Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 1427219, 11 pages
https://doi.org/10.1155/2022/1427219

Research Article

Task Offloading and Resource Allocation Strategy Based on Deep Learning for Mobile Edge Computing

Zijia Yu, Xu Xu, and Wei Zhou

School of Information Engineering, Suzhou University, Suzhou, Anhui 234000, China

Correspondence should be addressed to Zijia Yu; yuzj@ahszu.edu.cn

Received 14 June 2022; Revised 2 August 2022; Accepted 13 August 2022; Published 31 August 2022

Academic Editor: Le Sun

For the problems of unreasonable computation offloading and uneven resource allocation in Mobile Edge Computing (MEC), this paper proposes a task offloading and resource allocation strategy based on deep learning for MEC. Firstly, in the multiuser multiserver MEC environment, a new objective function is designed by combining calculation model and communication model in the system, which can shorten the completion time of all computing tasks and minimize the energy consumption of all terminal devices under delay constraints. Then, based on the multiagent reinforcement learning system, system benefits and resource consumption are designed as rewards and losses in deep reinforcement learning. Dueling-DQN algorithm is used to solve the system problem model for obtaining resource allocation method with the highest reward. Finally, the experimental results show that when the learning rate is 0.001 and discount factor is 0.90, the performance of proposed strategy is the best. Furthermore, the proportions of reducing energy consumption and shortening completion time are 52.18% and 34.72%, respectively, which are better than other comparison strategies in terms of calculation amount and energy saving.

1. Introduction

With the rise of computing-intensive applications and explosive growth of data traffic, users' requirements for the computing power and service quality of mobile devices are also increasing [1]. At present, cloud computing also faces many problems and challenges. Due to its resource-intensive architecture, mobile cloud computing imposes a huge additional load on the backhaul link of mobile networks [2, 3]. Thus, Mobile Edge Computing (MEC) technology is proposed, which physically integrates computing and storage resources into the edge of mobile network architecture [4, 5]. This not only effectively reduces the transmission delay but also solves the problems of high load and high delay caused by mobile cloud computing [6]. At the same time, MEC has the characteristics of distributed architecture, being at the edge of network, low latency, user location awareness, and network status awareness [7, 8]. However, deploying a large number of computing and storage devices at the edge of network for users to choose and accessing neighboring service providers for edge computing will bring a series of complexities such as access and resource allocation strategy selection, user mobility management, and computing task migration problems [9].

In order to achieve the goal of short completion time and lower terminal energy consumption under delay constraint, this paper proposes a task offloading and resource allocation strategy based on deep learning for MEC. In order to shorten the completion time of computing tasks and minimize the energy consumption of all terminal devices while satisfying delay constraints, the proposed strategy is designed in a multiuser multiserver MEC environment, combined with computing model and communication model in system. Moreover, a new objective function is designed, which uses objective optimization to further reduce energy consumption and time delay. It uses Dueling-DQN algorithm to solve the optimization model to shorten completion time and minimize energy consumption of all terminal devices while meeting the delay constraints.

The remaining chapters of this paper are arranged as follows: Section 2 introduces the relevant research work on mobile task unloading. Section 3 introduces the system model. Section 4 introduces the new computing offload method based on improved DQN. In Section 5, simulation experiments are designed to verify the performance of the proposed model. Section 6 is the conclusion.

2. Related Work

In MEC network, computation offloading may occur in three types: full offloading, partial offloading, and local processing [10]. An important research hotspot in the field of computation offloading is computation offloading decisions. Generally speaking, the offloading goal mainly focuses on minimizing the overall delay or minimizing energy consumption for user devices while meeting the minimum delay requirements [11]. Reference [12] proposed a distributed task offloading strategy for low-load base station groups in MEC environment. It selects the best MEC node offloading amount by game equation on the basis of quantifying offloading cost and delay. But it is not suitable for high-load application environments. In reference to the problem of unbalanced computing resources on the edge server in vehicle edge computing network, [13] proposed a load balancing task offloading scheme based on software-defined network. This solution can effectively reduce delay and improve the efficiency of task offloading processing. However, the processing method used has poor performance, which affects the distribution efficiency. Reference [14] used greedy selection to design a maximum energy-saving priority algorithm to achieve optimal offloading of computing tasks on mobile devices, but it does not consider the delay constraints of task offloading. Reference [15] combined long short-term memory (LSTM) and candidate network set to improve the deep reinforcement learning algorithm and used this algorithm to solve the problem of offloading dependency of multinode and mobile tasks in large-scale heterogeneous MEC. But they ignore the optimal allocation problem of computing resources.

Similar to computation offloading, resource allocation is also one of the core issues in MEC [16]. In MEC network, technologies such as content caching and ultradense deployment are introduced, and multiple resources are deployed to mobile network edge according to the specific needs of users. This can further ensure the quality of service and greatly increase system capacity [17]. However, due to objective reasons such as physical volume and power consumption, the mobile network edge has limited computing resources, storage cache capacity, and spectrum resources. How to allocate multiple resources and improve the system service efficiency has a huge effect on the improvement of MEC network system performance [18]. Reference [19] proposed a time average computation rate maximization (TACRM) algorithm, which allows joint allocation of radio resources and calculation resources. However, the overall performance and task requirements of devices were not considered comprehensively in the allocation process, and the allocation efficiency still needs to be improved. Reference [20] comprehensively considered factors such as CPU, hard disk space, and required time and distance and proposed a comprehensive utility function for MEC resource allocation to achieve the optimal allocation of resources in MEC and cloud computing. However, this function considers many factors, which will seriously affect the efficiency of allocation in real applications. Reference [21] designed a two-layer optimization method for MEC, which uses pruning candidate modes to reduce the number of unfeasible offloading decisions. Through ant colony algorithm to achieve the upper-level optimization, the resource allocation effect is better. However, the server computing resource constraints and task delay constraints are not considered, and the overall timeliness is not good. Reference [22] constructed a low-complexity advanced branch model, which can be used for resource scheduling in large-scale MEC scenarios.

