

Research Article

Computing Unloading Strategy of Massive Internet of Things Devices Based on Game Theory in Mobile Edge Computing

Xinhui Ding  and Wenjuan Zhang 

Zhoukou Normal University, Zhoukou, Henan 466001, China

Correspondence should be addressed to Xinhui Ding; xinhuiding2019@126.com

Received 2 June 2020; Revised 28 January 2021; Accepted 1 February 2021; Published 2 March 2021

Academic Editor: Quanxin Zhu

Copyright © 2021 Xinhui Ding and Wenjuan Zhang. 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.

Due to the limited computing resources of the mobile edge computing (MEC) server, a massive Internet of things device computing unloading strategy using game theory in mobile edge computing is proposed. First of all, in order to make full use of the massive local Internet of things equipment resources, a new MEC system computing an unloading system model based on device-to-device (D2D) communication is designed and modeled, including communication model, task model, and computing model. Then, by using the utility function, the parameters are substituted into it, and the optimization problem with the goal of maximizing the number of CPU cycles and minimizing the energy consumption is constructed with the unloading strategy and power as constraints. Finally, the game theory is used to solve the problem of computing offload. Based on the proposed beneficial task offload theory, combined with the mobile user device computing offload task amount, transmission rate, idle device performance, and other factors, the computing offload scheme suitable for their own situation is selected. The simulation results show that the proposed scheme has better convergence characteristics, and, compared with other schemes, the proposed scheme significantly improves the amount of data transmission and reduces the energy consumption of the task.

1. Introduction

With the increasing popularity of smart devices, many new mobile applications and technologies have emerged such as face recognition, car networking system, mobile communication and payment, natural language processing, and interactive games. Of course, this inevitably brings about problems such as lack of resources, a large amount of calculation, and high energy consumption [1, 2]. Due to physical size limitations, mobile devices are often constrained by processing resources, storage resources, and battery capacity. Task offloading has become an interesting and promising solution to alleviate this tension and has become the focus of academia and industry [3]. In the past decade, many researchers have focused on mobile cloud computing, and mobile users can transfer computing-intensive tasks to remote, resource-rich cloud servers through wireless access [4]. Although this paradigm has been used as a form of commercial cloud service, it still has many problems, such as unstable wireless

connections between mobile devices and the cloud (for example, weak cellular signals), Wide Area Network (WAN) delays, and other issues. Mobile users usually have a high demand for localized and location-based information services, and it is often inefficient to retrieve localized data from a remote cloud [5].

In recent years, with the explosion of mobile communication traffic, abnormally large amounts of data have been transmitted to cloud servers. This not only puts a heavy burden on the communication bandwidth but also causes unbearable transmission delays and reduces the quality of service to end users. As mobile users and traffic services become mainstream, in addition to the low-latency guarantee of real-time interaction, support for mobility and location management is also crucial [6]. Therefore, as cloud processing becomes the primary method of centralized information storage, retrieval, and management, mobile devices become the main destination of information transmission, and the successful integration of cloud

computing and mobile applications has become an important task. In order to face the above challenges, edge computing is proposed by researchers as an extension of cloud computing. It uses a large number of collaborative user terminals or adjacent user equipment (UE) to perform a large number of computing tasks. The goal is to partially process workloads and services on edge computing devices (such as rugged routers, switches, and IP cameras), rather than transmitting them to cloud devices. Therefore, edge computing introduces an intermediate layer between the mobile user and the cloud to achieve mutual compensation with cloud computing and provide low latency and high rate services to mobile users [7]. It can independently provide predefined service applications to mobile users without the help of the cloud or the Internet. On the other hand, you can also connect the edge computing server to the cloud to take advantage of the cloud's rich functions and application tools. Edge computing is not a substitute for cloud computing, but a supplement to cloud computing, to reduce the bandwidth burden and reduce the transmission delay [8]. In particular, edge computing services can support and promote the development of applications that are not suitable for cloud computing, such as the following four scenarios:

- (1) Applications that require very low and predictable latency, such as online games and video conferencing
- (2) Geographically distributed applications such as pipeline monitoring and sensor networks
- (3) Fast mobile applications, such as smart connected cars
- (4) Large-scale distributed control systems such as intelligent energy distribution and intelligent traffic lights

Task migration and resource management are key research issues in mobile edge computing [9]. Among them, task migration is to solve how to select the processing server to process the task (such as local/proximity mobile device/edge server/cloud server), how to divide the related tasks in the same application, and how to determine the time and sequence of task offload/transmission; resources management is to study the wireless (task transmission) and computing (task processing) resource allocation and scheduling schemes required in the calculation process, the task unloading access control scheme, and the edge server networking collaboration scheme [10, 11]. In mobile edge computing, different edge servers need to process tasks from massive mobile terminals in real time and need to coordinate the allocation of edge server resources and task loads at all levels to meet the heterogeneity of processing delay, execution energy consumption, and reliability of different tasks demand [12]. Task migration and resource management directly affect the performance of mobile edge computing and are of great research significance [13].

Therefore, this study proposes a massive computing offload strategy based on game theory in mobile edge computing. This strategy can combine factors such as calculation offload task, transmission rate, and performance of idle devices of mobile user equipment to select a calculation

offload solution suitable for its own situation. The results show that the proposed scheme has better convergence characteristics, and compared with other schemes, the overall system load of the proposed scheme is smaller, and it significantly increases the amount of data transmitted and reduces the task energy consumption. Its main innovations are reflected in the following points:

- (1) In order to make full use of the massive local IoT device resources, a new MEC system computing offload system model based on device-to-device (D2D) communication is designed. Mobile users can offload tasks to the MEC server, or use D2D to offload to idle and rich computing resources, and can also choose to perform offload tasks locally.
- (2) Aiming at the problem of huge data volume in the Internet of Things and the low efficiency of the calculation offloading scheme, game theory is used to solve the calculation offloading problem. This paper proposes to choose a computing offloading scheme suitable for one's own situation based on the beneficial task offloading theory, combined with mobile user equipment's calculation of offloading task volume, transmission rate, and performance of idle devices.

