Research Article

An Intelligent Adaptive Algorithm for Servers Balancing and Tasks Scheduling over Mobile Fog Computing Networks

Xuejing Li¹, Yajuan Qin¹, Huachun Zhou¹, Du Chen¹, Shujie Yang² and Zhewei Zhang³

¹National Engineering Laboratory for Next Generation Internet Interconnection Devices, Beijing Jiaotong University, Beijing 100044, China

²State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

³The Third Research Laboratory, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

Correspondence should be addressed to Xuejing Li; lixuejing@bjtu.edu.cn and Yajuan Qin; yjqin@bjtu.edu.cn

Received 24 April 2020; Revised 6 June 2020; Accepted 16 June 2020; Published 23 July 2020

Academic Editor: Fuhong Lin

Copyright © 2020 Xuejing Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the increasing popularity of terminals and applications, the corresponding requirements of services have been growing significantly. In order to improve the quality of services in resource restrained user devices and reduce the large latency of service migration caused by long distance in cloud computing, mobile fog computing (MFC) is presented to provide supplementary resources by adding a fog layer with several servers near user devices. Focusing on cloud-aware MFC networks with multiple servers, we formulate a problem with the optimization objective to improve the quality of service, relieve the restrained resource of user device, and balance the workload of participant server. In consideration of the data size of remaining task, the power consumption of user device, and the appended workload of participant server, this paper designs a machine learning-based algorithm which aims to generate intelligent adaptive strategies related with load balancing of collaborative servers and dynamic scheduling of sequential tasks. Based on the proposed algorithm and software-defined networking technology, the tasks can be executed cooperatively by the user device and the servers in the MFC network. Besides, we conducted some experiments to verify the algorithm effectiveness under different numerical parameters including task arrival rate, available server workload, and wireless channel condition. The simulation results show that the proposed intelligent adaptive algorithm achieves a superior performance in terms of latency and power consumption compared to candidate algorithms.

1. Introduction

With the blossom of Internet of Things (IoT) [1, 2], we need to process a huge amount of data deriving from various applications like real-time monitoring [3]. Meanwhile, the requirements of corresponding services have been growing significantly and most of the end devices have finite computing resource and limited energy resource. The cloud computing (CC) technology and service migration approach are proposed for enabling users to utilize powerful cloud servers which can effortlessly achieve service function with high performance [4]. However, it would result in heavy traffic burden over the transmission network and large responding latency, since the cloud servers are far away from end users. By deploying fog servers for providing supplementary

resources between cloud and users, mobile fog computing (MFC) is designed as a novel computing paradigm in order to reduce responding latency of services and avoid traffic congestion of network [5]. Hence, in the cloud-aware MFC network [6], most services of users can be performed at the fog servers and others at the cloud servers. Differently, mobile edge computing (MEC) mainly considers the task offloading problem on mobile user devices [7].

Note that the cloud-aware MFC is different from CC and MEC, it is necessary to design an appropriate intelligent management strategy for the special system architecture. Based on software-defined networking (SDN) technology [8], we assume that the tasks of services in each user can be executed cooperatively by multiple servers in MFC network according to appropriate strategy. As shown in Figure 1, we

FIGURE 1: MFC architecture consisted of multiple servers in both fog layer and cloud layer.

focus on an architecture including fog layer servers and cloud layer servers, to which the sequential tasks of computational services can be scheduled, respectively. Considering the data size of remaining task, the power consumption of user device [9, 10], and the appended workload of participant server [11], this paper designs an algorithm for deriving the intelligent adaptive strategies related with load balancing of collaborative servers and dynamic scheduling of sequential tasks. Particularly, the optimization objective is to improve the quality of service, relieve the restrained resource of user device, and balance the workload of participant server. Since conventional algorithms are not suitable to find optimal solution of this NP-hard combinatorial optimization problem [12, 13], the proposed algorithm is designed based on some machine learning (ML) methods including deep learning and reinforcement learning.

There are some contributions of this paper as follows. First, we formulate a combinatorial problem related with load balancing of collaborative servers and dynamic scheduling of sequential tasks. Second, we propose an ML-based algorithm to derive intelligent adaptive strategies for improving the quality of service, relieving the restrained resource of user device, and balancing the workload of servers. Third, we conduct some simulation experiments based on Python platform and evaluate the proposed algorithm in terms of latency and power consumption compared to candidate algorithms.

The remaining part of this paper is organized as follows. Section 2 reviews related works mainly about task scheduling and server balancing. Section 3 introduces a MFC system and formulates a joint problem. Section 4 presents an intelligent adaptive algorithm based on ML methods. Section 5 describes the experimental setup and illustrates simulation results. Finally, the paper is concluded with some future research directions in section 6.