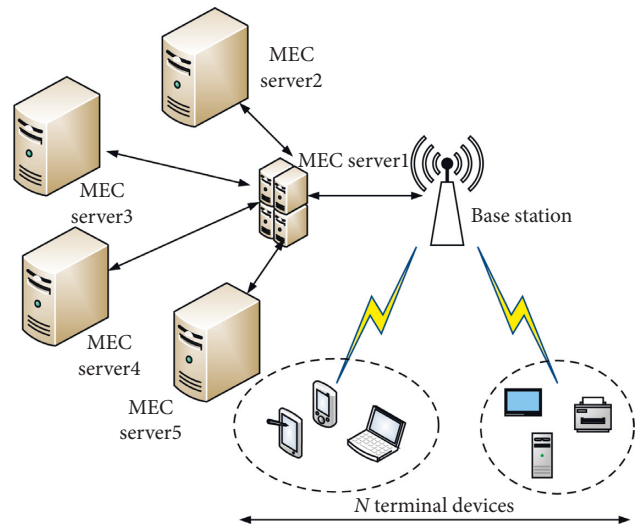

Figure 1: System model diagram.

Due to the lack of powerful processing algorithms, the overall efficiency and performance are not ideal. To this end, comprehensively considering the task offloading and resource allocation problem, a deep learning-based MEC task offloading and resource allocation strategy is proposed to coordinate and optimize the allocation and offloading between computing resources and computing tasks, which improves the comprehensive computing efficiency of MEC.

3. System Model

The system model is a multiuser multiserver application scenario, in which there are N terminal devices and M MEC servers. The base station is used to provide communication resources for user equipment. Each base station is connected to an edge computing server through optical fiber, through the wireless communication link to connect to MEC server to calculate task data of offloading terminal devices, as shown in Figure 1. It is assumed that each terminal device can perform offloading computations or local calculations for its own execution tasks. And when offloading, the task can only be offloaded to one MEC server for calculation, and each terminal device is within the range of wireless

connection. However, the computing power of each MEC server is limited; it cannot accept the offloading request of each terminal at the same time.

The collection of terminal devices is $U = \{1, 2, \ldots, n, \ldots, N\}$, the collection of MEC server is $H = \{1, 2, \ldots, m, \ldots, M\}$, and the collection of all tasks is G. Each terminal device n has a calculation-intensive task G_n to be processed, which specifically includes the data D_n (code and parameters) required for computing task G_n, the CPU workload ϕ_n required for computing task G_n, and the completion time of task G_n. The extension constraint is τ_n, namely, $G_n \cong (D_n, \phi_n, \tau_n)$. The set of offloading decisions for each G_n is $X = [x_1, x_2, \ldots, x_n, \ldots, x_N]$.

When $x_n = \{0, 1, \ldots, m, \ldots, M\}$, and $x_n = 0$ is local offloading, the rest means offloading G_n to m MEC servers.

3.1. Communication Model. In the computation offloading problem, two links are mainly studied: wireless link from terminal devices to MEC and the wired link from MEC to cloud in the core network. In the wireless link, Finite-State Markov Channel (FSMC) model based on fading characteristics is used. FSMC model has a wide range of applications in wireless networks [23, 24].

The channel is divided into nonoverlapping intervals through the division of channel-related parameter ranges, and each interval of selected parameters represents a state in FSMC model. The relevant parameter used in FSMC may be Signal-to-Noise Ratio (SNR) amplitude of received signal at the receiving end or collected energy. SNR can be selected as a parameter that composes SNR model [25]. The SNR of receiving end is divided into K levels, and each level is associated with a state of Markov chain. The block fading channel is considered to be that the SNR of receiving end is a constant within a period of time but will change according to Markov transition probability between different periods. Assume that random variable γ is the SNR of receiving end of terminal device n. That is, γ can be improved according to Markov chain of finite states, and all its states can be expressed as $\kappa = \{1, 2, \ldots, K\}$. The realization of random variable γ of terminal device n in the time period t is represented as $\Gamma(t)$, specifically expressed as

$$\Gamma_n(t) = k, \text{if } \gamma_n(t) \in \left[h^{i-1}, h^k\right), \tag{1}$$

where $k \in \kappa = \{1, 2, \ldots, K\}$ and $h^0 = 0 < h^1 < h^2 < \ldots$. Let $\rho_{s_n' s_n''}(t)$ denote the probability of state $\Gamma_n(t)$ transitioning from state s_n' to state s_n'' in the time period t. The $K \times K$ channel state transition probability matrix of terminal device n is denoted as $\Phi_n(t) = [\rho_{s_n' s_n''}(t)]_{K \times K}$, where $\rho_{s_n' s_n''}(t) = \Pr(\Gamma_n(t+1) = s_n' \mid \Gamma_n(t) = s_n'')$, s_n', and $s_n'' \in \kappa$.

In practical applications, the transfer matrix can be observed and measured from wireless environment in the past. In addition, it is considered that $\{\Gamma_n(t), 1 \le t \le T\}$ exists independently for terminal device n. Based on FSMC channel model, $\Gamma_{n,m}$ is used here to represent SNR between terminal device n and MEC server m. Since there is no interference between terminal devices, its channel efficiency can be expressed as $\vartheta_{n,m} = \log_2(1 + \Gamma_{n,m})$. Considering that the bandwidth W_m of MEC server m is divided into W_m/B_m, the bandwidth of each channel is B_m. Assuming that each user is allocated at most one channel, the transmission rate from terminal device n to MEC server m can be expressed as

$$v_{n,m}(t) = B_m \vartheta_{n,m}(t). \tag{2}$$

The subchannel owned by MEC server m has certain restrictions on receiving W_m/B_m; that is, the bandwidth allocated by MEC server m to all connected users cannot exceed the total bandwidth of MEC server m. Besides, MEC server is limited by cache and computing capacity. On the one hand, MEC server can only handle a limited number of tasks; on the other hand, the load that MEC server can handle is also limited (such as the number of computing tasks). Therefore, some tasks will be further offloaded to the core network to be processed by the core network. Use $g_u(t) \in \{0, 1\}$ to represent the computation offloading decision indicator, which is used to indicate the way the server provides services. Among them, $g_n(t) = 0$ means that the terminal device n is processed by connected MEC server for computing tasks. And $g_n(t) = 1$ indicates that the task will be further offloaded to core network for processing by connected MEC server.

In order to further offload tasks to cloud, the wired backhaul link from MEC server to core network is considered. Assuming that the backhaul link capacity of network is Z (in bits per second), the backhaul link capacity allocated by MEC server m is Z_m. Then, the following restrictions must be met:

$$\begin{aligned} \sum_{n=1}^{N} g_n(t)\theta_{n,m}(t)\varpi_{n,m}(t) &\le Z_m, \\ \sum_{m=1}^{M}\sum_{n=1}^{N} g_n(t)\theta_{n,m}(t)\varpi_{n,m}(t) &\le Z, \end{aligned} \tag{3}$$

where $\theta_{n,m}$ is the connection between terminal device n and MEC server m and $\varpi_{n,m}$ is the transmission rate between terminal device n and MEC server m.