2. Related Research

There has been much related research on mobile edge computing technology. Some scholars compare the performance of local computing with edge computing and consider the choice of computing tasks. Reference [14] has designed a computational offload strategy to achieve the purpose of minimizing the energy overhead during the computational offload; reference [15] considers the time overhead of computing tasks during transmission and calculation to minimize the system delay. Purpose: computation efficiency maximization problems in [16] are formulated in wireless-powered MEC networks under both partial and binary computation offloading modes. The research in [17] mainly studies the dynamic migration of users during the calculation offload process. It also proposes a computing offload strategy based on the Markov chain to predict the possible future migration and access location of users; the main work in reference [18] is to consider the problem of computing offload for resource allocation. In [19], the optimal selection scheme on whether users choose to locally compute or offload computation tasks is proposed for the binary computation offloading mode. In [20], in order to improve the security of computation tasks offloading and enhance user connectivity, physical layer security and nonorthogonal multiple access (NOMA) are studied in MEC-aware networks.

According to the edge computing architecture used for task offloading, existing research can be roughly divided into two categories, namely, local cloud-based architecture and user collaboration-based architecture.

In a local cloud-based architecture, network operators and cloud providers collaborate to deploy microclouds in the

local area network where their base stations are located to provide computing resources. At the same time, they provide an authorized controller (such as a base station) that combines wireless and computing resources for mobile users coordinating the scheduling and offloading of computing tasks [21]. For example, the study in [22] considered the task offloading problem as a joint optimization of wireless resources and computing resources to minimize the energy consumption of the overall users.

The study in [23] proposed a bilateral control algorithm to jointly perform task offloading and cloud resource scheduling decisions to minimize the overall user overhead cost. Reference [24] constructed edge computing multiuser competition in WLAN and Code Division Multiple Access (CDMA) single-cell network as noncooperative game problems and designed distributed solutions to solve. Generally speaking, there are three advantages of the local cloud-based architecture. First of all, network operators can be authorized to obtain device status information and task configuration files in order to make reasonable offload decisions. Secondly, the authorized controller (such as the base station) is responsible for task offload scheduling and can provide comprehensive offload decision support for all users without consuming mobile device energy. Furthermore, authorized controllers can use their global network information (such as user real-time cellular signals) to make task offload decisions. However, its existence clearly determines that all tasks are offloaded to the local cloud through cellular access, and this additional traffic will increase the burden on future cellular networks [25]. In addition, the resources in the local cloud are limited, which may limit the scale of services and require dedicated resource scheduling strategies. In addition, since this architecture only provides computing resources, it is not helpful for traffic offloading and some mobile tasks that require smartphone awareness.

In the user collaboration-based architecture, mobile users can use resources that are not fully utilized by nearby mobile devices to assist in the execution of tasks. The rationale is that, in daily life, users will encounter this opportunity very often, which provides mobile users with a large number of opportunities to form a local network to share their resources [26]. In addition, the improvement and diversification of mobile device performance have brought a large number of heterogeneous resources to the local network, such as advanced CPUs supporting high-level applications, high-speed network access, and various sensors. Furthermore, D2D communication is expected to improve cellular communication in terms of improving user throughput and expanding the network coverage. There are many studies currently in this category, for example, the task offload framework proposed in [27], where mobile users offload computing tasks to nearby users based on available processing power. The study in [28] designed a traffic offloading framework to offload users' data files to nearby users to achieve better cellular quality. However, in order to successfully implement an architecture based on user collaboration, there are two prerequisites. First, user mobility, such as time spent in a place and contact behavior, should be predictable. Second, cooperative users should trust each

other or provide an incentive mechanism. Most of the existing research uses users' social relationships to meet these needs but, at the same time, may limit the services in the community and lose a lot of opportunities to cooperate with strangers nearby. In addition, they usually focus on single-user optimization rather than full network optimization, ignoring the energy consumption of devices used to predict user mobility and make offload decisions. Of course, these two architectures can be complementary, and the complementary relationship between the two has also attracted the attention of many researchers, which has also produced a lot of research results on task scheduling strategies. For example, the study in [29] designed a new D2D edge computing architecture based on the above two architectures and considered combining user incentives and collaboration into a wide range of task offload optimization problems.

Although many scholars have done relevant research on mobile edge computing, most of the research is only on the resource allocation of computing offload and the optimization of network overhead during the offload of computing. It did not consider improving the users' experience by reducing the computing pressure of MEC server [30]. The development and application of mobile edge computing make the edge computing server not only limited to the "edge computing layer" at the edge of the network but also gradually expand to neighboring devices around the user. By establishing a D2D connection with the surrounding mobile users, the computing task is offloaded to the user equipment with a large amount of idle computing resources; that is, the D2D computing task offloading. D2D computing task offloading is a new mobile task offloading framework based on D2D collaboration. In this collaboration, users can share each other's computing resources. Compared with offloading to an edge server, D2D assisted unicast is easier to implement, because users can help each other and benefit together. Based on the information in the current time slot, their tasks are dynamically offloaded to the nearby user equipment that has a lot of idle CPU resources. In a large amount of research work on edge computing and D2D offloading, a few research results have taken into account information security issues [31]. Whether it is a centralized or distributed architecture, the focus of most research is how to make offload decisions, how to schedule resources, and how to optimize system performance after base stations or UEs have mastered information about various mobile devices and edge servers.

