Hindawi
Mobile Information Systems
Volume 2020, Article ID 8825643, 11 pages
https://doi.org/10.1155/2020/8825643

Research Article

DRL-Based Edge Computing Model to Offload the FIFA World Cup Traffic

Hongyi Li[1] and Xinrui Che[2]

[1]*Dalian Maritime University, Dalian 116026, China*
[2]*Liaoning Police College, Dalian 116036, China*

Correspondence should be addressed to Xinrui Che; chexinrui@lnpc.cn

Received 14 September 2020; Revised 22 October 2020; Accepted 1 November 2020; Published 19 November 2020

Academic Editor: Jianhui Lv

In recent years, the volume of global video traffic has been increasing rapidly and it is considerably significant to offload the traffic during the process of video transmission and improve the experience of users. In this paper, we propose a novel traffic offloading strategy to provide a feasible and efficient reference for the following 2022 FIFA World Cup held in Qatar. At first, we present the system framework based on the Mobile Edge Computing (MEC) paradigm, which supports transferring the FIFA World Cup traffic to the mobile edge servers. Then, the Deep Reinforcement Learning (DRL) is used to provide the traffic scheduling method and minimize the scheduling time of application programs. Meanwhile, the task scheduling operation is regarded as the process of Markov decision, and the proximal policy optimization method is used to train the Deep Neural Network in the DRL. For the proposed traffic offloading strategy, we do the simulation based on two real datasets, and the experimental results show that it has smaller scheduling time, higher bandwidth utilization, and better experience of user than two baselines.

1. Introduction

The mobile Internet and Internet of Things (IoT) have been developed in recent years [1] especially during the process of building the smart cities [2], which has been generating the billions of Internet traffic due to the sharp increasing number of mobile devices (such as smart phone and wearable monitors). For example, the Cisco Annual Internet Report (CAIR) [3] shows that the total number of global mobile devices will grow from 5.1 billion (66 percent of population) in 2018 to 5.7 billion (71 percent of population) by 2023. In addition, the global IP traffic is expected to reach 5.3 ZB by 2023 [4] and the video traffic will account for 67.5% of the global IP traffic due to the introduction of new techniques and applications such as 4K/8K [5] and AR/VR [6] which are the necessary products in the smart cities. As we know, there are usually three kinds of video transmission, i.e., video on demand, carousel, and live streaming [7]. At the era of mobile Internet, more and more users pay attention to the applications of live streaming. Particularly, at the special time frame, the internationalized sport events which belong to the field of live streaming attract a large number of audiences without doubt. For example, the FIFA World Cup is the famous live sport event. Given this, this paper plans to investigate the FIFA World Cup and gives a network solution from the perspective of traffic offloading in order to provide support for the following 2022 FIFA World Cup in Qatar.

The traditional video delivery strategies usually depend on the technique of Content Delivery Networks (CDN) [8, 9]. In other words, the Internet content providers deliver the abundant contents to the edge users based on CDN in the push mode, in which the intermediate and the edge servers store the hop contents in advance. However, the strategies based on CDN are not suitable to the traffic offloading for the FIFA World Cup and the main reasons are analyzed as follows. The volume of the FIFA World Cup traffic is the unspeakably large and the corresponding features show the periodicity, abruptness, and explosivity. For the traffic, CDN fails to deliver them to the Metropolitan Area Network

(MAN) or the Access Network (AN) which is very close to the mobile end-users, due to the fact that CDN servers are very expensive and it is impossible for the Internet content providers to deploy many CDN servers at the MAN or AN. As an alternative solution on the traffic offloading of the FIFA World Cup, the edge computing [10] can support the closest contents caching at the edge servers to satisfy the requirements of users. Nevertheless, at the era of mobile Internet, the ability of edge computing cannot be handled with the billions of mobile devices. At the right time, the Mobile Edge Computing (MEC) [11–13] has been regarded as the relatively appropriate alternative solution.

Different from the centralized cloud computing mode, the computation resources and storage resources in MEC are deployed at the edge network (such as mobile base station, wireless hotspot, and edge router) in the distributed way. On this basis, the computation tasks on the FIFA World Cup traffic can be offloaded to the mobile edge servers for running, which greatly reduces the communication overhead and the network delay of application programs. At the same time, the pressure faced by Internet content providers as well as core networks can be relieved effectively. In fact, the network performance improvement in MEC strictly depends on the tasks offloading decision [14]. Furthermore, the decision problem usually involves some necessary factors, such as network bandwidth, timing sequence of application program, and dependence between tasks. As a result, the mobile application programs can usually be built as a Directed Acyclic Graph (DAG) model [15] to realize the fine-grained traffic scheduling and enable the parallel processing for the multiple tasks. However, the task scheduling based on DAG belongs to the NP-hard problem [16], which indicates that those heuristic and approximate traffic offloading strategies cannot satisfy the requirements of users (especially for the real-time requirements) during the process of watching the FIFA World Cup. Therefore, it is extremely urgent to find a stable and powerful method to solve such problem.

To the best of our knowledge, the Deep Reinforcement Learning (DRL) [17, 18] has attracted much attention from the global researchers in the field of Artificial Intelligence (AI), which integrates the advantages of Reinforcement Learning (RL) [19] and Deep Neural Networks (DNN) [20] and enables obtaining the optimal decision by automatically learning the network environment based on the multiple interactions. To be specific, the DRL has the following benefits. (1) DRL can learn the optimal decision strategy in the model-free way, and the whole process has no need for the environment modelling, reflecting the great flexibility. (2) DRL has the strong presentation ability and generalization ability of supporting the huge state space in terms of the DAG-based traffic scheduling problem. (3) DRL is a global optimization strategy and it has the large probability of obtaining the optimal solution. Although there have been some proposals on using DRL to address the traffic offloading problem in MEC, they cannot relieve the network pressure well and cannot be used for the traffic offloading in the FIFA World Cup directly. With the above considerations, this paper plans to propose a novel DRL-based MEC traffic offloading (NDMT) strategy for the 2022 FIFA World Cup, and the major contributions are summarized as follows:

(i) We compute the priorities of tasks and transfer DAG into a sequence of tasks according to the computed priorities, in which the scheduling process with respect to the tasks sequence is regarded as the Markov Decision Process (MDP)

(ii) A DNN model based on the Sequence to Sequence (S2S) form is designed and used to fit the scheduling strategy of MDP, where the DAG is converted into the sequence of tasks to be input into the DNN

(iii) The proximal policy optimization (PPO) method is used to train the DNN in the DRL, for obtaining the high stability and reliability

The rest paper is organized as follows. Section 2 reviews and compares the related work. Section 3 introduces the system framework of NDMT. Section 4 gives the problem description on traffic offloading in MEC. Section 5 presents the construction method of MDP. The traffic scheduling strategy based on DRL is proposed in Section 6. Section 7 reports the experimental results. Section 8 concludes this paper and gives the future research direction.