The sum of the rates of offloading computation tasks to the terminal device of core network by MEC server m cannot exceed the backhaul capacity of MEC server m. And the sum of speeds of all terminal devices processing computing tasks in the cloud cannot exceed the total backhaul capacity of system.

3.2. Calculation Model. If G_n is processed locally, use T_n^L to represent the time when G_n is executed locally, which is specifically defined as

$$T_n^L = \frac{\phi_n}{f_n^L}, \tag{4}$$

where workload ϕ_n is the total number of CPU cycles required to complete G_n and f_n^L is the local computing power of terminal device n (i.e., the number of CPU cycles executed per second).

Use E_n^L to represent the energy consumption of devices executed locally by G_n, which is defined as follows:

$$E_n^L = \phi_n \times e_n, \tag{5}$$

where e_n is terminal device n to calculate the energy consumption per unit of CPU cycle, $e_n = (f_n^L)^2 \times 10^{-27}$.

If G_n is processed at the edge, delay T_n^O and device energy consumption E_n^O under G_n edge execution should be calculated from three parts: data upload, data processing, and data return [26]. The specific calculation is as follows.

First, terminal device n uploads data G_n to the corresponding MEC server by wireless channel. Let T_n' be the time when device n uploads G_n data, which is defined as

$$T_n' = \frac{D_n}{v'}, \tag{6}$$

where D_n is the data size of G_n and v' is data upload rate in the system model (i.e., the amount of data uploaded per second).

Then, the energy consumption E_n' of terminal device n uploading data is

$$E_n' = T_n' \times P', \tag{7}$$

where P' is the uplink transmission power of terminal device n.

Then, MEC allocates computing resources for calculation after receiving processed data. Use T_n'' to represent the time when the offloading data is calculated in MEC server, which is defined as

$$T_n'' = \frac{\phi_n}{f_{nm}^O}, \tag{8}$$

where f_{nm}^O are the computing resources allocated by m MEC servers for G_n offload execution (i.e., the number of CPU cycles executed per second). When G_n is unloaded to the local or other MEC server, f_{ij}^O is zero and serves as a constraint in the model, namely,

$$f_{nm}^O = 0, x_n \neq m. \tag{9}$$

At this time, terminal device n has no computing task and is in a waiting state and generates idle energy consumption. Suppose P_n^I is the idle power of terminal device n, then the idle energy consumption E_n'' of terminal device n under offloading computation is

$$E_n'' = T_n'' \times P_n^I. \tag{10}$$

Finally, MEC server returns the calculation result to terminal device n. The calculation result during backhaul is small and downlink rate is high, so the time delay and energy consumption when terminal device is received are ignored. Therefore, delay T_n^O under G_n edge execution is the sum of transmission delay T_n' and the calculation delay T_n'' of MEC server, namely,

$$T_n^O = T_n'' + T_n'. \tag{11}$$

The device energy consumption E_n^O under G_n edge execution is the sum of upload energy consumption E_n'' of device n and the idle energy consumption E_n'' of device n waiting for G_n to complete calculation on MEC server, namely,

$$E_n^O = E_n' + E_n''. \tag{12}$$

In summary, the time delay T_n and energy consumption E_n of the entire calculation process of task G_n in terminal device n are

$$\begin{aligned} T_n &= \begin{cases} T_n^L, & x_n = 0, \\ T_n^O, & x_n \neq 0, \end{cases} \\ E_n &= \begin{cases} E_n^L, & x_n = 0, \\ E_n^O, & x_n \neq 0. \end{cases} \end{aligned} \tag{13}$$

Note that T_n and f_{nm}^O should meet the following restrictions:

$$\begin{aligned} T_n &\leq \eta_n, \\ T_n &\leq \eta_n. \end{aligned} \tag{14}$$

The time delay constraint η_n of G_n is that computing power is twice 1.4 GHz. F_m is the overall computing resources of MEC server m; that is, the sum of computing resources allocated by each G_n that is offloaded to MEC server m should not exceed F_m.

3.3. Problem Model. The purpose of this paper is to jointly optimize offloading decision-making and resource allocation scheme in the multiuser multi-MEC server scenario, considering the limited computing resources and time delay constraint of computing tasks. This allows all computing tasks to shorten the completion time and minimize energy consumption of all terminal devices while meeting the delay constraints and extend the use time of terminal devices [27, 28]. Thus, the system objective function Ψ is defined as

$$\Psi = \sum_{n=1}^{N} E_n + 10 \times \sum_{n=1}^{N} \frac{T_n}{\eta_n}. \tag{15}$$

(T_n/η_n) is the ratio of completion time G_n to the delay constraints. According to the calculation results of simulation experiment, the difference between $\sum_{n=1}^{N} E_n$ and $\sum_{n=1}^{N} (T_n/\eta_n)$ is a decimal order of magnitude. Therefore, to ensure that the two are of the same order of magnitude and optimized together, $\sum_{n=1}^{N} (T_n/\eta_n)$ is multiplied by a factor of 10. The objective function Ψ minimizes the ratio of overall energy consumption of terminal devices to the task execution time and delay constraints by solving the optimal offloading decision and resource allocation plan to achieve research purpose. The overall problem model is as follows:

$$
\begin{aligned}
&\min_{X,f}(\Psi),\\
&X=[x_1,x_2,\ldots,x_n,\ldots,x_N],\\
&X=[x_1,x_2,\ldots,x_n,\ldots,x_N],\\
&y_n=\begin{cases} f_n^L, x_n=0,\\ f_{nm}^O, x_n=m,\end{cases}\\
&\text{s.t.}\\
&C_1\colon x_n\in\{0,1,\ldots,m,\ldots,M\},\forall n\in U,\\
&C_2\colon y_n>0,\forall n\in U,\\
&C_3\colon f_{nm}^O=0, x_n\neq m,\\
&C_4\colon T_n\le\eta_n,\forall n\in U,\\
&C_5\colon \sum_{n=1}^{N} f_{nm}^O\le F_m,\forall m\in H,\\
&C_6\colon \begin{cases} \sum_{n=1}^{N} g_n(t)\theta_{n,m}(t)\varpi_{n,m}(t)\le Z_m,\\ \sum_{m=1}^{M}\sum_{n=1}^{N} g_n(t)\theta_{n,m}(t)\varpi_{n,m}(t)\le Z,\end{cases}
\end{aligned}
\tag{16}
$$

where X is the task offloading decision amount and Y is the calculation resource allocation amount. Constraints C_1, C_2, and C_3 indicate that each task G_n can only be offloaded to the local or one of MEC servers for calculation. C_4 represents the constraint of task completion delay, and C_5 and C_6 represent the constraint that allocated computing resources should meet.