3. Modeling

3.1. System Modeling. The system model considered is a single cell, which includes a base station and users divided into two sets. In a set, each user has a computationally intensive task to be performed. These users are called computing requesters (CRs), denoted as $\{1, 2, \dots, R\}$. The users in the other set, because they currently have free CPU resources, can be used as computing providers (CPs), denoted as $\{1, 2, \dots, P\}$ as shown in Figure 1.

In this case, the base station will encourage CRs to offload their tasks to the nearby CPs through energy-efficient

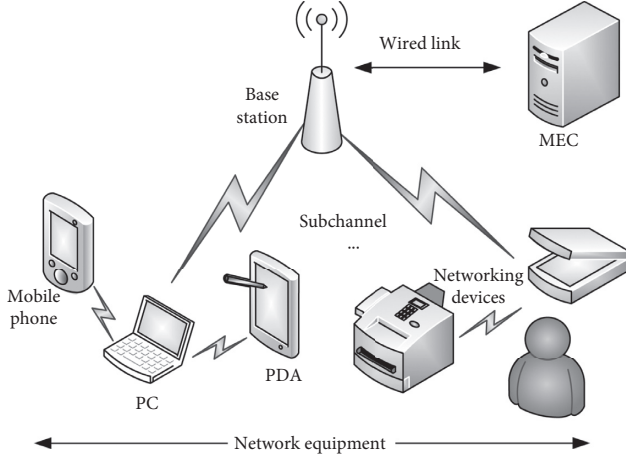


FIGURE 1: The model of Internet of Things device unloading system.

D2D links. Of course, users can also choose to perform their tasks locally [32]. The available spectrum in this single-cell system is divided into N orthogonal subchannels, and the bandwidth of each subchannel is w . Each D2D link will be assigned an OFDMA subchannel. The operation of the system is performed in each time slot, and the length of each time slot is T . Assuming that the base station has complete network information and high computing power, each user has only one task in a time slot, and the user's offload decision and the base station's decision to allocate subcarriers for each pair of D2D are completed in a time slot.

The D2D task uninstall process includes the following steps:

- (1) CRs carrying computing tasks look for CPs of neighboring computing providers and then measure and collect information of these neighboring users.
- (2) After calculation, CRs make the best matching decision. Of course, they can also choose to perform tasks locally.
- (3) The base station allocates subcarriers for each pair of D2D, CRs offloads the task to the selected CPs, and then the task is executed at the CPs or CRs.
- (4) Finally, the calculation results will be returned to CRs from CPs.

3.2. Related Model Building

3.2.1. Communication Model. Consider that the working mode of D2D communication is an over mode. In this mode, there is no interference between D2D users and cellular users and between D2D pairs. In the following, r means a CR and p means a CP. Therefore, in the D2D link established between r and p , the transmission rate v_{rp} in a time slot T can be expressed as

$$v_{rp} = w \log \left(1 + \frac{P_r H_{rp}}{\rho_0 w} \right). \quad (1)$$

Here, H_{rp} is the channel gain between r and p and w is the bandwidth allocated to the D2D link between the two. ρ_0 is the background noise power spectral density, and P_r represents the transmission power of r in the current time slot.

3.2.2. Task Model. In the D2D task offload architecture, consider using a parameter set $\langle b_{in}, b_{out}, s_r \rangle$ to describe the calculation task of the current slot user r , where b_{in} is incoming data, b_{out} is outgoing data of the task, and s_r is the number of computing resources required by the task, expressed as

$$s_r = b_{in} \cdot k. \quad (2)$$

Here, k is a constant.

3.2.3. Calculation Model

(1) Local Computing. f_r^l is used to represent the computing power of r (such as the amount of data the device can process per second), and different CRs have different computing power [32]. According to the task's parameter set, the execution time of the r local computing task can be expressed as

$$t_r^l = \frac{s_r}{f_r^l}. \quad (3)$$

Then, the corresponding energy consumption is expressed as

$$E_r^l = e_r t_r^l. \quad (4)$$

Here, e_r is the energy consumed per second calculated according to the current processor frequency.

(2) Offload Calculation. The time delay for offload calculation includes three parts, r and p communication delays σ_{in} and σ_{out} due to outgoing and incoming data through the D2D link, and task processing delay σ_{exe} . Therefore, the total time delay for r offload calculation is

$$t_{rp}^p = \sigma_{in} + \sigma_{out} + \sigma_{exe},$$

$$\sigma_{in} = \frac{b_{in}}{v_{rp}}, \quad (5)$$

$$\sigma_{exe} = \frac{s_r}{f_p}.$$

Here, f_p is the computing power of p . Compared with the size of the incoming data, the amount of data transmitted by the CP is too small, so the transmission delay σ_{out} of the outgoing data can be regarded as a constant. The total energy consumption of the offload calculation mode can be expressed as

$$\begin{aligned}
E_{rp}^P &= e_{\text{in}} + e_{\text{out}} + e_{\text{exe}}^P, \\
e_{\text{in}} &= P_r \cdot \sigma_{\text{in}}, \\
e_{\text{out}} &= P_p \cdot \sigma_{\text{out}}, \\
e_{\text{exe}}^P &= e_p \cdot \sigma_{\text{exe}}.
\end{aligned} \tag{6}$$

Here, P_r and P_p represent the D2D transmission power of r and p , respectively, and e_p is the energy consumed by p to perform the calculation task per second.