2. Related Work

In this section, we review the researches on the task (traffic) offloading of MEC in last three years (2018–2020) from two aspects, i.e., the heuristic methods and the DRL-based methods.

2.1. Heuristic Methods. There have been a lot of heuristic traffic offloading strategies in MEC. For example, the authors in [21] studied the scenario where multiple mobiles uploaded tasks to a MEC server in a single cell by allocating the limited server resources and wireless channels between mobiles devices. In particular, the authors formulated the optimization problem for the saved energy on the mobile devices with the tasks being dividable and utilized the selection maximum saved energy first algorithm to realize the solving process. In [22], the authors investigated an energy-efficient joint computation offloading, load balancing, and transmission power control problem and further proposed a heuristic algorithm to obtain the good traffic offloading while guaranteeing load balancing among the multiple servers. In [23], a distributed computation offloading and resource allocation optimization scheme in the heterogeneous networks with MEC was proposed, in which an optimization problem was formulated to provide the optimal computation offloading strategy. In [24], the traffic offloading strategy based on Software-Defined Networking (SDN) in the ultra-dense network was devised to minimize the delay while saving the battery life of mobile devices. It transformed this optimization problem into task placement subproblem and resource allocation subproblem, which could reduce 20% of the task duration with 30% energy saving. In [25], a MEC-enabled multicell wireless network

was considered where each base station is equipped with a MEC server that assisted mobile users in executing computation-intensive tasks via task offloading. It formulated the involved problem as a mixed integer nonlinear program, including the task offloading decision, uplink transmission power of mobile users, and resource allocation computation at the MEC servers. Furthermore, the authors in [26] jointly decided on the computing resource allocation for the hosted applications and designed a novel thoughtful decomposition based on the technique of the logic-based benders decomposition, with the heterogeneity in the requirements of the offloaded tasks (different computing requirements, latency, and so on) and limited MEC capabilities consideration. In [27], the authors studied the trade-off between task execution time and energy consumption at end-users under varying wireless channel conditions for soft real-time applications and involved tasks. It proposed a genetic algorithm with constrained mutation for optimal job partitioning and introduced an edge-proposing deferred acceptance algorithm to solve the preference based matching game. In [28], a task offloading algorithm that utilized the cache function of edge server was proposed. When the task with the same cached type of contents was uploaded to the edge server, the preset evaluation parameters took some factors into account to calculate the optimal processing position for the task. In [29], each base station was integrated with a MEC server in terms of the executing intensive computation task, and an iterative algorithm was proposed to solve the optimization problem in a single mobile user MEC system. In [30], an agent was introduced into the offloading of computation tasks, and a novel framework of agent-enabled task offloading in the unmanned aerial vehicle aided MEC was proposed to help the users obtain the good Quality of Experience (QoE). In [31], the authors addressed the problem of coordinating the offloading decisions of wireless devices that periodically generated computationally intensive tasks due to the various delay sensitive applications. In addition, they also developed a game theory based model and proposed a polynomial complexity algorithm for computing an equilibrium. In [32], the authors considered a system where most mobile devices migrated the duplicate computation tasks to the edge servers and shared the requested contents for computation tasks. Therein, an efficient Lyapunov online algorithm that could perform joint task offloading and dynamic data caching strategies for computation tasks or contents was proposed to reduce the overall latency of all mobile devices. Although these traffic offloading strategies had good performance, they could not decrease the network pressure well and could not be used for the traffic offloading in the FIFA World Cup directly due to the special traffic features, i.e., the periodicity, abruptness, and explosivity.

2.2. DRL-Based Methods. There have also been some DRL-based traffic offloading strategies in MEC. For example, in [33], a multiuser MEC system was considered, where the multiple users could perform computation offloading via wireless channels to a MEC server. Particularly, the RL-based optimization framework was introduced to tackle the resource allocation in wireless MEC. In [34], a deep-Q network based task offloading and resource allocation algorithm for the MEC was proposed, where each mobile terminal had the multiple tasks offloaded to the edge server. It also designed a joint task offloading decision and bandwidth allocation optimization to minimize the overall offloading cost in terms of energy cost, computation cost, and delay cost. In [35], the DRL was first proposed to solve the offloading problem of multiple service nodes for the cluster and multiple dependencies for mobile tasks in the large-scale heterogeneous MEC. In particular, it used the long short term memory network layer and the candidate network set to improve the deep-Q network algorithm in combination with the actual environment of the MEC. In [36], the authors considered MEC for a representative mobile user in a sliced Radio Access Network (RAN), where the multiple base stations were available to be selected for computation offloading. Meanwhile, a double DQN-based strategic computation offloading algorithm to learn the optimal policy without knowing a priori knowledge of network dynamics was proposed to break the curse of high dimensionality in state space. In [37], an intelligent offloading system for vehicular edge computing by leveraging DRL was constructed, in which both communication and computation states were modelled by the finite Markov chains. In [38], the authors investigated the problem of delay sensitive task scheduling and resource management on the server side in multiuser MEC scenario, where a new online algorithm based on DRL was devised to reduce average slowdown and average timeout period of tasks in the queue. In [39], the computing aware scheduling strategy in MEC was proposed, in which a support vector machine based multiclass classifier was adopted. Although these DRL-based strategies also showed good effect on the traffic scheduling, they always had some limitations to be improved, such as scheduling time, bandwidth utilization, and QoE of user. This motivates the study of this paper. Besides, this paper also gives a special application scenario, i.e., the 2022 FIFA World Cup, which can provide a significant reference.

3. System Framework

This section introduces the system framework of NDMT, including MEC-based traffic offloading architecture and DRL-based workflow for MEC traffic offloading. In addition, the abbreviations frequently used in this paper are listed in Table 1.

3.1. MEC-Based Traffic Offloading Architecture. In this paper, we present a MEC-based traffic offloading architecture, as shown in Figure 1, where the MEC servers are deployed at the edge network (e.g., AN) to provide the handy and low-latency computation services for the mobile users. In particular, MEC allocates the specialized hardware and software resources for each user and separates such resources by using the virtualization technology, so that the quality of service and the privacy of user can be guaranteed effectively.

Table 1: Abbreviations in alphabetical order.