4. New Computation Offloading Method Based on Improved DQN

4.1. Multiagent Reinforcement Learning Algorithm. The multiagent reinforcement learning system is shown in Figure 2, where multiple agents act at the same time. Under the joint action, the entire system will be transferred, and each agent will be rewarded immediately [29, 30].

For multiagent reinforcement learning, it is first necessary to establish a Markov game model. Markov game can be described by a multigroup $(n, S, A_1,\ldots,A_n,R_1,\ldots,R_n)$. Among them,

(1) n is the number of agents; that is, N is the number of terminal devices. S is the system state, which generally refers to the joint state of multiple agents, that is, the joint state of each agent. The terminal device shares the current load status of edge computing servers, which can be expressed as

$$LD(t)=[LD_1(t),LD_2(t),\ldots,LD_m(t)], \tag{17}$$

where LD_m is the load of MEC.

(2) R_i is the instant reward function of each agent. That is, in current state s, after joint action $(A_1,\ldots,A_n)$ taken by multiple agents, the reward is obtained in the next system state $\hat{s}$.

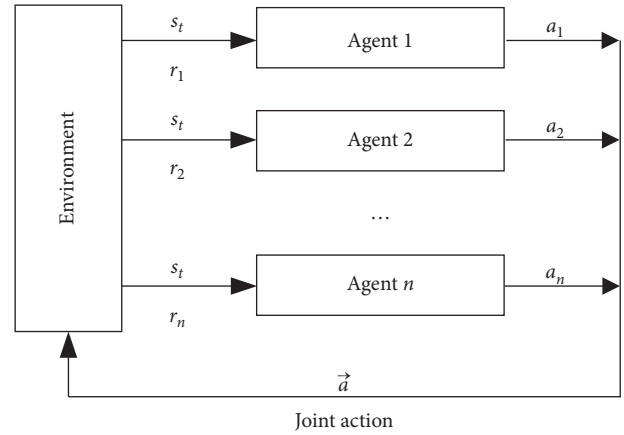

Figure 2: Multiagent reinforcement learning system.

The reward function completely describes the relationship between multiple agents. When the reward function of each agent is the same, that is, $R_1=R_2=\cdots=R_n$, it means that the agent is a complete cooperative relationship. When there are only two agents and reward function is opposite, that is, $R_1=-R_2$, it means that the agents are in perfect competition. When the return function is between the two, it is a mixed relationship between competition and cooperation.

4.2. Problem Description and Modeling

4.2.1. Network Status. $S=\{s(t)\}$ represents the network state space, where $s(t)$ represents the network state at time period t, and improvements are made in the entire time period T. The network status consists of SNR of each terminal device and cache status of each MEC server. $s(t)$ can be defined as

$$
\begin{aligned}
s(t)=\big(&\Gamma_1(t),\ldots,\Gamma_n(t),\ldots,\Gamma_N(t),\\
&\psi_1(t),\ldots,\psi_m(t),\ldots,\psi_M(t)\big),
\end{aligned}
\tag{18}
$$

where $\Gamma_n=\{\Gamma_{n,m},m\in M\}$ represents SNR between user terminal device n and all MECs. $\psi_m(t)=\{\varphi_{k,m},k\in K\}$ represents the cache status of MEC servers.

4.2.2. Network Behavior. The intelligent agent needs to determine the attachment relationship between the terminal device and MEC server in each time period. That is the determination of the terminal device's computing offload, the allocation of computing resources, and the service cache policy of each MEC server. Thus, each executable action of terminal devices in the time period t can be defined as follows:

$$
\begin{aligned}
a(t)=\big(&A_1(t),\ldots,A_n(t),\ldots,A_N(t),\\
&G_1(t),\ldots,G_n(t),\ldots,G_N(t),\\
&\psi_1(t),\ldots,\psi_m(t),\ldots,\psi_M(t)\big),
\end{aligned}
\tag{19}
$$

where $A_n(t)=\{a_{n,m}(t),m\in M\}$ represents the attachment indicator of terminal device n and $G_n(t)$ represents the calculation and offloading decision of terminal device n.

4.2.3. Reward Function. The goal is to maximize total benefit of system, but the reward function should be set to current benefit of system. First calculate the system leased spectrum and backhaul resources and allocate them to terminal devices part of the revenue. The unit price of spectrum leased from MEC server m is set to δ_m per Hz, and the unit price of backhaul link from MEC server m to core network is set to σ_m per bps. Corresponding to this, the calculation data is transmitted to MEC server corresponding to terminal device n and backhaul link from MEC server to the core network is used for charging. The unit price is defined as α_n per Hz and β_n per bps. Therefore, by summarizing this part of the income and expenditure, part of income for leased spectrum and backhaul resources obtained by terminal device n can be obtained:

$$R_n'(t) = \alpha_n \sum_{m=1}^{M} a_{n,m}(t) B_m + \beta_n g_n(t) \sum_{m=1}^{M} a_{n,m}(t) R_{n,m}(t) - \sum_{m=1}^{M} \delta_m a_{n,m}(t) B_m - g_n(t) \sum_{m=1}^{M} \sigma_m a_{n,m}(t) R_{n,m}(t). \tag{20}$$

Then, calculate the profit obtained by terminal devices from allocating computing resources. On the one hand, when MEC side performs computing tasks, it needs to pay communication company for the loss of processing computing tasks and define the unit price of MEC server m energy consumption as χ_m. On the other hand, the terminal device needs to pay a certain price for the server on MEC side, and computing resource allocated for each unit computing task is set to ζ_n. Therefore, the benefit obtained by allocating computing resources to terminal device n can be calculated as

$$R_n''(t) = \left(1 - d_n(t)\right) \sum_{m=1}^{M} \alpha_{n,m} \frac{\zeta_n F_{n,m}(t)}{L_{u_n} - \chi_m E_{n,m}^{MEC,e}(t)}. \tag{21}$$