2. Review of Related Work

To cope with the complicated problems which are rapidly changing over time, it becomes more challenging to solve adaptive fine-grained control problems related about dynamic scheduling of task and load balancing of servers. Applied to artificial intelligence (AI) and ML methods, the complicated problems can be solved efficiently [14]. As an important branch of ML, the deep learning (DL)-based algorithms specialize in approximating the input-output nonlinear mapping relationship to solve expensive computational problems [15]. Extended from DL, deep neural network (DNN) is designed from biological neural network, and convolutional neural network (CNN) is applied to compress the input values. The supervised learning algorithm can achieve data classification and prediction by a training process with manually labeled samples. To solve the problems about dynamic programming and Markov decision process (MDP), it is effective to use learning-based algorithms without the external supervisor to minimize long-term cost, such as incremental learning and reinforcement learning (RL) [16]. With a learning-based structure, rational solutions can be obtained after sufficient training episodes with successive input values [17]. Further, Q-learning is a simple RL-based algorithm with tabular-search nature, which is not suitable for handling dynamic problems with high dimensional space. To improve the algorithm efficiency, deep Q-network (DQN) is one kind of deep RL (DRL)-based algorithms combined with DL technique. Reference [18] presented a comprehensive survey on deep learning applied in mobile and wireless networking. In reference [19], learning-based approaches were proposed for the radio resource interference management and wireless transmitting power allocation. Xu et al. [20] designed a novel DRL-based framework for power-efficient resource

allocation while meeting the demands of wireless users in highly dynamic cloud RANs.

There have been many researches which apply these intelligent algorithms to offloading decision-making. In paper [21], the authors proposed a security and cost-aware computation offloading strategy for mobile users in MEC environment. Under the risk probability constraints, the goal of this strategy was to minimize the overall cost, which was including energy consumption, processing delay, and task loss probability. By formulating the problem as a Markov decision process (MDP), a DQN-based algorithm was designed for deriving the optimal offloading policy. In reference [22], a double dueling DQN-based algorithm was proposed to enable dynamic orchestration of networking, caching, and computing resources. It improved the performance of applications but did not consider the energy efficiency issue. Based on CNN and RL, reference [23] presented an offloading scheme for an individual IoT device with energy harvesting, which aimed to select edge server and offloading rate according to current battery level, previous radio transmission rate, and predicted harvested energy. Simulation results showed that the scheme can reduce the energy consumption, computation latency, and task drop rate. Similar to reference [23], which only considered a single device, reference [24] added a task queue state model to dynamic statistics as well as a channel qualities model and an energy queue state model. A double DQN-based computation offloading algorithm was proposed for a single user device as well. Considering a joint optimization problem, a parallel DNN model was designed in reference [25] to make binary offloading decisions and bandwidth allocation decisions. For minimizing the utility calculated by energy consumption and task completion delay, the algorithm only considered the different computation amount of tasks in several devices without the dynamic wireless channel state. However, none of the above works considered the impact of different server workload on task scheduling. Particularly, load balancing is a crucial issue to improve the resource utilization of servers. To the best of our knowledge, all of the methods mainly designed for the load balancing of servers are not suitable for task scheduling in MFC network. Motivated by all the above studies, we design an intelligent adaptive algorithm for MFC networks with multiple servers, which focuses on the joint optimization objective of improving the quality of service, relieving the restrained resource of user device, and balancing the workload of participant server.

3. The Combinatorial Problem of Load Balancing and Task Scheduling

In this section, we first elaborate a MFC system consisting several servers. Then, the local computing model and scheduled computing model are introduced. Finally, we formulate the combinatorial problem related to load balancing and task scheduling. For ease of reference, the main notations and corresponding semantics are presented in Table 1.

3.1. MFC System Model. In this paper, we consider a typical MFC system with multiple servers as shown in Figure 1.

TABLE 1: Main notations and corresponding explanations of the system model.

Notation	Explanation
F	The number of fog servers
C	The number of cloud servers
t	The period duration of equal length time slots in a quasi-static system
τ	The index of time period
T	The number of periods
b_τ	The initial task data size in bit at the beginning of period τ
w_τ	The task computation workload of CPU in cycle per bit
r_τ	The task arrival rate in bit per second
λ_τ	The mean rate of task arrival rate in bit per second
d_τ	The scheduled decision
f^l	The available computation capacity of end device in cycle per second
f_τ^l	The operating computation frequency of end device in cycle per second
P_τ^l	The computing operating power on the user device
r_τ^l	The task computing rate on the user device
G_τ	The ever-changing channel gain
N_τ	The channel noise
P_τ^t	The transmitting operating power on the user device
r_τ^t	The transmitting rate of scheduled task
r_τ^f	The task computing rate on the fog server
b_τ^l	The processed data size in local computing mode during period τ
b_τ^t	The processed data size in scheduled computing mode during period τ
$b_{\tau+1}$	The remaining data size at the end of period τ
P_τ	The operating power on user device during period τ
$L_{\tau,k}$	The appended workload on server k by scheduled task
C_τ	The weighted sum cost of optimization problem
Ω_i	The weighting factors for adjusting the trade-off among the three factors