Abbreviation	Full name
AI	Artificial Intelligence
AN	Access Network
ByLu	Baseline by Lu et al.
ByCh	Baseline by Chen et al.
CDN	Content Delivery Networks
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DNN	Deep Neural Networks
DRL	Deep Reinforcement Learning
DTU	Data Transmission Unit
MAN	Metropolitan Area Network
MDP	Markov Decision Process
MEC	Mobile Edge Computing
NDMT	Novel DRL-based MEC traffic offloading
PPO	Proximal policy optimization
QoE	Quality of Experience
RL	Reinforcement Learning
RNN	Recurrent Neural Network
S2S	Sequence to Sequence

For the side of user with the mobile device, the computation tasks generated from the mobile applications (e.g., the FIFA World Cup Traffic) can be performed at the local mobile device's Central Processing Unit (CPU) directly or can be sent to the MEC server via the Data Transmission Unit (DTU) and be performed by the corresponding service instance (i.e., the remote traffic offloading). Meanwhile, the traffic offloading module is used to make the scheduling decision for all tasks in the mobile device, including two functions, i.e., the execution way and the scheduling order.

An application program usually includes the multiple computation tasks that have the dependence among them, which can help realize the fine-grained traffic scheduling and enable the parallel processing for the multiple tasks. In this paper, we model the mobile application program related to the FIFA World Cup Traffic as a DAG, denoted by $G = (T, L)$, where T is the set of tasks and L is the set of links constructed by the tasks. Here, the arbitrary task is denoted by t_i; the arbitrary link is denoted by $l(t_i, t_j)$. Particularly, for $l(t_i, t_j)$, t_i is the precursor task of t_j while t_j is the successor task of t_i; that is to say, the performing of t_j relies on t_i. In DAG, the task without any precursor task is called the entry task, whilst the task without any successor task is called exit task. In addition, it allows an application program to have the multiple entry tasks and the multiple exit tasks in case of the parallel processing. For t_i, it has three attributes, i.e., the input volume of traffic, the number of CPU cycles, and the output volume of traffic, denoted by Iv_i, Cy_i, and Ov_i respectively, and the corresponding values can be obtained by the program analyzer, just like in [40], which reflect the required transmission cost and computation cost.

As depicted in Figure 1, the performing of task has two ways, i.e., offloading performing and local performing. If t_i is scheduled to the remote edge server for performing, the whole process consists of three phases, i.e., task sending, edge performing, and result returning. At the first phase, the volume of traffic Iv_i is transmitted to the remote edge server. Let Rul denote the transmission rate of uplink, and the required transmission time Tul_i of t_i is defined as follows:

$$\mathrm{Tul}_i = \frac{Iv_i}{\mathrm{Rul}}. \tag{1}$$

Then, at the edge performing phase, the Cy_i CPU cycles are performed at the corresponding server instance in the MEC server. Let Fv denote the virtual clock frequency of server instance for t_i, and the required performing time Tep_i of t_i is defined as follows:

$$\mathrm{Tep}_i = \frac{Cy_i}{\mathrm{Fv}}. \tag{2}$$

Similar to the first phase, the returning time at the result returning phase Tdl_i is defined as follows:

$$\mathrm{Tdl}_i = \frac{Ov_i}{\mathrm{Rdl}}, \tag{3}$$

where Rdl is the transmission rate of downlink.

Furthermore, let Tof_i denote the total time cost in case of performing t_i via the traffic offloading way, and it concludes three parts of time costs, i.e.,

$$\mathrm{Tof}_i = \mathrm{Tul}_i + \mathrm{Tep}_i + \mathrm{Tdl}_i. \tag{4}$$

If t_i is performed at the local mobile device, it is unnecessary to upload and download the data to which t_i corresponds; this is to say, the total time cost only depends on the local computation overhead with respect to the consumption of CPU resources. Let Tlo_i denote the total time cost in case of performing t_i at the local mobile device, and we have

$$\mathrm{Tlo}_i = \frac{Cy_i}{\mathrm{Fl}}, \tag{5}$$

where Fl is the local CPU's clock frequency.

3.2. DRL-Based Workflow for MEC Traffic Offloading. In this section, we describe the DRL-based workflow for MEC traffic offloading, as shown in Figure 2. The whole workflow consists of four main modules, i.e., MEC problem description, MDP construction, DNN-based strategy fitting, and PPO-based reinforcement training. Among them, the first module is to present the involved problem on traffic offloading, including the local scheduling and the offloading scheduling. The second module is to transfer the scheduling process with the tasks sequence as the MDP, where the priority of task is computed and regarded as the transferred attribute. In the third module, the S2S-based DNN model is used to fit the scheduling strategy. In the last module, the PPO-based DRL method is adopted to train the DNN because the PPO has the good stability and reliability.

Furthermore, during the whole process of workflow, all processing units (including CPU, DTU, virtual CPU, and virtual DTU) only perform and send one task, which indicates that the multiple tasks preemption phenomenon is not allowed. In addition, the DAG-based task scheduling in MEC satisfies the following two features. (1) Given the bandwidth limitation of edge network, the transmission rate

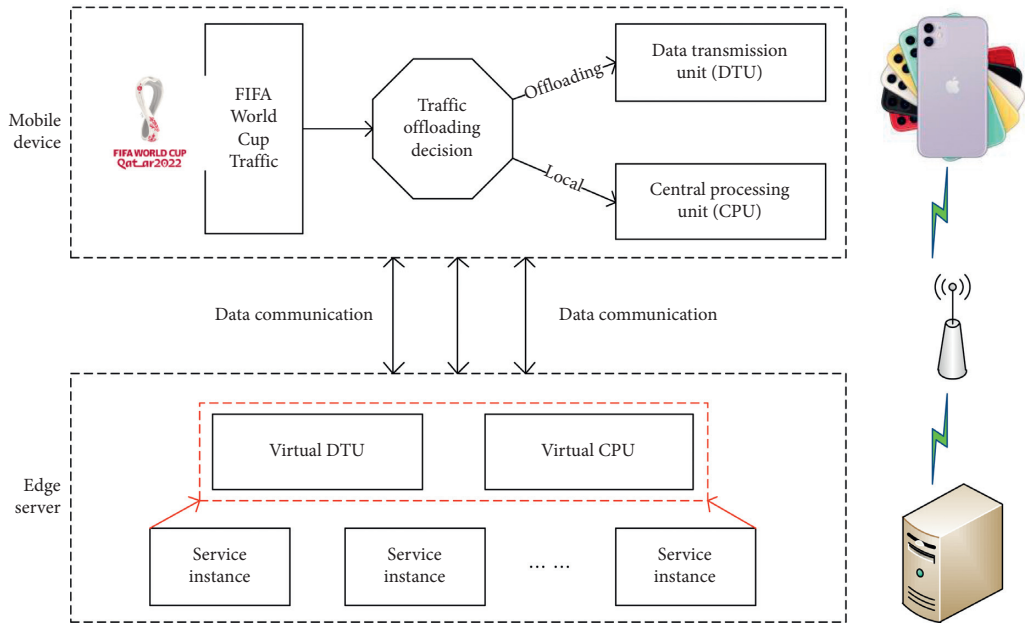

Figure 1: MEC-based traffic offloading architecture.