3.3. The Prediction of MEC Server Waiting Time. The prediction of the waiting time when too many tasks are queued on the MEC server because the MEC server is a single queue multiservice queuing model: according to Little's law in queuing theory, under balanced conditions, the average time a task waits at the MEC server is the average waiting queue length of the system divided by the average entry rate of the task, which is

$$t_{\text{wait}} = \frac{\bar{N}_q}{\bar{\lambda}}, \bar{\lambda} = \frac{N_t - N_0}{t}. \tag{7}$$

Here, \bar{N}_q is the average waiting team leader, $\bar{\lambda}$ is the average entry rate of tasks, and N_t is the total number of tasks at t , the number of tasks in the system at the beginning of the N_0 decision, rather than the number of tasks in the initial system. Next, you need to measure these two parameters. The measurement of these two parameters needs to be performed on the MEC server.

For \bar{N}_q , the number of tasks waiting at the MEC $N_t - C$ can be counted in each time slot t , and the average waiting queue length can be calculated as the time increases:

$$\bar{N}_q = \frac{\sum_{i=0}^t (N_t - C)}{t}. \tag{8}$$

3.4. Utility Function. Because each mobile user wants to transmit as many CPU cycles (transmitted data) as possible, each participant wants to maximize their transmission power. However, if each participant tends to maximize its transmission power, it will cause severe interference, which will affect the transmission rate and generate huge energy consumption [33]. In this way, the overall performance of the system declines instead. At the same time, while maximizing power, it will generate a lot of energy consumption, which will also increase costs. Therefore, the utility function wants to maximize the number of CPU cycles while making the energy consumption as small as possible, so the utility function used is

$$U(S) = \alpha D_n(S) - \beta q(E_n(s_n)). \tag{9}$$

The utility function maximizes the number of CPU cycles while minimizing energy consumption, where α is the price required per unit of CPU cycles, β is the price required per unit of energy consumption, $D_n(S)$ is the total amount of data transferred by the mobile user n , and E_n is the total energy consumption. The larger the utility function, the

better the effect. After that, each parameter is substituted into the utility function; with the unloading strategy and power as constraints, the optimization problem with the objective of maximizing the utility function is constructed and solved.

4. Distributed Computing Offload Algorithm Based on Game Theory

4.1. Useful Task Offload. Through the analysis of the above communication model and computing task load model, it can be seen that when too many mobile devices select the same channel for task offloading, the mutual interference will become very serious. This leads to a reduction in the data rate between each mobile device and the base station, which results in more time when uploading calculation task data. And spending too much time uploading computing tasks will cause more energy consumption of mobile devices [34, 35]. In this case, mobile devices are more suitable to perform tasks locally to avoid the excessive load caused by offloading tasks.

Since each mobile user is an individual, each individual will only consider their own interests in a multiuser scenario. The benefit here is to complete the calculation task with the minimum energy consumption and the shortest delay. The beneficial task offloading concept is defined below.

Definition 1. Given a multiuser task offload strategy result vector x , the decision result is $x_n (x_n \geq 0)$ for the user n who selected the offload task. If the load of the user performing the calculation task on the MEC server by offloading is less than the load of performing the calculation task locally, then it is called a beneficial task offloading ($K_n^m \geq K_n^c(x)$).

The concept of beneficial task offloading has an important meaning in the task offloading strategy of mobile edge computing. On the one hand, from the perspective of mobile users, a user will not offload tasks to the MEC server for execution and cannot obtain a smaller load than local execution. Only by offloading tasks to the MEC server which can get a smaller load, mobile users are motivated to uninstall. On the other hand, from the perspective of the MEC server operator, if more users can achieve a beneficial task offload status, it means more users of the MEC server, which means higher profits. Therefore, the offload strategy can be derived from the calculation load (local calculation load or load offloaded to the MEC for calculation) and according to the beneficial task offload concept.

With the concept of beneficial task offloading, a goal can be to maximize the number of users who achieve a beneficial state through task offloading in a multiuser scenario. This model can be expressed by the following formula:

$$\begin{aligned}
&\max_x \sum_{n \in N} I_{\{x_n \geq 0\}} \\
&\text{satisfy : } K_n^c(x) \leq K_n^m, \forall x_n > 0, n \in N \\
&x_n \in \{-1, 0, 1, \dots, M\}, \forall n \in N.
\end{aligned} \tag{10}$$

Here, $I_{\{x_n \geq 0\}}$ is an indicator function, defined as follows:

$$I_{\{x_n \geq 0\}} = \begin{cases} 1, & x_n \geq 0, \\ 0, & x_n = -1. \end{cases} \quad (11)$$

At this time, the task offload strategy problem in the multiuser scenario can be described as follows: in the multiuser scenario, most users are allowed to achieve a beneficial offload state by offloading the computing task. However, by translating this problem into the maximum

packing problem of multiple boxes, it can be concluded that this problem is NP-hard. Let us prove it.

To prove this problem, we introduced the maximum packing problem of multiple boxes. Given N objects, the weight of each object is W_i , $i \in \{1, \dots, N\}$, and there are M boxes, and the capacity of each box is C . The goal of the problem is to pack the objects in the box so that most objects can be packed in the box. The problem can be described as follows:

$$\begin{aligned} & \max \sum_{i=1}^N \sum_{j=1}^M h_{ij} \\ & \text{satisfy } \sum_{i=1}^N W_i h_{ij} \leq C, \quad \forall j \in \{1, \dots, M\} \\ & \sum_{j=1}^M h_{ij} \leq 1, \quad \forall i \in \{1, \dots, N\} \\ & h_{ij} \in \{0, 1\}, \quad \forall i \in \{1, \dots, N\}, j \in \{1, \dots, M\}. \end{aligned} \quad (12)$$

The user's mobile device is compared to the object in the maximum packing problem, and the channel is compared to the box in the maximum packing problem. The weight W_i of an object can be expressed as $W_i = q_n g_{ns}$, and the capacity of the box can be expressed as the data rate of the channel. If a device n selects the channel m , it can be interpreted that the object n is loaded into the box m . And if the equipment reaches a beneficial task unloading state, it means that the box m has not exceeded the capacity limit.