The fog layer consists of a number of fog servers and the cloud layer consists of several cloud servers, which are denoted by $fs \in \{1, 2, \dots, F\}$ and $cs \in \{1, 2, \dots, C\}$, respectively. We assume that fog servers and cloud servers are connected via optical fiber so that we ignore the latency caused by transmission between these servers. With the service migration technology, all servers can execute tasks of services and send the results back to users. For easy analysis, we consider a quasi-static system in which the system time is divided into equal-length time slots termed as period t . In detail, the environment of network remains unchanged during a period, while it could change across contiguous periods. With the SDN technology, a centralized controller is deployed in the

fog layer so that the task scheduled decisions can be generated sequentially at the beginning of each period. Specifically, the index of time period is denoted as $\tau \in \mathcal{T} = \{0, 1, \dots, T\}$. Particularly, we focus on one dynamic service in one single user, which consists of consecutive tasks and is characterized by three-tuple of parameters (b_τ, w_τ, r_τ) . Therein, b_τ denotes the initial task data size in bit at the beginning of period τ . Besides, w_τ denotes the computation workload consumption in CPU cycles per bit, which depends on the nature of the task and can be measured through experiments. Considered the dynamicity, we use r_τ to denote the task arrival rate in bit per second. And it is independent and identically distributed over periods with mean rate λ_τ . As for scheduled decision d in period τ , $d_\tau = 0$ means that the task is executed locally, and $d_\tau = k$ means that the task is scheduled to server k ($\forall k \in \{1, 2, \dots, F + C\}$).

3.2. Local Computing Model. If the scheduled decision $d_\tau = 0$, the computational task of current period will be executed on end user locally at current period. We denote the available computation capacity of end device as f^l , which is measured by the processor computation clock frequency in the CPU cycles per second. And the operating computation frequency f_τ^l cannot exceed the available computation frequency f^l , which is constrained by $f_\tau^l < f^l$. Besides, we denote the computation energy efficiency coefficient as e_τ^l , which is depended on the chip architecture. Thereby, the computing power consumption on the user device can be estimated by the frequency f_τ^l and the coefficient e_τ^l , which can be expressed as follows,

$$P_\tau^l = e_\tau^l * f_\tau^{l^3}. \quad (1)$$

Based on the dynamic voltage and frequency scale (DVFS) method, the operating frequency f_τ^l can be controlled by the chip operating power P_τ^l so that we can adjust the computing rate of the task. With the parameter w_τ , the task computing rate on local device can be given by

$$r_\tau^l = f_\tau^l / w_\tau = (P_\tau^l / e_\tau^l)^{1/3} / w_\tau. \quad (2)$$

3.3. Scheduled Computing Model. If the scheduled decision $d_\tau = k$ ($k \in \{1, 2, \dots, F + C\}$), the computational task of user device in current period will be scheduled to fog servers or cloud servers. Considering a dynamic system, the scheduled decision should be generated by some factors including the workloads of participant servers, the feature of arrival task, and the condition of the current network. In our proposed MFC system, we consider the time consumption of uploading data and ignore the time consumption of downloading data, since the data size of computing feedback results is small enough. Without loss of generality, the task is scheduled through a dynamic wireless channel where we denote the ever-changing channel gain as G_τ and the channel noise as N_τ . And the scheduling rate can be formulated by the Shannon Hartley equation $C = W \log_2(1 + S/N)$. Obviously, the bigger the uploading transmitting signal

power S is and the available uplink bandwidth W is, the larger the maximum information transmitting rate C is. Thus, if we denote the transmitting operating power on the user device as P_τ^t , the transmitting rate r_τ^t of scheduled task can be given by

$$r_\tau^t = W * \log_2(1 + P_\tau^t * G_\tau / N_\tau). \quad (3)$$

Similar to the local computing model, we denote the available computation capacity of the fog server as f_τ^s . Based on the relationship between the operating frequency of processor computation clock and computing rate, the task computing rate r_τ^f on fog server can be expressed as

$$r_\tau^f = f_\tau^s / w_\tau. \quad (4)$$

Due to the sufficient computational capability of cloud server with high-frequency multiple core, we ignore the time consumption at cloud servers for executing the scheduled tasks. While there is a long distance between the cloud and the edge, the transmission latency is considered in this scheduled computing model. Hence, we denote the time consumption as t_τ^c .