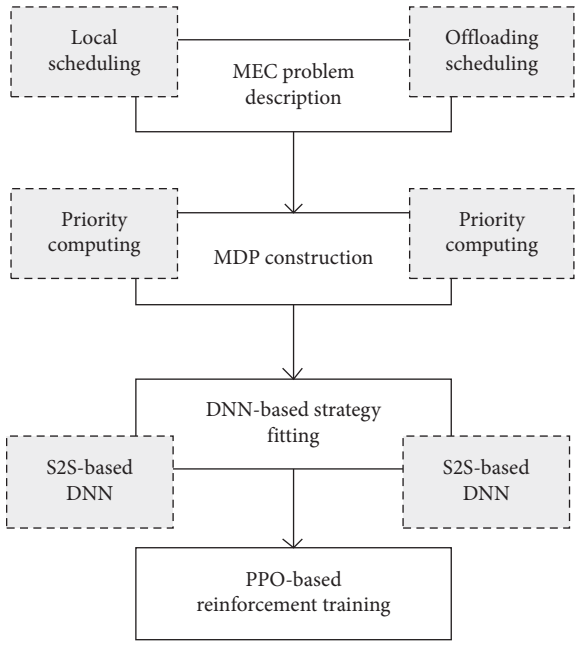

Figure 2: DRL-based workflow for MEC traffic offloading.

between mobile device and edge server (uplink or downlink) keeps the fixed value. (2) The whole scheduling can be finished until the computation result is returned to the mobile device.

4. Problem Description

At first, we define four timestamps with respect to the completed time, i.e., task sending, edge performing, result returning, and local performing, denoted by Tsos_i, Tsop_i, Tsor_i, and Tslp_i, respectively. If t_i is performed at the local device, we have $\mathrm{Tsos}_i = \mathrm{Tsop}_i = \mathrm{Tsor}_i = 0$; otherwise, $\mathrm{Tslp}_i = 0$. In particular, before t_i is scheduled, it is required that all precursor tasks of t_i have to be performed in advance.

Consider the condition where t_i is performed at the local mobile device, let RTslp_i be the ready timestamp regarding scheduling t_i, and we have

$$\mathrm{RTslp}_i = \max_{t_j \in \mathrm{pre}_i} \left\{ \mathrm{Tslp}_j, \mathrm{Tsor}_j \right\}, \tag{6}$$

which indicates that RTslp_i is the earliest timestamp in terms of such condition where all precursor tasks are completed, pre_i is the set of precursor tasks with respect to t_i, and t_j is one precursor task of t_i. In particular, it perhaps needs the queueing for each task before being performed at CPU; thus, the starting timestamp may be not equal to the ready timestamp. Let STslp_i denote the starting timestamp regarding scheduling t_i, and we have $\mathrm{STslp}_i \geq \mathrm{RTslp}_i$, satisfying

$$\mathrm{Tslp}_i = \mathrm{STslp}_i + \mathrm{Tlo}_i. \tag{7}$$

Consider the condition where t_i is scheduled to the remote edge server for performing, let RTsos_i denote the ready timestamp regarding sending t_i, and we have

$$\mathrm{RTsos}_i = \max_{t_j \in \mathrm{pre}_i} \left\{ \mathrm{Tsos}_j, \mathrm{Tslp}_j \right\}, \tag{8}$$

where all precursor tasks of t_i are performed at the local mobile device or the remote server. Similarly, let STsos_i denote the starting timestamp when t_i is sent, and we have $\mathrm{STsos}_i \geq \mathrm{RTsos}_i$ and $\mathrm{Tsos}_i = \mathrm{STsos}_i + \mathrm{Tul}_i$.

Then, let RTsop_i denote the ready timestamp regarding performing t_i via the service instance, and we have

$$\mathrm{RTsop}_i = \max \left\{ \mathrm{Tsos}_i, \max_{t_j \in \mathrm{pre}_i} \mathrm{Tsop}_j \right\}. \tag{9}$$

Among them, RTsop_i depends on the fact that the input traffic of t_i is completed; i.e., the transmission of Iv_i is

finished. Similarly, the starting performing timestamp of t_i at the virtual CPU relies on the corresponding queuing situation, and we have $\mathrm{STsop}_i \geq \mathrm{RTsop}_i$ and $\mathrm{Tsop}_i = \mathrm{STsop}_i + \mathrm{Tep}_i$, where STsop_i is the starting timestamp when t_i is performed via the service instance.

Finally, when the performing of v_i is finished, v_i gets into the ready status of result returning. Let RTsor_i denote the ready timestamp regarding returning the computed result of t_i, and we have $\mathrm{RTsor}_i = \mathrm{Tsop}_i$. Furthermore, let STsor_i denote the starting timestamp when the computed result of t_i is returned, and we have $\mathrm{STsor}_i \geq \mathrm{RTsor}_i$ and $\mathrm{Tsor}_i = \mathrm{STsor}_i + \mathrm{Tdl}_i$.

Let $\mathrm{TTotal}(G)$ denote the required time to perform an application program (i.e., all involved tasks have completed the results returning), and we have

$$\begin{aligned}\mathrm{TTotal}(G) &= \max_{t_i \in \mathrm{exit}(G)} \{\max\{\mathrm{Tsor}_i, \mathrm{Tslp}_i\}\} \\ &= \max_{t_i \in \mathrm{exit}(G)} \{\max\{\mathrm{Tof}_i, \mathrm{Tlo}_i\}\},\end{aligned} \tag{10}$$

where $\mathrm{exit}(G)$ is the combinatorial set of exit tasks in G. This paper considers the delay sensitive FIFA World Cup traffic; therefore, the main purpose is to maximize the QoE while minimizing $\mathrm{TTotal}(G)$ based on different scheduling strategies. Particularly, during the process of scheduling, each task's execution way and scheduling order should be determined. In addition, we emphasize that the scheduling order of task at the local mobile device and that at the service instance keep consistent.