4.2. Algorithm for Calculating Offload. A game theory method is used to solve the problem of computing offloading, which is used to realize an efficient computing offloading strategy among users. The reasons for adopting game theory are summarized as follows:

- (1) Each user has different mobile devices. They may pursue different interests and adopt different strategies. For example, when the battery status of mobile devices is low, mobile users will be more inclined to choose to uninstall to edge devices with high latency but low energy consumption. When users run some applications that are sensitive to latency, users pay more attention to processing time. If low latency is required, mobile users will choose to offload to mobile edge servers. Game theory is a framework for analyzing interactions between multiple mobile device users. These users choose their own unloading decisions based on their own interests and maximize their own interests. A certain incentive computing offload mechanism can be designed so that each mobile user has no incentive to unilaterally deviate [36]. It is very suitable for research on mobile edge computing regarding the offload of computing.

- (2) Because of the future Internet of Things scenarios with massive devices, each mobile user's device will generate data, and massive devices will generate massive data. If we adopt a centralized processing method, it will inevitably bring huge energy consumption and delay to the server. Therefore, the research method of noncooperative games has become the main method for processing tasks in a distributed manner. Generally, in a distributed system, each mobile user prefers to choose a method of task execution that enables his user equipment to have low overhead.

Assume that $S_{-n} = (S_1, \dots, S_{n-1}, S_{n+1}, \dots, S_N)$ is the calculation offload decision of all other mobile users except for user n . Suppose that when user n selects strategy S_n , the user knows the calculation offload strategy S_{-n} of other MEC. So, you can choose the best strategy at this time.

Therefore, a noncooperative game model is established, each user decides their own offload strategy, and a distributed architecture model is jointly established. Under the conditions of power and different unloading strategies, each user should maximize their respective utility functions:

$$\begin{aligned} & \forall n \in N, \max_{S_n} U_n \\ & \text{s.t. } p_n \leq P_{\max} \\ & \sum_{k=0}^{K+1} x_{n,k} = 1 \quad \forall n \in N. \end{aligned} \quad (13)$$

If under the S^* strategy, no user can change it alone and the other utility function becomes larger, then the $S^* = (s_1^*, \dots, s_N^*)$ strategy is to calculate the Nash equilibrium solution of the offload game:

$$U_n(s_n^*, s_{-n}^*) \geq U_n(s_n, s_{-n}^*), \quad \forall s_n \in S_n, n \in N. \quad (14)$$

Among the many game models, the potential game is one of the most effective game theoretical models applied in the distributed network. In game theory, if a single global function called a potential function can be used to express the incentives for all players to change their strategy, the game is called a potential game.

In addition, design a computing offload algorithm suitable for the distributed model in this section. First, the algorithm can enable users to coordinate with each other and know each other's uninstall decisions before choosing an uninstall strategy. Secondly, after multiple time slot iterations, the joint offload game model proposed in this paper can achieve Nash equilibrium. Let us first consider the update of the task offload decision within a time slot, and introduce the concepts of better response and optimal response.

Definition 2. The mobile user offload decision is changed from s_n to s'_n , and its utility function changes to satisfy

$$U_n(s'_n, s_{-n}) > U_n(s_n, s_{-n}). \quad (15)$$

Then, the strategy is called s'_n as the better response strategy.

Definition 3. For a given other participant strategy s_{-n} , the mobile user offload decision changes from s_n to s_n^* , and there is no better strategy than s_n^* . Its utility function changes satisfy

$$U_n(s_n^*, s_{-n}) > U_n(s_n, s_{-n}). \quad (16)$$

Then, the strategy is called s_n^* as the optimal response strategy.

According to the concept of better response in the potential game in formula (14), mobile users can choose a relatively superior offload strategy in each time slot. According to the concept of the optimal response of the potential game in formula (15), each mobile user is the optimal participant in the strategy set of other mobile users [37, 38].

Description of how mobile users update their own decisions: First, at the beginning of each time slot, the mobile user's computing offload strategy is initialized. When the mobile user has an uninstall task that needs to be performed, under the condition that other mobile user strategies remain unchanged, choose a strategy that is better for him. After multiple timeslot iterations, each participant's decision will achieve relative superiority [39, 40]. If the offload strategy of all mobile users at this time satisfies formula (15), it indicates that the entire offload model has reached the Nash equilibrium and the algorithm has converged to the global optimal situation.

The joint offloading algorithm proposed based on the system model is convergent and will always have Nash equilibrium. When that equilibrium is reached, the proposed utility function reaches the global optimum, so the potential game can finally obtain a satisfactory

solution. The flow of the proposed algorithm is shown in Figure 2.

5. Example Verification and Result Discussion

Based on the MATLAB platform, the performance of the proposed calculation offload strategy is evaluated through specific simulation data and necessary instructions. This includes parameter settings, simulation results, and analysis. The simulation process considers a single-cell D2D network model, in which all CRs and CPs are randomly distributed in the cell. The main simulation parameters are given in Table 1.

5.1. Iterative Analysis. Considering the number of different CRs, this paper conducts experiments on the convergence performance of the proposed algorithm. The result is shown in Figure 3, the maximum D2D distance at this time is fixed at 50 m.