3.4. Load Balancing and Task Scheduling Problem Formulation. In this paper, we focus on the load balancing and task scheduling problem from three aspects, including improving the quality of service, relieving the restrained resource of user device, and balancing the workload of the participant server. Firstly, it is crucial to reduce the data size of the remaining task for improving the quality of service. With the parameters of computing rate r_τ^l and transmitting rate r_τ^t mentioned previously, the processed data size in local computing mode and in scheduled computing mode during period τ can be expressed, respectively, as follows,

$$b_\tau^l = r_\tau^l * t, \quad (5)$$

$$b_\tau^t = r_\tau^t * t. \quad (6)$$

As we denote the initial data size as b_τ at the beginning of period τ , the remaining data size $b_{\tau+1}$ at the end of period is expressed as

$$b_{\tau+1} = \begin{cases} \max(b_\tau + r_\tau * t - b_\tau^l, 0), & \text{if } d_\tau = 0 \\ \max(b_\tau + r_\tau * t - b_\tau^t, 0), & \text{if } d_\tau > 0 \end{cases}. \quad (7)$$

Secondly, the power consumption plays an important role in relieving the restrained resource of the user device. With the parameters of computing operating power P_τ^l and transmitting operating power P_τ^t mentioned previously, the operating power on user device during period τ is given by

$$P_\tau = \begin{cases} P_\tau^l, & \text{if } d_\tau = 0 \\ P_\tau^t, & \text{if } d_\tau > 0 \end{cases}. \quad (8)$$

Thirdly, the appended workload caused by scheduled task has a nonnegligible influence on balancing the workload of servers from the system point of view. While different servers have different capacity and the workload of them varies across period τ , we denote the appended workload on server k by scheduled task as $L_{\tau,k}$. Therefore, we mathematically formulate the combinatorial optimization problem as follows,

$$\begin{aligned} Op1 = \min_{d,p} & C_{\tau}(b_{\tau}, r_{\tau}, G_{\tau}/N_{\tau}, d_{\tau}, P_{\tau}) \\ \text{s.t.} & C1 : C_{\tau} = \Omega_1 * b_{\tau+1} + \Omega_2 * P_{\tau} + \Omega_3 * L_{\tau,k} \\ & C2 : d_{\tau} = k \ (k \in \{0, 1, 2, \dots, F + C\}) \\ & C3 : P_{\tau} < P_{\max}. \end{aligned} \quad (9)$$

The objective is to minimize the weighted sum cost C_{τ} including the data size $b_{\tau+1}$ of the remaining task, the power consumption P_{τ} of the user device, and the appended workload L_{τ} of participant server. The weight coefficient $\Omega_i (i \in \{1, 2, 3\})$ are used for adjusting the trade-off among the three factors referred above. Thus, we need to design the proper algorithm to find optimal solutions to this problem. It is worth noting that the proposed combinatorial problem is NP-hard as it belongs to the mixed integer nonlinear programming (MINLP) problem.

Furthermore, it is difficult to minimize the cost model in a long term when we consider uncertainty about the dynamic MFC environment. The decision derived in the current period can affect the future environment states including the remaining data size, the wireless channel condition, and the participant server workload, thereby affecting the decisions at subsequent time periods. In such a situation, we need to make the scheduled decision in the current period τ by considering the long-term reward. Thus, we utilize discrete-time Markov Decision Process (MDP) to model and analyse the sequential decision-making processes. Correspondingly, we assume one MDP process covers several decision-making periods, and we term one process as an episode indexed by e . Therefore, we formulate another optimization problem as

$$\begin{aligned} Op2 = \min_{d,p} & E \left(\sum_{\tau=1}^T C_{\tau}(b_{\tau}, r_{\tau}, G_{\tau}/N_{\tau}, d_{\tau}, P_{\tau}) \right) \\ \text{s.t.} & C1 : C_{\tau} = \Omega_1 * b_{\tau+1} + \Omega_2 * P_{\tau} + \Omega_3 * L_{\tau,k} \\ & C2 : d_{\tau} = k \ (k \in \{0, 1, 2, \dots, F + C\}) \\ & C3 : P_{\tau} < P_{\max}. \end{aligned} \quad (10)$$

In a word, this problem would be really complicated if we want to achieve fine-grained management. In order to relieve heavy problem complexity and generate adaptive strategies, we design an intelligent adaptive algorithm based on ML methods.

4. The Proposed Intelligent Adaptive Algorithm

In this section, we propose an intelligent adaptive algorithm (IAA) to generate strategies to achieve load balancing of collaborative servers and dynamic scheduling of sequential tasks. As the formulated problem is the MDP issue, the algorithm is designed based on the online learning framework of RL, which can minimize the long-term weighted sum cost. Besides, we apply the DRL method instead of the conventional RL method, since the space of state information is large. Therein, we denote the user device as an agent and the MFC network as an environment.