5. MDP Construction

5.1. Priority Computation. For all traffic scheduling strategies, computing the priority for each task is the indispensable operation. Furthermore, based on the computed priority, the scheduling order can be determined [41]. For the arbitrary t_i in the DAG, its computation cost with respect to the time can be obtained by(4)fd4 and expressed by Tof_i. Based on Tof_i, the priority of t_i is defined as follows:

$$\mathrm{Pr}_i = \max_{t_j \in \mathrm{suc}_i} \mathrm{Pr}_j + \mathrm{Tof}_i, \tag{11}$$

where Pr_i is the priority of t_i and suc_i is the set of successor tasks with respect to t_i. It is obvious that the (11)fd11 shows the recursive form. If t_i is the exit task, we have

$$\mathrm{Pr}_i = \mathrm{Tof}_i, \quad t_i \in \mathrm{exit}(G). \tag{12}$$

Then, the DAG is depth-firstly traversed with starting from the exit task, and all tasks' priorities can be obtained by (11) and (12)fd12. These tasks are arranged according to the corresponding priorities in the descending order, and the scheduling sequence of tasks can be defined as follows:

$$Q = (t_1', t_2', \ldots, t_n'), \tag{13}$$

where n is the number of tasks (or nodes in DAG). In particular, Q is the special topological sorting result on G, and the original dependence among tasks can be guaranteed according to the sequential scheduling in Q.

5.2. Construction Method. The sequential scheduling decision process for all sorted tasks in Q can be modelled as one MDP, denoted by $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{D}_0, \mathscr{R}, \lambda)$, where $\mathscr{S}$, $\mathscr{A}$, $\mathscr{P}$, $\mathscr{D}_0$, $\mathscr{R}$, and λ are the state space, action space, state-transition matrix, the probability distribution of initial state, reward function, and discount factor, respectively.

Let k denote the scheduled number of tasks in Q, and the current state space can be expressed as follows:

$$\begin{aligned}\mathscr{S} &= \left\{s | st = n\left(G, k, \mathscr{A}^k\right)\right\}, \\ \mathscr{A}^k &= (a_1, a_2, \ldots, a_k),\end{aligned} \tag{14}$$

where $\mathscr{A}^k$ is used to describe the scheduling condition (i.e., state space) on the first k tasks in Q and a_k is used to record the execution way of task: $a_k = 1$ means that k-th task in Q is performed in the offloading way; otherwise, it is performed at the local mobile device.

Furthermore, let $Q_{1->i}$ denote the scheduled sequence with respect to the first i tasks in Q, and we can construct a scheduled subgraph of G, denoted by $G_{1->i} = (T', L')$; here $T' \subseteq T$, $L' \subseteq L$, $G_{1->n} = G$ and $G_{1:\,0} = \Phi$. Under such condition, in order to minimize the scheduling time, we give a reward function for the current task, defined as follows:

$$r_i = \mathscr{R}(s_i, a_i) = \mathrm{TTotal}\left(G_{1->i}\right) - \mathrm{TTotal}\left(G_{1->i+1}\right), \tag{15}$$

which refers to the time difference between a_i performed before and that after in terms of s_i.

Moreover, the traffic offloading decision module in Figure 1 can be defined as a conditional probability function, denoted by $\theta(a_i | ts_i)$. From the initial state s_0, upon the traffic offloading decision module completes an action, the system enters a new state and further gets the corresponding reward based on (15)fd15 until the last task in Q is completed. According to the above statements, the whole task scheduling process based on MDP is described as follows:

$$\mathrm{MDP} := \left(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{n-1}\right), \tag{16}$$

where s_{n-1} is the termination state to mean that all tasks have been completed. Then, the accumulated reward with the discount factor consideration is defined as follows:

$$\begin{aligned}R &= \sum_{i=0}^{n-1} \lambda * r_i = \lambda * \left(\mathrm{TTotal}\left(G_{1->i}\right) - \mathrm{TTotal}\left(G_{1->i+1}\right)\right) \\ &= -\lambda * \mathrm{TTotal}(G),\end{aligned} \tag{17}$$

which indicates that the maximization of the accumulated reward with the discount factor consideration is consistent with the minimization of the total scheduling time.

6. DRL-Based Traffic Scheduling Strategy

In fact, the DAG has the feature of diversity and the involved state space is infinitely great; thus, it is impossible to obtain the corresponding state-transition matrix in advance. Given this, this paper uses the DRL to find the optimal scheduling strategy for the traffic scheduling decision module.

6.1. DNN-Based Strategy Fitting. We employ DNN to fit $\theta_x(a_i|ts_i)$, where x is the set of parameters related to DNN. As we know, the input of DNN is s_i which is related to G but G cannot be input to the DNN directly due to the restriction of data features; therefore, the DAG is converted into the sequence of tasks (just like Q) to be input into the DNN. For such sequence, it consists of the following three vectors: (1) time vector including Tul_i, Tep_i, Tdl_i, and Tlo_i, (2) precursor vector including all precursor indexes, and (3) successor vector including all successor indexes. Among them, the size of precursor/successor vector is set as a fixed value, denoted by sz. If the number of precursor/successor tasks is smaller or equal to sz, the corresponding locations are filled by -1; otherwise, the extra parts are ignored directly.

The output of DNN is the probability distribution of executable actions based on the current task state. In fact, for the current task, its scheduling strategy decision action (a_i) has the direct influence on the next task's state (s_{i+1}); therefore, this paper leverages the S2S-based DNN structure model, including encoder and decoder, as shown in Figure 3. In particular, both encoder and decoder are realized by the Recurrent Neural Network (RNN). Meanwhile, the encoder receives such input sequence in turn and finally outputs the hidden layer(s) as the features of DAG. The decoder initializes its own hidden layer(s) by using the output result from the encoder. In addition, the decoder sequentially inputs the scheduling actions ($\mathscr{A}^i$) and then outputs the corresponding $\theta_x(a_i|ts_i)$. The above process can be still performed until s_{n-1} is completed.

6.2. PPO-Based Reinforcement Training. In order to obtain the optimal traffic scheduling decision, the training objective of DRL can be defined as follows:

$$\max_x \mathrm{Objective}(x) = \max_x \left(\theta_x^*\left(\mathscr{A}^{n-1}|tG\right) * \sum_{i=0}^{n-1} r_i \right),$$
$$\theta_x^*\left(\mathscr{A}^{n-1}|tG\right) = \prod_{i=0}^{n-1} \theta_x a_i|s_i, \tag{18}$$

which indicates that the whole traffic scheduling decision depends on all scheduling decisions of tasks. In this paper, we use the PPO [42] to train such traffic scheduling decisions, which can accelerate the efficient convergence while guaranteeing the scalability and reliability.