It can be seen from the figure that when the number of CRs is 5, 10, and 15, the total energy consumption generated by the system can always reach convergence after a series of consecutive iterations. At the same time, as the number of CR gradually increases, the total energy consumption also increases accordingly. This is because the more the user transmission and calculation tasks are, the more the energy consumption will be generated. This shows that the algorithm is guaranteed to converge. In addition, as the number of CRs increases, the rate of convergence also gradually weakens. Therefore, when the number of CRs is large, the complexity of the algorithm will become very high, which will be a problem that needs to be further solved in the future.

During the iterative process, the number of users in a beneficial uninstall state changes as shown in Figure 4.

It can be seen from the figure that, during the iterative process, the number of users in a beneficial uninstall state continues to increase and eventually reaches a stable state. This shows that the algorithm can reach the Nash equilibrium state within a limited number of times (35 iterations in this simulation experiment). Among the 30 users, 25 users selected tasks to be offloaded to the MEC server for execution.

In addition, under the condition of a different number of users, the number of iterations required for the system to reach Nash equilibrium is shown in Figure 5.

It can be seen from the figure that, as the number of users increases, the number of iterations required for the system to reach Nash equilibrium also increases with a linear trend. This shows that the proposed multiuser distributed task offloading algorithm has better performance.

5.2. Parameter Discussion. In order to analyze how the background noise power spectral density ρ_0 affects the energy consumption of the distributed computing offload algorithm, the energy cost is compared by changing the density value. The results are shown in Figure 6. This article compares the situation of 10 users to 50 users, respectively. It

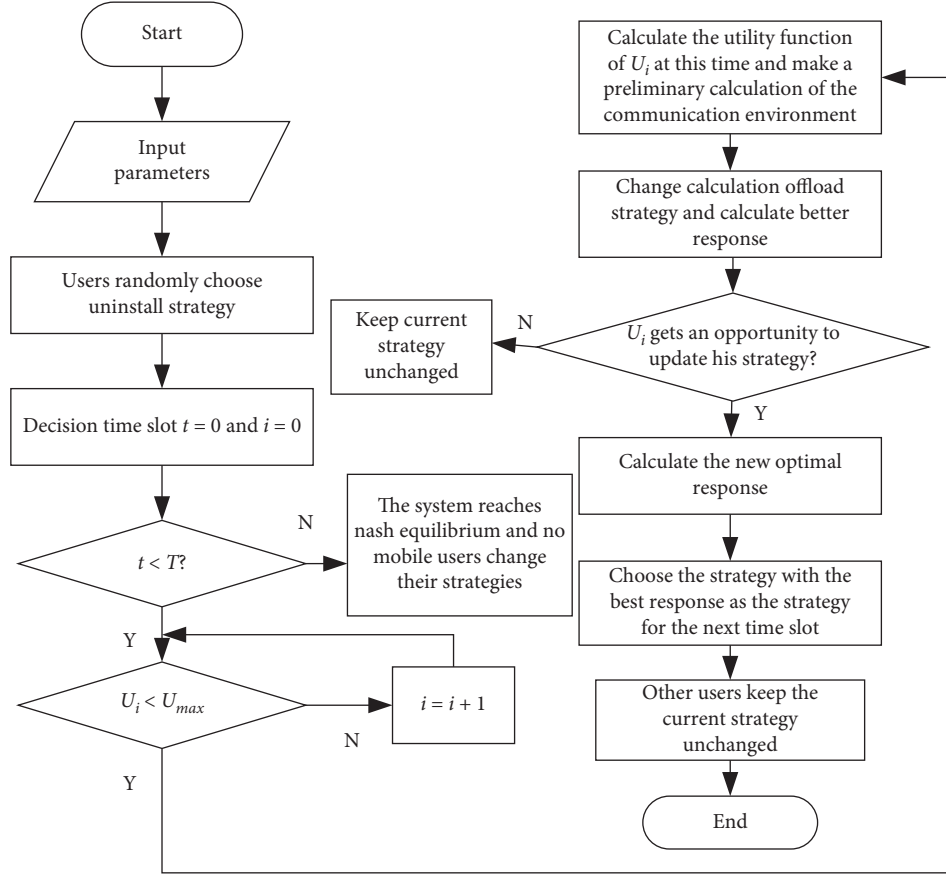


FIGURE 2: Optimal response based computing offload strategy.

TABLE 1: Simulation parameters.

Parameter	Value
Channel bandwidth	1 MHz
Maximum tolerated delay of the task	500 ms
The size of the incoming data	CN (1000, 50)
The computing power of CP	CN (1, 0.05)
D2D transmission power	20 dBm
Noise power spectral density	-174 dBm/Hz
The power of CPU	0.2 w
The computing power of CR	U (0.1, 5)

is assumed that the data size of the calculation task of each mobile device is the same, the external environment is consistent, and only the background noise power spectral density value is changed.

