a QoS maximization problem. Inspired by the success of DRL technique in solving complex problems, a DQN framework is applied to allocate resource flexibly. Furthermore, the advantage of DQN framework in managing QoS-aware resource allocation is validated in comparison of RR scheduling and PB scheduling. Simulation results show that the DQN based algorithm achieves better performance than other resource allocation algorithms.

In this paper, three typical services in [29] are selected. Those services are characterized by different delay tolerances

and bitrates. By retraining the DQN with new parameters, the proposed resource allocation algorithm can still work for other multimedia services. Furthermore, the radio delay is set to 1 ms for simplification of the wireless channel model. The researches on channel-state-aware scheduling are left for future works.

REFERENCES

[1] "Cisco visual networking index: Forecast and trends, 2017–2022," Cisco, San Jose, CA, USA, White Paper c11-741490, Feb. 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] D. Wu, Z. Zhang, S. Wu, J. Yang, and R. Wang, "Biologically inspired resource allocation for network slices in 5G-enabled Internet of Things," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2888543.

[3] P. Zhang, X. Kang, X. Li, Y. Liu, D. Wu, and R. Wang, "Overlapping community deep exploring-based relay selection method toward multi-hop D2D communication," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1357–1360, Oct. 2019.

[4] Z. Zhang and L. Wang, "Social tie-driven content priority scheme for D2D communications," *Inf. Sci.*, vol. 480, pp. 160–173, Apr. 2019.

[5] Z. Li, J. Chen, and Z. Zhang, "Socially aware caching in D2D enabled fog radio access networks," *IEEE Access*, vol. 7, pp. 84293–84303, 2019.

[6] D. Wu, B. Liu, Q. Yang, and R. Wang, "Social-aware cooperative caching mechanism in mobile social networks," *J. Netw. Comput. Appl.*, vol. 149, Jan. 2020, Art. no. 102457, doi: 10.1016/j.jnca.2019.102457.

[7] P. Zhang, X. Kang, D. Wu, and R. Wang, "High-accuracy entity state prediction method based on deep belief network toward IoT search," *IEEE Wireless Commun. Lett.*, vol. 8, no. 2, pp. 492–495, Apr. 2019.

[8] Z. Li, Y. Jiang, Y. Gao, D. Yang, and L. Sang, "On buffer-constrained throughput of a wireless-powered communication system," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 2, pp. 283–297, Feb. 2019.

[9] D. Wu, H. Shi, H. Wang, R. Wang, and H. Fang, "A feature-based learning system for Internet of Things applications," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1928–1937, Apr. 2019.

[10] D. Wu, Q. Liu, H. Wang, Q. Yang, and R. Wang, "Cache less for more: Exploiting cooperative video caching and delivery in D2D communications," *IEEE Trans. Multimedia*, vol. 21, no. 7, pp. 1788–1798, Jul. 2019.

[11] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper 11, 2015, pp. 1–16, vol. 11, no. 11.

[12] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[13] Z. Ke, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 39–45, May 2018.

[14] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[15] V. Balasubramanian, M. Wang, M. Reisslein, and C. Xu, "Edge-boost: Enhancing multimedia delivery with mobile edge caching in 5G-D2D networks," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2019, pp. 1684–1689.

[16] R. Viola, A. Martin, M. Zorrilla, and J. Montalbán, "MEC proxy for efficient cache and reliable multi-CDN video distribution," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2018, pp. 1–7.

[17] X. Chen, L. He, S. Xu, S. Hu, Q. Li, and G. Liu, "Hit ratio driven mobile edge caching scheme for video on demand services," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2019, pp. 1702–1707.

[18] A. Mehrabi, M. Siekkinen, G. Illahi, and A. Ylä-Jääski, "D2D-enabled collaborative edge caching and processing with adaptive mobile video streaming," in *Proc. IEEE 20th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2019, pp. 1–10.

[19] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 116–122, Jun. 2018.