4.1. State Information. Although we have no prior knowledge of the precise information about the feature of the arrival task and the condition of wireless channel, there are some perceptive information which are crucial to solve the problem. And we apply the DRL method instead of the conventional RL method since the space of crucial state information is large. At the beginning of period τ , the user can observe the initial data size b_{τ} of residual task from the last period. During period τ , although the arrival rate of the task is changing constantly and hard to measure, we can estimate the mean arrival rate λ_{τ} . As the wireless channel conditions between sequential periods are highly interrelated, we collect the information in the previous period $\tau - 1$ to estimate the information in current period τ , including channel gain G_{τ} and the channel noise N_{τ} . For ease of description, we use a list with three elements to denote the state information of period τ as follows,

$$S_{\tau} = [b_{\tau}, \lambda_{\tau}, G_{\tau}/N_{\tau}]. \quad (11)$$

4.2. Action Information. Referring to the perceptive state information S_{τ} , the algorithm will generate corresponding strategies including the scheduled decision and the operating power. Specifically, the algorithm inputs the state list into the policy function and outputs the value which represents the information of strategy. In our algorithm, we apply the DNN method to approximate the policy function. After some calculated process, we can obtain the scheduled decision d_{τ} . The computational task will be executed locally on end user at the current period if the scheduled decision $d_{\tau} = 0$. And the computational task will be scheduled to fog servers or cloud servers if the scheduled decision $d_{\tau} = k$. Besides, we also explore the optional operating power for task controlling. In order to relieve the algorithm complexity, the proposed algorithm applies the DQN method which belongs to DRL. Therein, the operating power is constrained by several discrete values denoted by $P_{\tau} \in \{P_{\max}/L, 2P_{\max}/L, \dots, P_{\max}\}$. As a result, we use a list with two elements to denote the action information of period τ as follows,

$$A_{\tau} = [d_{\tau}, P_{\tau}]. \quad (12)$$

4.3. Reward Information. In the DRL-based framework, we need to criticize the strategies derived from every period. In order to learn a better intelligent adaptive strategy, we should connect the reward information to the data size of the

remaining task, power consumption of user device, and appended workload of participant server. Thus, the reward in each period can be calculated by the cost model which is mentioned in section 3.4 and expressed in equation (9). As a result, the reward function R_τ generated after performing action A_τ in period τ is given by

$$R_\tau = -C_\tau = -\Omega_1 * b_{\tau+1} - \Omega_2 * P_\tau - \Omega_3 * L_{\tau,k}. \quad (13)$$

Besides, as the action derived in the current period can affect the future states, we add a discount factor denoted by γ to express the reward degraded influence. Thus, the long-term reward is given by

$$R = E \left[\sum_{\tau=1}^T \gamma^{\tau-1} * R_\tau \right]. \quad (14)$$

4.4. Algorithm Learning Iteration. As shown in Figure 2, the proposed intelligent adaptive algorithm is designed based on the RL framework and DNN model. The agent is constantly interacted with the environment to learn better strategies online by training the DNN based function $f_{\theta_\tau}(S_\tau, A_\tau)$. The parameter θ_τ is denoted the weight and bias parameters of DNN in period τ . For improving performance of convergence, we divide system time to sequential MDP processes consisted with a series of periods. Specifically, each of the MDP processes is denoted as an episode indexed by e , and each iteration process is denoted as a period indexed by τ . During each episode, the learning iteration is started with a random initial state, and the total number of iterated times is a constant value denoted by T . In each period τ , the algorithm will store the most recent experience denoted by tuple $(S_\tau, A_\tau, R_\tau, S_{\tau+1})$ into the experience memory set $\mathcal{M}_\tau$ with a limited size $|\mathcal{M}_\tau|$. When the number of restored tuples reach the size of memory set, the oldest tuple will be popped when the new tuple will arrive. With a learning rate parameter denoted by α , the Bellman optimality equation for the state action quality function $Q_{\theta_\tau}(S_\tau, A_\tau)$ can be expressed as

$$Q_{\theta_{\tau+1}}(S_\tau, A_\tau) = Q_{\theta_\tau}(S_\tau, A_\tau) + \alpha[R_\tau + \gamma * \max_{A_{\tau+1}} Q_{\theta_\tau}(S_{\tau+1}, A_{\tau+1}) - Q_{\theta_\tau}(S_\tau, A_\tau)]. \quad (15)$$

In order to enhance the training efficiency, the DNN-based function is not trained in every period. Thus, the DNN will be trained once every several periods, and we call this period as a training period if the training process is executed in this period. And the training interval is denoted as δ . In every training period, it will cause large algorithm complexity if we use entire tuples in experience memory set $\mathcal{M}_\tau$. So we apply the experience sampling method for reducing the complexity. In detail, we randomly extract a mini-batch of experience tuples from the experience memory set $\mathcal{M}_\tau$ and store them into the empty replay buffer set $\mathcal{B}_\tau$ with a small size $|\mathcal{B}_\tau|$. The time frame index of sampled tuple is denoted by b , and the set of these indexes is denoted by β . Subsequently, the extracted tuples are used to train the

DNN-based function $f_{\theta_\tau}(S_\tau, A_\tau)$ by minimizing the cross-entropy loss with an optimizer function denoted as $f_{opt}(\theta_\tau)$. As a result, the parameters of the DNN-based function are updated from θ_τ to $\theta_{\tau+1}$ based on a loss function $L(\theta_\tau)$ which is expressed as

$$L(\theta_\tau) = \frac{-1}{|\mathcal{B}_\tau|} \sum_{\tau \in \beta} (R_\tau + \gamma * \max_{A_{\tau+1}} Q_{\theta_\tau}(S_{\tau+1}, A_{\tau+1}) - Q_{\theta_\tau}(S_\tau, A_\tau)). \quad (16)$$

In this way, the algorithm will be gradually learned with the increasing of training period. As for the learning framework, there are two stages including training stage and testing stage. It is noted that the proposed algorithm can be trained previously at the control centre of the MFC network, which can collect the perceptive state information S_τ including the data size b_τ of the remaining task, the mean rate λ_τ of arrival task, and the ratio G_τ/N_τ of signal to noise. Then, the algorithm model can be sent to the user device, and it can achieve fast deployment and be tested online with the dynamic information and preset parameters.