Since the scheduled DAG has the feature of diversity and the involved state space is infinitely great, it is impossible to search all DAGs. With such consideration, we can continuously collect the scheduled DGAs after strategy deployment to construct the training set related to the DAG. Then, based on the training set, we also can train the traffic scheduling strategies. In addition, in order to obtain the better effect on the convergence, we scale back the obtained reward by each scheduling decision during the process of training; that is, r_i always keeps in $[0, 1]$.

7. Performance Evaluation

7.1. Setup. The proposed NDMT is implemented by the C++ programming, and the involved simulation parameters are set in Table 2. Among them, the network scale is dynamically changing from $n = 20$ to $n = 60$ according to the pattern of Figure 4 due to the fact that it accords with the network deployment feature to offload the FIFA World Cup traffic, where the step length is 10. Particularly, for each scale, there are 2500 DAGs being used for training the DRL and there are 300 DAGs being used for testing the DRL; in other words, there are $5 * 2500 = 12500$ and $5 * 300 = 1500$ DAGs being used to train and test the DRL, respectively. The TensorFlow [43] is used to realize the DRL, where both encoder and decoder have 256 hidden neurons. In addition, the method of layer normalization [44] is used to improve the training efficiency.

Furthermore, the researches from [35, 36], respectively, are used as two baselines, because they are the latest research representatives on the DRL-based traffic offloading in MEC. Therein, [35] is proposed by Lu et al. while [36] is proposed by Chen et al. and in this paper they are abbreviated to ByLu and ByCh, respectively. In terms of experiments, the DRL convergence for traffic offloading is analyzed firstly. Then, three metrics, i.e., scheduling time, bandwidth utilization, and QoE are used to measure the proposed NDMT's performance.

7.2. Convergence Analysis. This section verifies the convergence of DRL based on 12500 DAGs, in which the average accumulated reward is considered as the evaluation metric. For each training, we record the corresponding training result. When the training within one period is finished, we input 300 testing DAGs into the S2S-based DNN and the related traffic offloading strategy is obtained. Then, the obtained traffic offloading strategy is simulated and the average accumulated reward can be computed. The relationship between the average accumulated reward and the training period is shown in Figure 5. We can observe that the whole training process includes three stages, i.e., rapid increasing stage, stable increasing stage, and stationary stage, and the stationary stage is reached with the need of 190 training periods. It indicates that the proposed NDMT can converge to the optimal state space and further obtain the optimal traffic scheduling decision.

7.3. Scheduling Time. The scheduling time is defined as the time difference between the time point when the first task is sent and that when the computation result of the last task is obtained by the local mobile device. The average scheduling times of NDMT, ByLu, and ByCh are shown in Figure 6.

We can observe that NDMT always has the smallest average scheduling time, followed by ByLu and ByCh, and there are two main reasons. On the one hand, NDMT considers the process of scheduling as the MDP, which can improve the processing speed for each task in the S2S-DNN structure model. On the other hand, NDMT adopts the layer normalization method to increase the training efficiency, which can further save the training time and strive for the scheduling time as small as possible. For two baselines, ByCh does not consider a priori knowledge of network dynamics to learn the optimal policy and the state space is the high

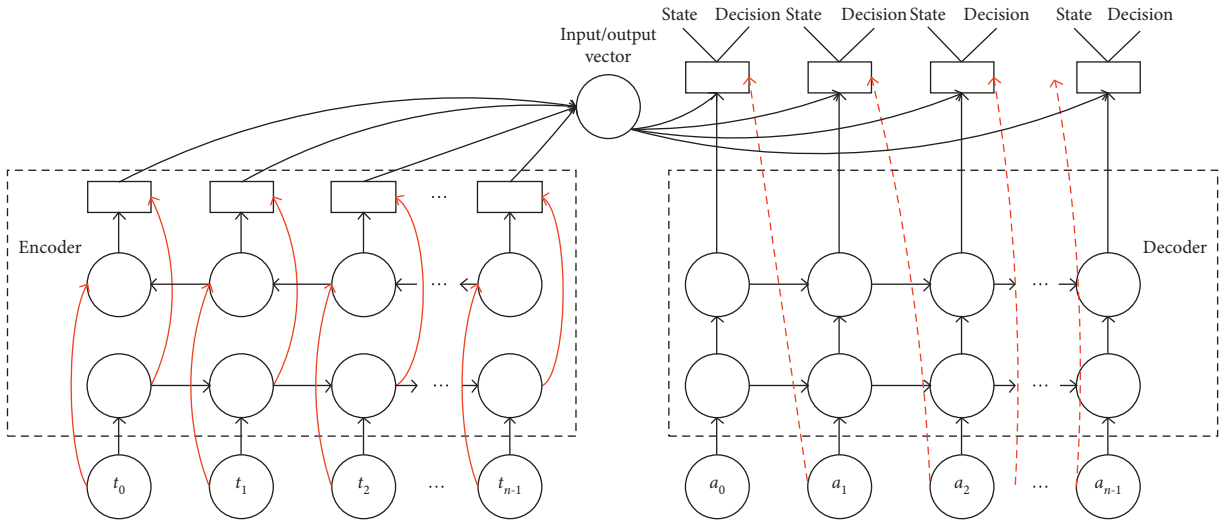

Figure 3: S2S-based DNN model.

Table 2: Simulation settings.

Parameter	Setting
Fl	2×10^9 cycles/s
Fv	1.5×10^10 cycles/s
Tul_i	10 Mbps
Tdl_i	10 Mbps
λ	0.85
sz	14
n	20, 30, 40, 50, 60
The number of simulations	100
The learning rate in DRL	2×10^{-4}
The training number of DAGs for n	2500
The testing number of DAGs for n	300

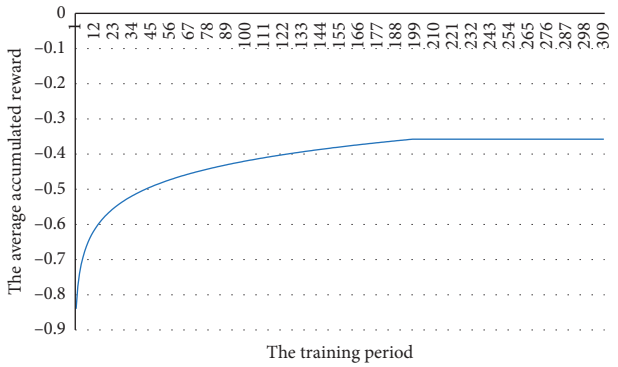

Figure 5: The convergence analysis result.