The amount of computing resources allocated to each unit computing task by the above formula has a very important impact on the completion time of computing task. Thus, the service cache cost mainly includes two parts: the cost of replacing type of cache supported on MEC side, and the cost of caching specific services on MEC server. Define the unit price of replacing cache type on MEC server m as ξ_m for each service type, and the unit price for caching services on MEC server is ξ_m per storage space. In order to increase the benefits of cache, the business type is quantified by weak backhaul from MEC server to the core network, which will be used to measure the cost of users. The benefits obtained by executing the cache service on MEC server m can be expressed as

$$R_m''(t) = \sum_{n=1}^{N} \beta_n \left(1 - g_n(t)\right) R_{n,m}(t) - \xi_m \left| I\left[\psi_m(t) - \psi_m(t-1)\right] \right| - \varsigma_m \kappa \left|\psi_m(t)\right|, \tag{22}$$

where $|\psi_m(t)|$ represents the number of nonzero elements, $I[\cdot]$ is an auxiliary function, and when $x > 0$, $I(x) = 1$; otherwise, $I(x) = 0$. The instant reward is designed as the total income of MVNO of all current users of system during the time period t, namely,

$$r(t) = \sum_{n=1}^{N} \left(R_n(t) + R_n''(t)\right) + \sum_{m=1}^{M} R_m''(t). \tag{23}$$

Here the long-term return $\mathfrak{R}(t)$ is expressed as

$$\mathfrak{R}(t) = \sum_{t=1}^{T} \epsilon r(t), \tag{24}$$

where $\epsilon \in [0, 1)$ is the discount rate of future earnings weights. When ϵ approaches 1, the system will pay more attention to long-term benefits, and when ϵ approaches 0, the system will pay more attention to short-term benefits.

4.3. Dueling-DQN. DQN is an effective reinforcement learning algorithm, which can make the agent learn good experience from the interaction with environments [31–33]. At the same time, according to DQN learning mechanism, there are improvements to DQN algorithm in different aspects. In DQN, due to the error in the Q estimated value itself, $\max_a Q$ process can be seen according to the expression. It is equivalent to putting forward the largest error, which also leads to the problem of overestimation. Double-DQN is an effective improved algorithm for this problem. In Double-DQN algorithm, the update form of $\widehat{Q}(S)$ is changed to

$$\widehat{Q}(s) = R(s) + \lambda \cdot \widehat{Q}\left(\widehat{s}, \max_a Q_{\text{eval}}(\widehat{s}, a; \alpha); \alpha^-\right), \tag{25}$$

where λ is the discount factor.

The Double-DQN algorithm takes advantage of double neural network and uses two neural networks to learn at the same time, effectively avoiding the overestimation problem caused by error amplification.

Dueling-DQN is also an improvement to DQN algorithm. Compared with previous algorithms, Dueling-DQN algorithm learns faster and has better results. Compared with DQN algorithm, Dueling-DQN retains most of the learning mechanism, and the only difference is the improvement of neural network, as shown in Figure 3.

In the traditional DQN algorithm, the output result is Q value corresponding to each action. In Dueling-DQN algorithm, the output is expressed as a combination of two parts: the value function and advantage function [34]. Among them, value function refers to the value of a certain state, and advantage function refers to the advantage obtained by each action on the state. Therefore, in Dueling-DQN algorithm, Q value problem in DQN can be reexpressed as the following form:

$$Q(s, a; \alpha, \omega_1, \omega_2) = V(s; \omega, \omega_2) + \ell(s, a; \omega, \omega_1) - \frac{1}{|\ell|} \sum_{a'} \ell(s, a'; \omega, \omega_1), \tag{26}$$

where $V(\cdot)$ and $\ell(\cdot)$ are the value function and advantage function, respectively, and ω is the parameter of neural

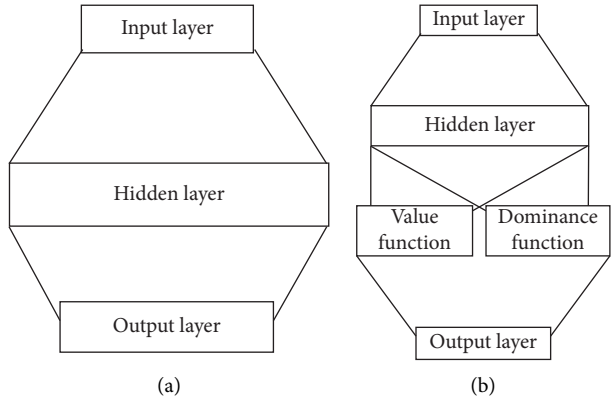

Figure 3: Comparison between DQN algorithm and Dueling-DQN algorithm. (a) DQN. (b) Dueling-DQN.

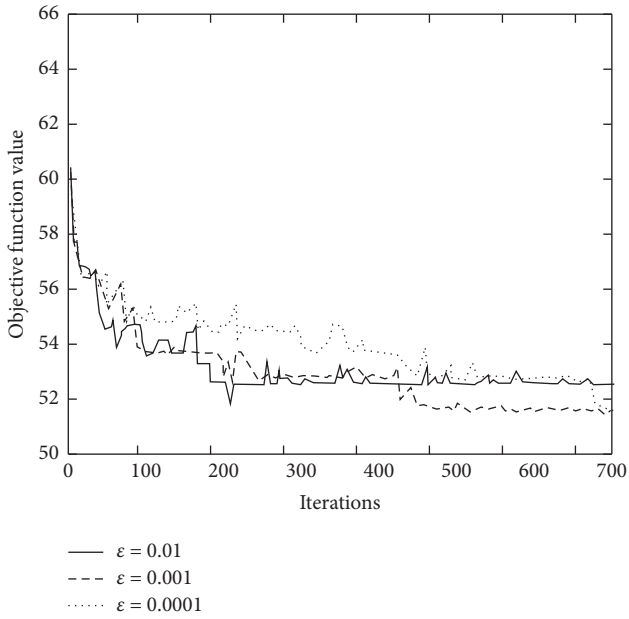

Figure 4: Convergence of the proposed algorithm under different learning rates.

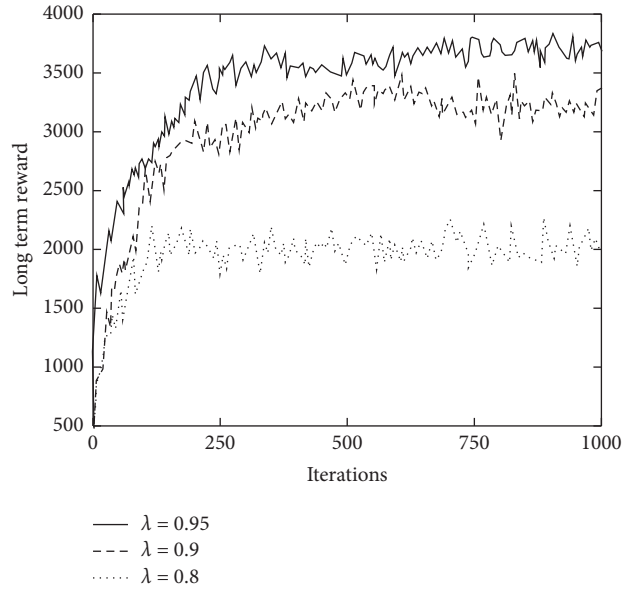

Figure 5: System long-term rewards with different discount rates.

network convolutional layer. ω_1, ω_2 are the parameters of two control flow layers, respectively. The latter item of the plus sign centralizes the advantage function in order to solve the uniqueness problem of Q value.