[20] J. Martín-Pérez, L. Cominardi, C. J. Bernardos, A. de la Oliva, and A. Azcorra, "Modeling mobile edge computing deployments for low latency multimedia services," *IEEE Trans. Broadcast.*, vol. 65, no. 2, pp. 464–474, Jun. 2019.

[21] N. Yu, Q. Xie, Q. Wang, H. Du, H. Huang, and X. Jia, "Collaborative service placement for mobile edge computing applications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[22] J. Ren, Y. Ruan, and G. Yu, "Data transmission in mobile edge networks: Whether and where to compress?" *IEEE Commun. Lett.*, vol. 23, no. 3, pp. 490–493, Mar. 2019.

[23] A. Younis, T. X. Tran, and D. Pompili, "On-demand video-streaming quality of experience maximization in mobile edge computing," in *Proc. IEEE 20th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2019, pp. 1–9.

[24] K. Koslowski, R. Santos, W. Keusgen, T. Haustein, A. Kassler, K. Sakaguchi, H. Ogawa, M. Nakamura, and Y. Tao, "SDN orchestration to optimize meshed millimeter-wave backhaul networks for MEC-enhanced eMBB use cases," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2018, pp. 1–5.

[25] L. Van Ma, V. Q. Nguyen, J. Park, and J. Kim, "NFV-based mobile edge computing for lowering latency of 4K video streaming," in *Proc. 10th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2018, pp. 670–673.

[26] M. A. Rahman, M. S. Hossain, G. Loukas, E. Hassanain, S. S. Rahman, M. F. Alhamid, and M. Guizani, "Blockchain-based mobile edge computing framework for secure therapy applications," *IEEE Access*, vol. 6, pp. 72469–72478, 2018.

[27] H. Hu, H. Shan, Z. Zheng, Z. Huang, C. Cai, C. Wang, X. Zhen, L. Yu, Z. Zhang, and T. Q. S. Quek, "Intelligent video surveillance based on mobile edge networks," in *Proc. IEEE Int. Conf. Commun. Syst. (ICCS)*, Dec. 2018, pp. 286–291.

[28] H. Trinh, "Energy-aware mobile edge computing and routing for low-latency visual data processing," *IEEE Trans. Multimedia*, vol. 20, no. 10, pp. 2562–2577, Oct. 2018.

[29] *System Architecture for the 5G System*, document TS 23.501, 3GPP, Jun. 2019.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[31] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[32] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, May 1992, doi: 10.1007/BF00992698.

[33] R. Bellman, "A Markovian decision process," *J. Math. Mech.*, vol. 6, no. 5, pp. 679–684, 1957.

BOREN GUO received the B.Eng. degree in electronics and communications engineering from the Beijing University of Posts and Telecommunications (BUPT), in 2015, where he is currently pursuing the Ph.D. degree with the Wireless Theories and Technologies Laboratory. His research interests include C-RAN, F-RAN, MEC, deep reinforcement learning, and other 5G NR techniques.

XIN ZHANG received the B.Eng. degree in communications engineering, the M.Eng. degree in signal and information processing, and the Ph.D. degree in communications and information systems from the Beijing University of Posts and Telecommunications (BUPT), in 1997, 2000, and 2003, respectively. He joined BUPT, in 2003, where he is currently an Associate Professor with the Wireless Theories and Technologies Laboratory. His research interest includes key technologies and performance analysis of air interface of wireless networks.

YAXIN WANG received the B.Eng. degree in electronics and communications engineering and the M.Eng. and Ph.D. degrees in telecommunications engineering from the Beijing University of Posts and Telecommunications, in 2012, 2015, and 2019, respectively. His research interests include C-RAN, NFV, machine learning, and other 5G technologies.

HONGWEN YANG received the B.S. and M.S. degrees from the Beijing University of Posts and Telecommunications (BUPT), in 1984 and 1987, respectively. After graduating, he joined the faculty of BUPT, where he is currently a Professor. His research interests include wireless physical layer, including modulation and coding, MIMO, and OFDM.