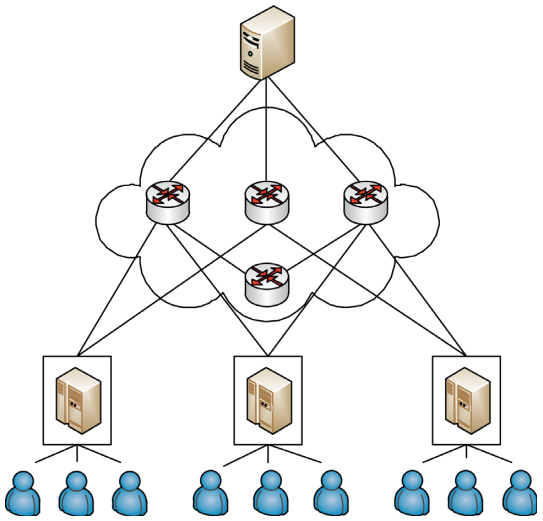

Figure 4: The network topology pattern used for simulation.

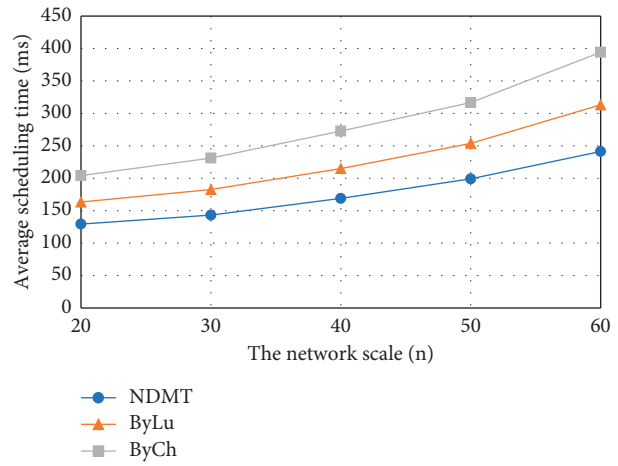

Figure 6: The average scheduling times among NDMT, ByLu, and ByCh.

dimensional; thus, it has larger average scheduling time than ByLu. In addition, we can also observe that the average scheduling time becomes larger and larger with the increasing of network scale, which results from two aspects. On the one hand, it needs much more time to train the DAGs; on the other hand, it needs much time to compute more tasks.

7.4. Bandwidth Utilization. The bandwidth utilization is defined as the ratio of the used bandwidth and the total network bandwidth. The average bandwidth utilizations of NDMT, ByLu, and ByCh are shown in Figure 7.

We can observe that NDMT always has the highest average bandwidth utilization, followed by ByLu and ByCh, which is regarded as an important benefit of NDMT. Regarding this, there are no the concrete reasons. Furthermore, we can observe that the average bandwidth utilization basically remains unchanged, i.e., having the strong stability for different network scales; this is because NDMT always can converge to the optimal solution (see Figure 5). It suggests that the proposed NDMT has the considerable reference value for the following 2022 FIFA World Cup. However, the average bandwidth utilizations of ByLu and ByCh become lower and lower with the increasing of network scale because the corresponding convergences are nondeterminate.

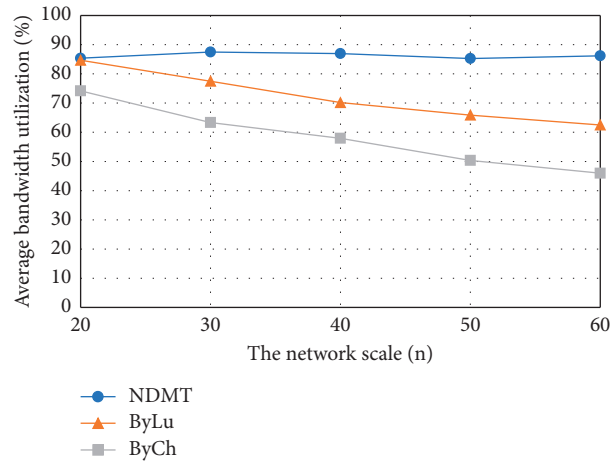

Figure 7: The average bandwidth utilizations among NDMT, ByLu, and ByCh.

7.5. QoE of User. We use the watching fluency to measure the QoE of user, and the watching fluency is defined as the number of network lags per 10 mins. The average numbers of network lags of NDMT, ByLu and ByCh are shown in Figure 8.

We can observe that NDMT always has the smallest average number of network lags, followed by ByCh and ByLu, which further indicates that the user has the best watching experience in NDMT, because NDMT has the highest bandwidth utilization and the smallest response time. For ByLu and ByCh, the latter deploys the large number of base stations to offload traffic and thus the required response time is relatively smaller than the former; as a result, ByCh has better QoE of user than ByLu. In addition, we can also observe that NDMT has the most stable QoE of user but the two baselines do not have, and similar reasons can be found in the above section. Moreover, the experimental results suggest that the users can enjoy the best experience when watching the following 2022 FIFA World Cup by the personal mobile devices under the environment of MEC.

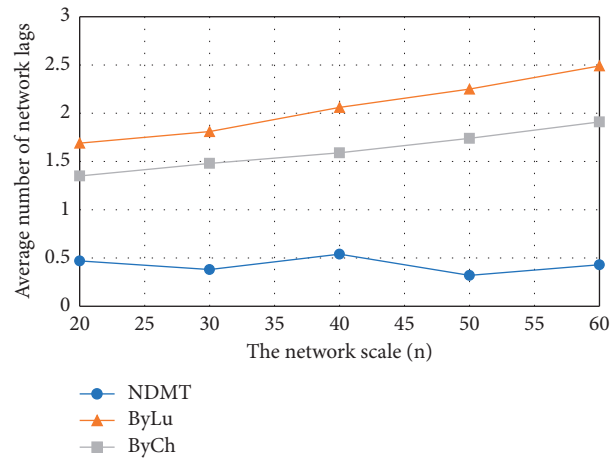

Figure 8: The average numbers of network lags among NDMT, ByLu, and ByCh.

8. Conclusions

In this paper, we investigate the MEC traffic offloading strategy based on DRL. At first, we introduce the proposed system framework, including MEC-based traffic offloading architecture and DRL-based workflow for MEC traffic offloading. Then, we give the problem description, i.e., minimizing the total performing time for one application program. For the concrete scheduling strategy, it includes three parts. At the first part, we compute the priorities of tasks and transfer DAG into a sequence of tasks according to the computed priorities, and the scheduling process with respect to the tasks sequence is regarded as the MDP. At the second part, one S2S-based DNN model is used to fit the scheduling strategy. At the third part, the PPO method is employed to train the DNN. We do the simulation experiments based on the TensorFlow including the convergence analysis and the performance comparison. Meanwhile, the scheduling time, bandwidth utilization, and QoE of user are considered three performance evaluation metrics, and we observe that the proposed NDMT outperforms two baselines. Based on the nice experiment results, we think that the proposed NDMT can be regarded as a feasible and efficient reference for the following 2022 FIFA World Cup held in Qatar. In the future, we plan to improve NDMT from the following two aspects. At first, more datasets are collected and used to train the DNN; then, NDMT is deployed at a real network environment to further verify its performance.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Z. Yan, P. Zhang, A. V. Vasilakos et al., "A survey on trust management for Internet of Things," *Journal of Network and Computer Applications*, vol. 42, no. 3, pp. 120–134, 2014.