5. Experimental Results and Analysis

The specific simulation parameters are as follows.

Assume that the computing power of each device n is 1.5 GHz, the uplink transmission power is 800 mW, the idle power is 100 mW, and the upload rate is 2.5 Mb/s. $M = 4$ and overall computing capacity of each MEC server is 6 GHz, 5 GHz, 3 GHz, and 1 GHz, respectively. The data D_n in task G_n obeys uniform distribution of (600, 1200), and the unit is k bits. The workload ϕ_n obeys uniform distribution of (1000, 1500), and the unit is Megacycles.

For the parameters of Dueling-DQN algorithm, set the learning rate ϵ to 0.001 and discount coefficient λ to 0.90. The size of experience replay set is 3000, and the number of randomly sampled samples is 40.

5.1. Parameter Analysis

5.1.1. Learning Rate Analysis. The learning rate of the algorithm will have a great impact on the performance of the proposed strategy. Therefore, three different learning rates ϵ of 0.01, 0.001, and 0.0001 are selected to compare the convergence of improved DQN algorithm, as shown in Figure 4.

5.1.2. Discount Factors Analysis. Similarly, the influence of discount factor on improved DQN algorithm is shown in Figure 5, where the discount factor takes values 0.8, 0.9, and 0.95.

It can be seen from Figure 5 that as the discount factor increases, the long-term reward is continuously increasing. When λ is 0.95, the long-term reward is 3700 when it is

stable. Because the discount factor will affect behavior selection strategy, that is to say, a larger discount factor will cause system to pay more attention to long-term benefits, and a lower discount factor will cause system to pay more attention to current benefits, a higher discount factor will often lead to greater long-term benefits. However, in actual use, using an overly high discount factor does not have corresponding benefits. This is because the system in reality is more changeable, and too much emphasis on future benefits will lead to excessive calculations and excessive losses in the system, which often requires a trade-off.

5.2. Optimization Comparison under Different Objective Functions. For multiobjective optimization problems that reduce time delay and energy consumption, the weighted sum of task execution time delay and terminal execution energy consumption is usually used as the objective function to solve problem, and the calculation is as follows:

$$\Psi' = \frac{\sum_{n=1}^{N} \left(\omega_t \times T_n + \left(1 - \omega_t\right) \times E_n\right)}{N}, \tag{27}$$

where ω_t is the weight coefficient of execution delay and $1 - \omega_t$ is the weight coefficient of execution energy.

Comparing (22) with the proposed objective function (13) to optimize the delay and energy consumption, the number of terminal devices is 12. Considering that the goal of the proposed strategy is to shorten time delay and reduce energy consumption while satisfying the time delay constraints, therefore, the values of ω_t are, respectively, 0.8, 0.6, and 0.4, and the joint experiments of Energy Reduced Scale (ERS) and Time Reduced Scale (TRS) are carried out, as shown in Table 1.

It can be seen from Table 1 that when ω_t is 0.8 and 0.6, the control strategy pays more attention to the optimization of time delay, and when ω_t is 0.4, optimization results pay more attention to the optimization of energy consumption. However, the optimization result of the proposed objective function is the best, and ERS and TRS are 52.18% and 34.72%, respectively, which can shorten time delay and reduce energy consumption under the time delay constraints.

When computing task is 150, comparing control strategies under the four objective functions with the random offloading strategy, the results of ratio of the time delay and energy consumption reduction are shown in Table 2.

It can be seen from Table 2 that delay and energy consumption optimization effect of the proposed optimization target is better, and the reduction ratio of delay and energy consumption is 2.58% and 30.67%, respectively, because the optimization objective of the proposed strategy comprehensively considers the offloading decision and resource allocation plan of joint optimization system when the computing resources are limited and computing tasks have time delay constraints. This allows all computing tasks to shorten completion time and minimize the energy consumption of all terminal devices while meeting the delay constraints. This demonstrates the effectiveness of the proposed objective function.

Table 1: Comparison results of optimization for different objective functions.

Objective function	ERS (%)	TRS (%)
$\omega_t = 0.4$	48.71	27.96
$\omega_t = 0.6$	41.85	31.38
$\omega_t = 0.8$	31.03	32.56
Formula (13)	52.18	34.72

Table 2: Comparison results of optimization for four objective functions.

Objective function	Delay reduction ratio (%)	Energy consumption reduction ratio (%)
$\omega_t = 0.4$	1.25	23.29
$\omega_t = 0.6$	1.97	21.16
$\omega_t = 0.8$	2.03	19.98
Formula (13)	2.58	30.67

5.3. Performance Comparison with Other Algorithms. In order to demonstrate the performance of the proposed strategy, compare it with [12], [19], and [14] in terms of objective function value, calculation amount, and time saving. Li and Jiang [12] proposed a distributed task offloading strategy, which selects the best MEC node offloading amount by game equation on the basis of quantifying offloading cost and delay. Reference [14] used the greedy selection algorithm to design the maximum energy-saving priority algorithm and energy priority strategy to achieve optimal offloading of computing tasks on mobile devices. Reference [19] used the time average calculation rate maximization algorithm to jointly and efficiently allocate radio resources and computing resources.

5.3.1. Algorithm Comparison under Different Cumulative Tasks. In the experiment, objective function value results of the four strategies are shown in Figure 6 for different accumulations of computing tasks.

It can be seen from Figure 6 that the value of objective function is gradually increasing with the increase of cumulative number of tasks for the four offloading strategies. However, the proposed strategy has a relatively lower objective function value than other strategies. That is, the energy consumption and delay are relatively small. For example, when the number of tasks is 180, the objective function value is only 298. Since the proposed strategy considers computation offloading and resource allocation comprehensively, improved deep learning algorithm is used for optimization, and delay and energy consumption are minimized. Reference [19] only matched computing resources but did not rationally optimize the task offloading scheme and computing resource allocation scheme, resulting in high task execution time delay and energy consumption. References [12] and [14] both used corresponding algorithms for optimization to achieve better resource allocation and task offloading. But their analysis of time delay is less, so the performance needs to be strengthened.