5. The Evaluation of Proposed Algorithm

In this section, we conduct some simulation experiments to evaluate the effectiveness of our proposed algorithm. The experiment parameters setup is presented in detail firstly. According to different scenarios, we evaluate the performance of the proposed DQN-based IAA under variables of task arrival rate, wireless channel condition, server workload degree, and weight coefficients set. Compared with candidate algorithms, the simulation results show that the proposed intelligent adaptive algorithm can achieve superior performance in terms of weighted reward, service latency, and power consumption.

5.1. Experiment Parameters Setup. On a computer with Intel Quad Core i5-4590 CPU @ 3.3 GHz and 4 GB RAM, we simulate our algorithm by Python 3.6 and Tensorflow 1.13.0 library. We set the experimental parameters referring to the related papers [21–25]. In detail, major parameters are described as follows.

For the system model, the number of fog servers F is set to 2 and the number of cloud server cluster C is set to 1. The period duration t of equal length time slots is set to 1 s. For the task model, the initial data size b_τ at the beginning of the period is set to a random integer value which is smaller than 50 Mbit. The task computation workload w_τ of CPU is set to 500 cycle/bit, i.e., 4000 cycle/byte. The mean arrival rate λ_τ of the task is set to 2 Mbps. For the computing model, the available computation capacity f^l of the end device is set to 1.26 GHz. And the maximized operating power on the user device is set to 2 W. For the wireless channel model, we apply a Gaussian Markov block fading autoregressive model. And the wireless channel gain G_τ caused by path loss is set to -30 dB and the channel noise N_τ is set to 10^{-9} W. For the weighted cost model, the weight coefficients for adjusting the trade-off among the three factors are initially set to

FIGURE 2: The framework of the proposed intelligent adaptive algorithm.

(1,6,3), which are mapping to the data size b_{t+1} of the remaining task, the power consumption P_t of the user device, and the appended workload L_t of participant server, respectively.

In the DRL-based algorithm, the number T of periods during one episode is set to 100. The number of episodes for the training stage is set to 800, and the number of episodes for the test stage is set to 200. The number L of discrete optional values for the operating power P_t is set to 5. The learning rate α is set to 0.001, and the discount factor γ of reward degraded influence is set to 0.99. The experience memory size is set to 20000, and the replay buffer size is set to 128. For the DNN model, we designed it by a four-layer fully connected neural network with two hidden layers which consist of 200 neurons and 300 neurons, respectively. As for model training, we apply Adam and RMSprop as the optimizer function $f_{opt}(\theta_t)$.

5.2. Performance Impact of Task Arrival Rate. In this subsection, we investigate the performance impact of the task arrival rate and illustrate the numerical results derived under different scenarios. Specifically, the task arrival rate is set to 1, 2, and 3 Mbps, respectively. The distances between the user device and three servers are all set to 100 m. The workload degrees of all servers are set to 1. The tuple of weight coefficients $(\Omega_1, \Omega_2, \Omega_3)$ is set to (1, 6, 3). We calculate the average values in all the periods of one episode, including weighted reward, data size of the remaining task, and power consumption of user device. With respect to episode, the corresponding learning curves are plotted in Figures 3, 4, and 5, respectively. And the smoothing values are calculated by a window of 10 episodes, which are plotted as semitransparent curves. In Figure 3, it can be observed that the weighted reward is increasing as episode index increases, which indicates the strategy can be learned gradually. And the learning effect of curve which represents task arrival rate is 2 Mbps

and is more notable under our setting parameters. In Figure 4, it can be observed that the average data size of the remaining task is decreasing as episode index increases, which indicates the effectiveness of the learning algorithm. In Figure 5, the learning curve of user device power converges to different values, which indicates that the algorithm will get the optimal operating power according to the environment information and the scheduling decision after some episodes of learning.

5.3. Performance Impact of Wireless Channel Condition. In this subsection, we investigate the performance impact of wireless channel condition and illustrate the numerical results derived under different scenarios. Specifically, the task arrival rate is set to 2 Mbps. The distances between the user device and three servers are set to 100 m, 150 m, and 200 m, respectively. The workload degrees of all servers are set to 1. The tuple of weight coefficients $(\Omega_1, \Omega_2, \Omega_3)$ is set to (1, 6, 3). We calculate the average values in all the periods of one episode, including weighted reward, data size of the remaining task, and power consumption of user device. With respect to episode, the corresponding learning curves are plotted in Figures 6, 7, and 8, respectively. And the smoothing values are calculated by a window of 10 episodes, which are plotted as semitransparent curves. The results show that the weighted reward and remaining data size have a high relevancy with different wireless channel condition.