[2] L. Quijano-Sanchez, I. Cantador, M. E. Cortes-Cediel et al., "Recommender systems for smart cities," *Information Systems*, vol. 92, pp. 1–22, 2020.

[3] Cisco Annual Internet Report (2018–2023) White Paper, http://www.cisco.com/c/en/us/solutions.

[4] F. Pacheco, E. Exposito, and M. Gineste, "A framework to classify heterogeneous Internet traffic with Machine Learning and Deep Learning techniques for satellite communications," *Computer Networks*, vol. 173, pp. 1–21, 2020.

[5] Y. Koike and A. Inoue, "High-speed graded-index plastic optical fibers and their simple interconnects for 4K/8K video transmission," *Journal of Lightwave Technology*, vol. 34, no. 6, pp. 1551–1555, 2016.

[6] S. Sukhmani, M. Sadeghi, M. Erol-Kantarci, and A. El Saddik, "Edge caching and computing in 5G for mobile AR/VR and tactile Internet," *IEEE MultiMedia*, vol. 26, no. 1, pp. 21–30, 2019.

[7] Y. Liu, S. Liu, Y. Wang, and H. Zhao, "Video coding and processing: a survey," *Neurocomputing*, vol. 408, pp. 331–344, 2020.

[8] S. Cui, M. R. Asghar, and G. Russello, "Multi-CDN: towards privacy in content delivery networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, pp. 984–999, 2020.

[9] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on peer-assisted content delivery networks," *Computer Networks*, vol. 116, pp. 79–95, 2017.

[10] H. Lin, S. Zeadally, Z. Chen et al., "A survey on computation offloading modeling for edge computing," *Journal of Network and Computer Applications*, vol. 169, pp. 1–13, 2020.

[11] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: a machine learning-based perspective," *Computer Networks*, vol. 182, pp. 1–15, 2020.

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

[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[14] M. Zamzam, T. Elshabrawy, and M. Ashour, "Resource management using machine learning in mobile edge computing: a survey," in *Proceedings of the 2019 International Conference on Intelligent Computing and Information Systems*, pp. 112–117, Cairo, Egypt, December 2019.

[15] G. Madraki and R. P. Judd, "Recalculating the length of the longest path in perturbed directed acyclic Graph," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1560–1565, 2019.

[16] J. D. Ullman, "NP-complete scheduling problems," *Journal of Computer and System Sciences*, vol. 10, no. 3, pp. 384–393, 1975.

[17] N. C. Luong, D. T. Hoang, S. Gong et al., "Applications of deep reinforcement learning in communications and networking: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[18] S. Cheng, L. Ma, H. Lu et al., "Evolutionary computation for solving search-based data analytics problems," *Artificial Intelligence Review*, 2020, in press.

[19] M. M. Drugan, "Reinforcement learning versus evolutionary computation: a survey on hybrid algorithms," *Swarm and Evolutionary Computation*, vol. 44, pp. 228–246, 2019.

[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on Graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020, in press.

[21] F. Wei, S. Chen, and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," *China Communications*, vol. 15, no. 11, pp. 149–157, 2018.

[22] S. Li, D. Zhai, P. Du et al., "Energy-efficient task offloading, load balancing, and resource allocation in mobile edge computing enabled IoT networks," *Science China Information Sciences*, vol. 62, pp. 1–3, 2019.

[23] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, 2018.

[24] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[25] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[26] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[27] X. Zhang and S. Debroy, "Adaptive task offloading over wireless in mobile edge computing," in *Proceedings of the ACM/IEEE Symposium on Edge Computing*, pp. 323–325, Washington, DC, USA, 2019.

[28] J. Fang and W. Zeng, "Offloading strategy for edge computing tasks based on cache mechanism," in *Proceedings of the International Conference on Computing and Artificial Intelligence*, pp. 129–134, Cairo, Egypt, April 2020.

[29] G. Wang, Q. Li, and X. Yu, "Computation task offloading for minimizing energy consumption with mobile edge computing," *Lecture Notes in Electrical Engineering*, vol. 571, pp. 2117–2123, 2020.

[30] R. Wang, Y. Cao, A. Noor, T. A. Alamoudi, and R. Nour, "Agent-enabled task offloading in UAV-aided mobile edge computing," *Computer Communications*, vol. 149, pp. 324–331, 2020.

[31] S. Josilo and G. Dan, "Computation offloading scheduling for periodic tasks in mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 667–680, 2020.

[32] N. Zhang, S. Guo, Y. Dong et al., "Joint task offloading and data caching in mobile edge computing networks," *Computer Networks*, vol. 182, pp. 1–10, 2020.

[33] J. Li, H. Gao, T. Lv et al., "Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing," in *Proceedings of the International Conference on Machine Learning and Intelligent Communications*, pp. 33–42, Hangzhou, China, July 2018.

[34] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Communications and Networks*, vol. 5, no. 1, pp. 10–17, 2019.

[35] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning," *Future Generation Computer Systems*, vol. 102, pp. 847–861, 2020.

[36] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019.

[37] Z. Ning, P. Dong, and X. Wang, "Deep reinforcement learning for vehicular edge computing: an intelligent offloading system," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 6, pp. 1–24, 2019.

[38] H. Meng, D. Chao, R. Huo et al., "Deep reinforcement learning based delay-sensitive task scheduling and resource management algorithm for multi-user mobile-edge computing systems," in *Proceedings of the International Conference on Mathematics and Artificial Intelligence*, pp. 66–70, Chegndu China, April 2019.

[39] K. Wang, X. Yu, W. Lin et al., "Computing aware scheduling in mobile edge computing system," *Wireless Networks*, vol. 25, pp. 1–17, 2019.

[40] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[41] H. Topcuoglu, S. Hariri, and M. Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[42] L. Ma, S. Cheng, and Y. Shi, "Enhancing learning efficiency of brain storm optimization via orthogonal learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, p. 1. in press, 2020.

[43] TensorFlow, https://tensorflow.google.cn.

[44] J. Ba, J. R. Kiros, and G. E. Hinton, *Layer Normalization*, Cornell University, Ithaca, NY, USA, 2016.