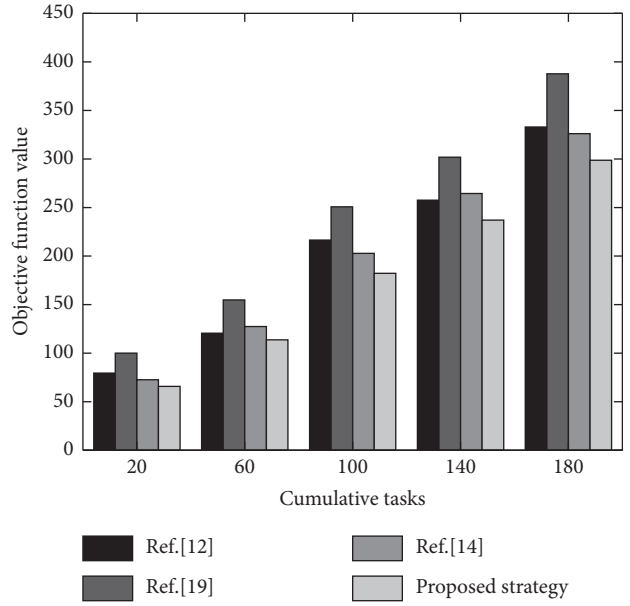

Figure 6: Comparison results of offloading strategies under different cumulative number of tasks.

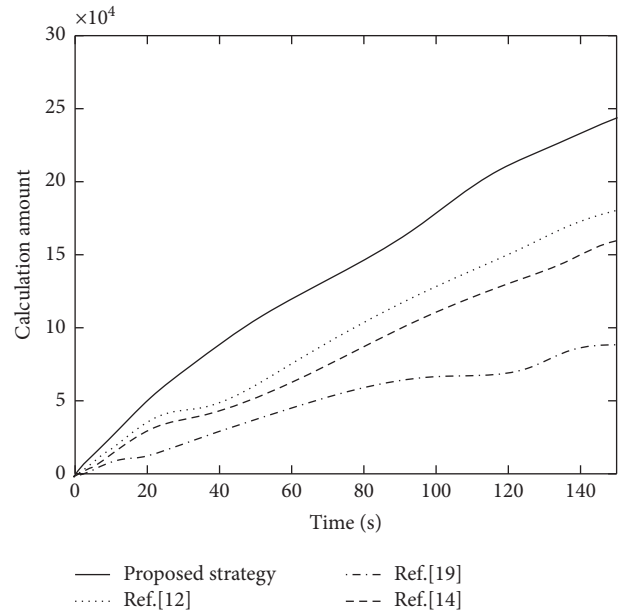

Figure 7: Comparison results of the computation number of offloading tasks under different offloading strategies.

5.3.2. Computation Number Comparison of Offloading Tasks under Different Offloading Strategies. Under four different computation offloading strategies, the comparison results of the computing number of offloading tasks on terminal device side are shown in Figure 7. Vertical axis represents the total calculation number of tasks performed by all terminal devices to perform calculation and offloading. The calculation number of tasks is used to represent the amount of calculation services provided by MEC server. Therefore,

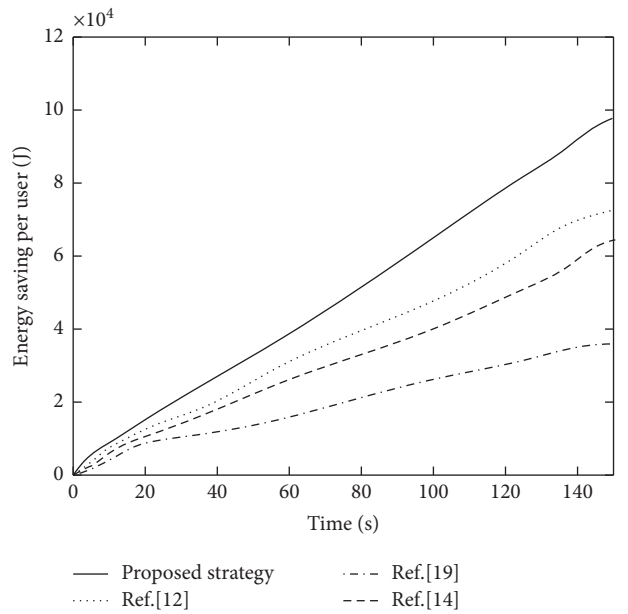

Figure 8: Comparison results of energy saving per unit terminal device under different offloading strategies.

the evaluation indicators in the figure also represent the benefits of computing terminal devices in the offloading mode.

It can be seen from Figure 7 that as time increases, computing tasks continue to increase, and the amount of task calculations also increases. However, the calculation amount of the proposed strategy is significantly better than other comparison strategies. Taking the simulation time of 140 s as an example, compared with [12], [19], and [14], the proposed strategy has increased by 11.54%, 20.83%, and 152.72%, respectively. It can be argued that the proposed strategy is the best compared to task offloading. It uses Dueling-DQN algorithm to process task offloading and resource allocation models, and its optimization performance is better than the greedy selection algorithm in [14] and the game equation model in [12].

5.3.3. Energy-Saving Comparison of per Unit Terminal Devices. Under four different computation offloading strategies, the comparison of energy consumption saved by each terminal device by computation offloading on average is shown in Figure 8. In the local calculation model, all energy consumption is generated by local calculations. In the computation offloading mode, the energy consumption is communication energy consumption caused by upload tasks. For the task of performing computation offloading, the difference between the two is energy saving.

It can be seen from Figure 8 that, compared with other comparison strategies, the proposed strategy has the largest energy-saving rate, which is close to 10×104 J; this also means the least energy consumption. Aiming at the overestimation problem in DQN, the proposed strategy uses Dueling-DQN algorithm for optimization. And it designs the system benefits and resource consumption as rewards and

losses, which improves the efficiency and rationality of task offloading and resource allocation by optimizing problem solution. Reference [19] only used the time average calculation rate maximization algorithm to efficiently allocate computing resources. The optimization algorithm is more traditional and has poor performance. Thus, the overall energy saving is not high. Reference [12] used the game equation model to optimize task offloading strategy but does not realize the rationalization of resource allocation. Therefore, the maximum energy saving is 710×104 J. Reference [14] used greedy selection algorithm to design an optimal energy-saving strategy but did not consider server computing resource constraints and task delay constraints. Therefore, the overall performance is not as good as the proposed strategy.