5.4. Performance Impact of Server Workload Degree. In this subsection, we investigate the performance impact of the server workload degree and illustrate the numerical results derived from different scenarios. Specifically, the task arrival rate is set to 2 Mbps. All of the distances between the user device and three servers are set to 100 m. The server workload degrees are set to 1, 10, and 30, respectively. The tuple

FIGURE 3: Learning curve of average weighted reward with respect to episode.

FIGURE 4: Learning curve of average data size of remaining task with respect to episode.

of weight coefficients ($\Omega_1, \Omega_2, \Omega_3$) is set to (1, 6, 3). We calculate the average values in all the periods of one episode, including weighted reward, data size of the remaining task, and power consumption of user device. With respect to

episode, the corresponding learning curves are plotted in Figures 9, 10, and 11, respectively. And the smoothing values are calculated by a window of 10 episodes, which are plotted as semitransparent curves. The results show that

FIGURE 5: Learning curve of average power consumption of user device with respect to episode.

FIGURE 6: Learning curve of average weighted reward with respect to episode.

the weighted reward varies a lot at the beginning, and they converge to similar values at last. This is because the algorithm will avoid to derive the strategy of scheduling tasks to the server with heavy load after some learning episode, thus achieving the effect of server load balancing.

5.5. Performance Impact of Weight Coefficients. In this subsection, we investigate the performance impact of weight coefficients and illustrate the numerical results derived from different scenario. Specifically, the task arrival rate is set to 2 Mbps. The distances between the user device and three

FIGURE 7: Learning curve of average data size of remaining task with respect to episode.

FIGURE 8: Learning curve of average power consumption of user device with respect to episode.

servers are all set to 100 m. The workload degrees of all servers are set to 10. The first tuple of weight coefficients $(\Omega_1, \Omega_2, \Omega_3)$ is set to (1, 6, 3). The second tuple of weight coefficients $(\Omega_1, \Omega_2, \Omega_3)$ is set to (1, 0, 3). The third tuple of weight coefficients $(\Omega_1, \Omega_2, \Omega_3)$ is set to (1, 6, 0). We calcu-

late the average values in all the periods of one episode, including weighted reward, data size of the remaining task, and power consumption of user device. With respect to episode, the corresponding learning curves of are plotted in Figures 12, 13, and 14, respectively. And the smoothing

FIGURE 9: Learning curve of average weighted reward with respect to episode.

FIGURE 10: Learning curve of average data size of remaining task with respect to episode.

values are calculated by a window of 10 episodes, which are plotted as semitransparent curves. The results show that the values of these curves are similar, because, although the weight coefficient of the remaining data size is small, its larger value still has a greater impact on the weighted cost model.

5.6. Performance of Proposed Algorithm Compared with Candidate Algorithms. In this subsection, we investigate the performance of the proposed algorithm compared with the candidate algorithms and illustrate the numerical results derived from one typical scenario. Specifically, the task

FIGURE 11: Learning curve of average power consumption of user device with respect to episode.

FIGURE 12: Learning curve of average weighted reward with respect to episode.

arrival rate is set to 2 Mbps. The distances between the user device and three servers are set to 100 m, 150 m, and 200 m, respectively. The server workload degree is all set to 1. The tuple of weight coefficients $(\Omega_1, \Omega_2, \Omega_3)$ is set to (1, 6, 3). We calculate the average values in all the periods of one epi-

sode, including weighted reward, data size of the remaining task, and power consumption of user device. With respect to episode, the corresponding learning curves are plotted in Figures 15, 16, and 17, respectively. In the entire local computing algorithm (ELCA), the tasks will be local processed

FIGURE 13: Learning curve of average data size of remaining task with respect to episode.

FIGURE 14: Learning curve of average power consumption of user device with respect to episode.

entirely with the maximum power. Similarly, the tasks will be offloaded entirely with the maximum power in the entire scheduling computing algorithm (ESCA). The results show

that the proposed intelligent adaptive algorithm can achieve superior performance in terms of weighted reward, service latency, and power consumption.

FIGURE 15: The curve of average weighted reward with respect to episode.

FIGURE 16: The curve of average data size of remaining task with respect to episode.

FIGURE 17: The curve of average power consumption of user device with respect to episode.