It can be seen from the figure that, as the background noise power spectral density increases, the energy consumption of the distributed task offload algorithm also increases accordingly. This is mainly because the mobile device is affected by background noise at this time, and it needs to spend more computing resources to process, so the energy consumption is higher. In the proposed scenario, although increasing the background noise power spectral density value will increase the system energy consumption, its improvement will significantly reduce the execution time of the computing task and greatly

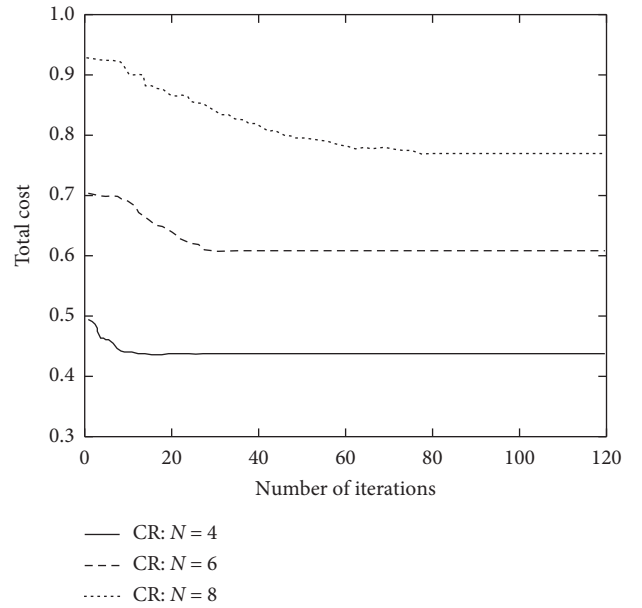


FIGURE 3: Convergence of the algorithm.

reduce the delay of the entire computing offload. For some applications that are more sensitive to delay, it can be appropriate to increase energy consumption in exchange

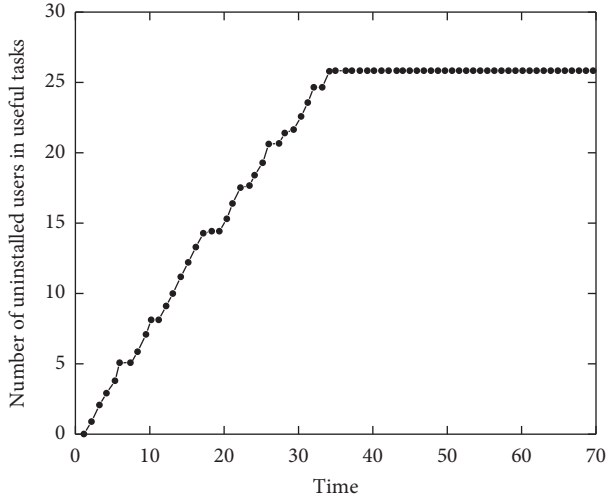


FIGURE 4: Changes in the number of users with beneficial uninstall status.

for the lower delay to ensure user QoS. In practical applications, the background noise power spectral density needs to be adjusted according to the user's different computing tolerances for delay and energy consumption. In the case of meeting the user's minimum delay requirements, reduce the system energy consumption as much as possible.

5.3. Performance Comparison with Other Strategies. This paper compares the proposed MEC-D2D joint offloading algorithm based on optimal response with the algorithms in [23, 28, 29]. The results are shown in Figure 7. Reference [28] is to uninstall all uninstall tasks to the MEC server. Reference [23] uninstalls all uninstall tasks to idle terminals, namely, D2D devices. Reference [29] randomly uninstalls the uninstall task to the MEC or D2D device.

It can be seen from the figure that the proposed algorithm combines MEC-D2D and uses game theory to select the optimal solution to offload the computing task. In the same number of iterations, the function value is the largest; that is, it can handle more offload tasks. References [23, 28] offload all computing tasks to MEC or D2D, and the processing effect is not good, especially the offloading ability of [23] is very limited. However, reference [29] uses random unloading and lacks an effective unloading algorithm, so it saturates the area in fewer iterations.

The average delay of the system with a different number of users is shown in Figure 8.

As can be seen from Figure 8, the average delay of the system increases with the increase of the number of users. However, compared with other offloading strategies, the proposed strategy has lower system delay, which is because it uses game theory to offload tasks. Considering the load of MEC and base station, the proposed strategy offloads tasks with the optimal offload decision and offload rate, which makes full use of local computing resources, balances the traffic load, avoids the extra delay overhead in congestion, and effectively reduces the cost. The system delay is reduced.

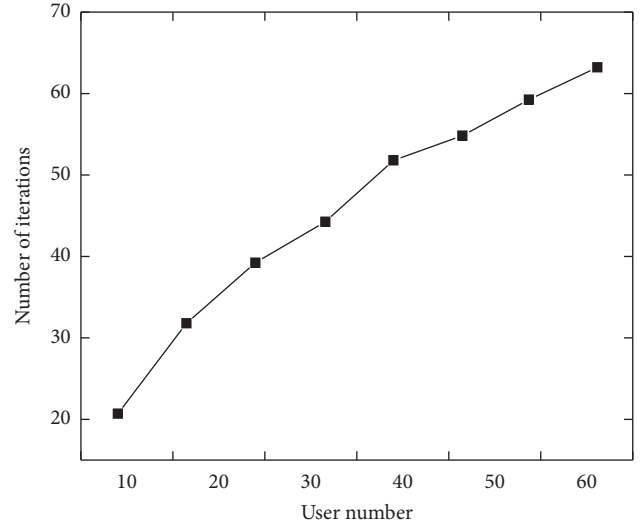


FIGURE 5: The number of iterations needed for the system to reach Nash equilibrium.