6. Conclusion

MEC server has limited computing resources and computing task has delay constraint. How to shorten completion time and reduce terminal energy consumption under the delay constraints becomes an important research issue. To solve this problem, this paper proposes a task offloading and resource allocation strategy based on deep learning for MEC. In the multiuser multiserver MEC environment, a new objective function is designed to construct mathematical model. In combination with deep reinforcement learning, the partially improved Dueling-DQN algorithm is used to solve the optimization problem model, which can reduce the completion time of computing tasks and minimize energy consumption of all terminal devices under the delay constraints. The proposed strategy is demonstrated by experiments based on Python platform. The experimental results show that when learning rate is 0.001 and discount factor is 0.90, the energy saving is close to 10×104 J, which is better than other comparison strategies. In terms of calculation amount, it increased by 11.54%, 20.83%, and 152.72%, respectively.

In practice, different users have different concerns about service quality. Therefore, we can refer to the different needs of users when making computation and offloading decisions in the following research. It can assign a certain weight to the factors affecting the quality of service and combine the task priority for scheduling.

Data Availability

The data used to support the findings of this study are included within the article.

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 Key Disciplines of Computer Science and Technology (2019xjzdxk1) and New Engineering Pilot Project (szxy2018xgk05).

References

[1] W. P. Peng, Z. Su, C. Song, and J. Zongpu, "Research on adaptive dual task offloading decision algorithm for parking space recommendation service," *The Journal of China Universities of Posts and Telecommunications*, vol. 26, no. 06, pp. 33–45, 2019.

[2] K. Wang, X. F. Wang, X. Liu, and A. Jolfaei, "Task offloading strategy based on reinforcement learning computing in edge computing architecture of internet of vehicles," *IEEE Access*, vol. 8, no. 6, pp. 173779–173789, 2020.

[3] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2021.

[4] Y. Sun, X. Guo, J. Song et al., "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3061–3074, 2019.

[5] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3415–3426, 2020.

[6] 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, no. 5, pp. 324–331, 2020.

[7] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3341–3356, 2020.

[8] Z. Wei, J. Pan, Z. Lyu, J. Xu, L. Shi, and J. Xu, "An offloading strategy with soft time windows in mobile edge computing," *Computer Communications*, vol. 164, no. 8, pp. 42–49, 2020.

[9] R. Zhang, P. Cheng, Z. Chen, S. Liu, Y. Li, and B. Vucetic, "Online learning enabled task offloading for vehicular edge computing," *IEEE Wireless Communications Letters*, vol. 9, no. 7, pp. 1–932, 2020.

[10] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile edge computing in dense cloud RAN," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[11] J. Zhang, H. Guo, and J. Liu, "Adaptive task offloading in vehicular edge computing networks: a reinforcement learning based scheme," *Mobile Networks and Applications*, vol. 25, no. 5, pp. 1736–1745, 2020.

[12] Y. Li and C. Jiang, "Distributed task offloading strategy to low load base stations in mobile edge computing environment," *Computer Communications*, vol. 164, no. 2, pp. 240–248, 2020.

[13] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: a load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[14] 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.

[15] 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, no. 3, pp. 847–861, 2020.

[16] L. Li and H. Zhang, "Delay optimization strategy for service cache and task offloading in three-tier architecture mobile edge computing system," *IEEE Access*, vol. 8, no. 9, pp. 170211–170224, 2020.

[17] X. Zhang, J. Zhang, Z. Liu, Q. Cui, X. Tao, and S. Wang, "MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3296–3309, 2020.

[18] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2443–2459, 2020.

[19] C. Li, W. Chen, J. Tang, and Y. Luo, "Radio and computing resource allocation with energy harvesting devices in mobile edge computing environment," *Computer Communications*, vol. 145, no. 09, pp. 193–202, 2019.

[20] Z. Ali, S. Khaf, Z. H. Abba, G Abbas, and L Jiao, "A Comprehensive Utility Function for Resource Allocation in Mobile Edge Computing," *arXiv preprint arXiv:2012.10468*, vol. 66, no. 2, pp. 1461–1477, 2020.

[21] P. Q. Huang, Y. Wang, K. Wang, and L. Zhi-Zhong, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 1–14, 2019.

[22] Y. Liu, Y. Li, Y. Niu, and D. Jin, "Joint optimization of path planning and resource allocation in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 9, pp. 2129–2144, 2020.

[23] Y. A. Lei, A. Cz, B. Qy, W. Zou, and A. Fathalla, "Task offloading for directed acyclic graph applications based on edge computing in Industrial Internet-ScienceDirect," *Information Sciences*, vol. 540, no. 7, pp. 51–68, 2020.

[24] X. F. He, R. C. Jin, and H. Y. Dai, "Peace: privacy-preserving and cost-efficient task offloading for mobile-edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1814–1824, 2020.

[25] B. Gu and Z. Zhou, "Task offloading in vehicular mobile edge computing: a matching-theoretic framework," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 100–106, 2019.

[26] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, no. 5, pp. 54074–54084, 2020.

[27] Y. Pan, M. Chen, Z. Yang, N. Huang, and M. Shikh-Bahaei, "Energy-efficient NOMA-based mobile edge computing offloading," *IEEE Communications Letters*, vol. 23, no. 2, pp. 310–313, 2019.

[28] Y. L. Jiang, Y. S. Chen, S. W. Yang, and C. H. Wu, "Energy-efficient task offloading for time-sensitive applications in fog computing," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2930–2941, 2019.

[29] X. Xu, Q. Liu, Y. Luo et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, no. 06, pp. 522–533, 2019.

[30] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: a dependency-aware and latency-optimal approach," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1678–1689, 2020.

[31] S. Hu and G. Li, "Dynamic request scheduling optimization in mobile edge computing for IoT applications," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1426–1437, 2020.

[32] J. Zeng, J. Sun, B. Wu, and X. Su, "Mobile edge communications, computing, and caching (MEC3) technology in the maritime communication network," *China Communications*, vol. 17, no. 5, pp. 223–234, 2020.

[33] Q. Lin, F. Wang, and J. Xu, "Optimal task offloading scheduling for energy efficient D2D cooperative computing," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1816–1820, 2019.

[34] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and M. Z. A. Bhuiyan, "Joint optimization of offloading utility and privacy for edge computing enabled IoT," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2622–2629, 2020.