6. Conclusions and Future Work

This paper formulates a combinatorial problem related to load balancing of collaborative servers and dynamic scheduling of sequential tasks in a cloud-aware MFC network. The optimization objective is to improve the quality of service, relieve the restrained resource of the user device, and balance the workload of the participant server. Besides, an intelligent adaptive algorithm is proposed based on the DRL method to find proper strategies, which takes the data size of the remaining task, the power consumption of the user device, and the appended workload of participant server into consideration. According to different scenarios, we evaluate the performance of the proposed algorithm under variables of task arrival rate, wireless channel condition, server workload degree, and weight coefficients. Compared with candidate algorithms, the simulation results show that the proposed intelligent adaptive algorithm can achieve superior performance in terms of weighted reward, service latency, and power consumption. In future work, we will further research other problems in the cloud-aware MFC network, which is related to security and applicability [26–28].

Data Availability

The related data and parameters of experimental simulation are referred to previously published researches, which have been cited in references.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2018YFA0701604).

References

- [1] H. Hui, C. Zhou, S. Xu, and F. Lin, "A novel secure data transmission scheme in industrial internet of things," *China Communications*, vol. 17, no. 1, pp. 73–88, 2020.
- [2] C. Gong, F. Lin, X. Gong, and Y. Lu, "Intelligent cooperative edge computing in the internet of things," *IEEE Internet of Things Journal*, 2020.
- [3] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. YANG, and W. Wang, "A survey on mobile edge networks: convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.
- [4] M. Li, Q. Wu, J. Zhu, R. Zheng, and M. Zhang, "A computing offloading game for mobile devices and edge cloud servers," *Wireless Communications and Mobile Computing*, vol. 2018, 10 pages, 2018.
- [5] T. Shuminoski, S. Kitanov, and T. Janevski, "Advanced QoS provisioning and mobile fog computing for 5G," *Wireless Communications and Mobile Computing*, vol. 2018, 13 pages, 2018.

- [6] S. Kim, "Novel resource allocation algorithms for the social internet of things based fog computing paradigm," *Wireless Communications and Mobile Computing*, vol. 2019, 11 pages, 2019.
- [7] Y. Chen, Y. Zhang, and X. Chen, "Dynamic service request scheduling for mobile edge computing systems," *Wireless Communications and Mobile Computing*, vol. 2018, 10 pages, 2018.
- [8] D. Thembelihle, M. Rossi, and D. Munaretto, "Softwarization of mobile network functions towards agile and energy efficient 5G architectures: a survey," *Wireless Communications and Mobile Computing*, vol. 2017, 21 pages, 2017.
- [9] F. Song, M. Zhu, Y. Zhou, I. You, and H. Zhang, "Smart collaborative tracking for ubiquitous power IoT in edge-cloud interplay domain," *IEEE Internet of Things Journal*, 2019.
- [10] Z. Ai, Y. Liu, F. Song, and H. Zhang, "A smart collaborative charging algorithm for mobile power distribution in 5G networks," *IEEE Access*, vol. 6, pp. 28668–28679, 2018.
- [11] M. Zhang, B. Hao, F. Song, M. Yang, J. Zhu, and Q. Wu, "Smart collaborative video caching for energy efficiency in cognitive content centric networks," *Journal of Network and Computer Applications*, vol. 158, p. 102587, 2020.
- [12] F. Song, Y.-T. Zhou, L. Chang, and H.-K. Zhang, "Modeling space-terrestrial integrated networks with smart collaborative theory," *IEEE Network*, vol. 33, no. 1, pp. 51–57, 2019.
- [13] F. Song, Z. Ai, Y. Zhou, I. You, K.-K. R. Choo, and H. Zhang, "Smart collaborative automation for receive buffer control in multipath industrial networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1385–1394, 2020.
- [14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [15] L. Deng and D. Yu, "Deep learning: methods and applications," *Foundations & Trends in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2013.
- [16] R. S. Sutton and A. G. Barto, "Reinforcement learning: an introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1054–1054, 1998.
- [17] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [18] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.
- [19] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: training deep neural networks for interference management," *IEEE Transactions on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.
- [20] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs," in *2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017.
- [21] B. Huang, Y. Li, Z. Li et al., "Security and cost-aware computation offloading via deep reinforcement learning in mobile edge computing," *Wireless Communications and Mobile Computing*, vol. 2019, 20 pages, 2019.
- [22] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: a big data deep reinforcement learning approach," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 31–37, 2017.
- [23] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, 2019.
- [24] 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.
- [25] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Networks and Applications*, 2018.
- [26] Z. Ai, Y. Liu, L. Chang, F. Lin, and F. Song, "A smart collaborative authentication framework for multi-dimensional fine-grained control," *IEEE Access*, vol. 8, pp. 8101–8113, 2020.
- [27] F. Song, Y.-T. Zhou, Y. Wang, T.-M. Zhao, I. You, and H.-K. Zhang, "Smart collaborative distribution for privacy enhancement in moving target defense," *Information Sciences*, vol. 479, pp. 593–606, 2019.
- [28] Z.-Y. Ai, Y.-T. Zhou, and F. Song, "A smart collaborative routing protocol for reliable data diffusion in IoT scenarios," *Sensors*, vol. 18, no. 6, p. 1926, 2018.