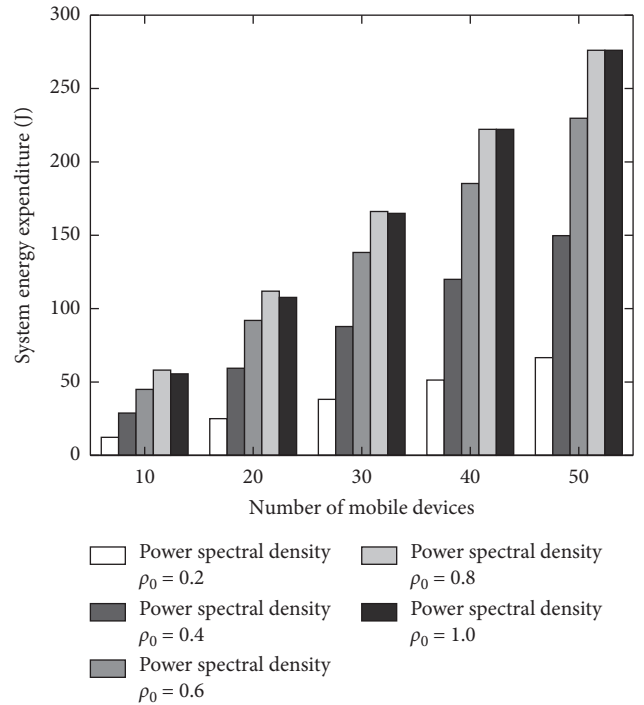


FIGURE 6: Influence of different background noise power spectral density on system energy consumption.

When the mobile user is 1000, the average delay is only 14 ms.

When the number of users is different, the overall system load in different schemes ([23, 28, 29]) is shown in Figure 9.

It can be seen from the figure that reference [23] has the largest overall system load, while other algorithms reduce the overall system load compared to the algorithm in [23]. This shows that offloading tasks to MEC for execution can bring obvious benefits to users. Among the other offloading strategies, the proposed multiuser

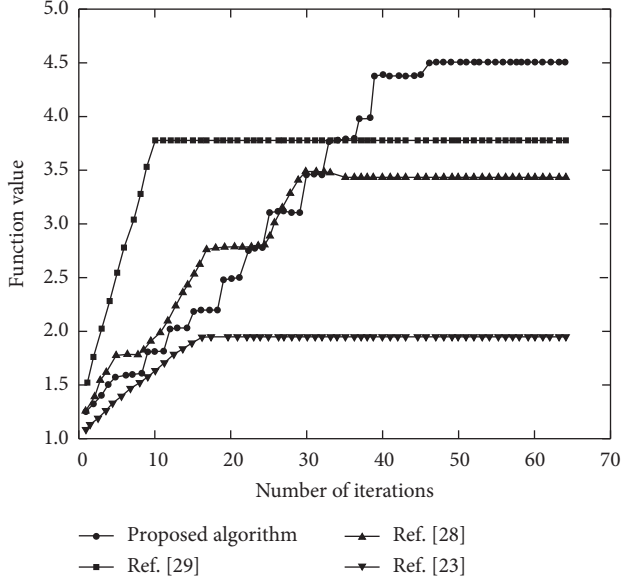


FIGURE 7: Comparison of MEC-D2D unloading algorithm based on optimal response with other algorithms.

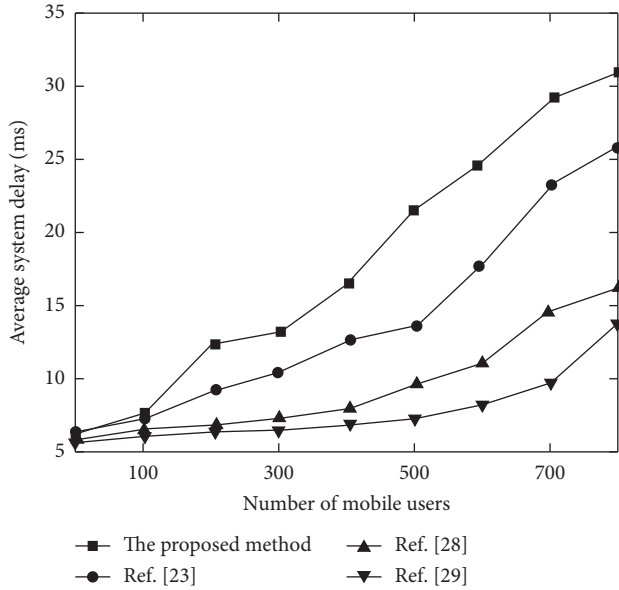


FIGURE 8: Average delay of the system with a different number of users.

distributed task offloading algorithm has the lowest overall system load, which reduces the system load by 67.5% on average compared with [23], and the performance improvement effect is obvious. Compared with

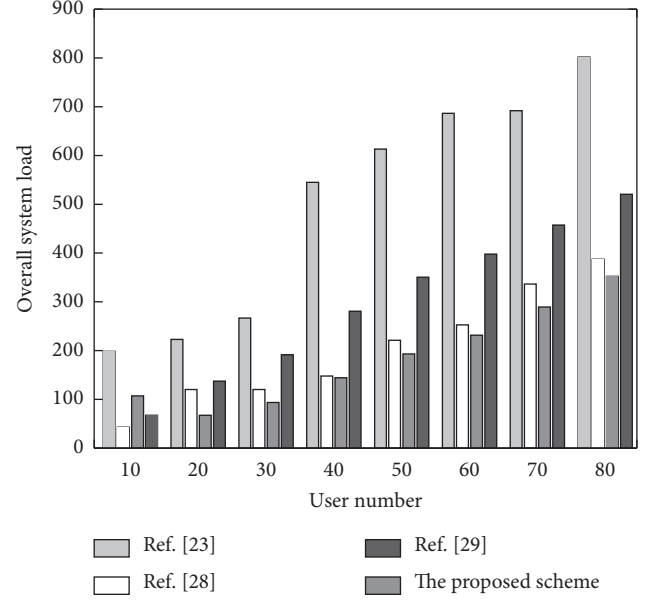


FIGURE 9: Average overall load of the system under different schemes.

[28], because it offloads all tasks to the cloud for execution, without considering the mutual influence of users, some offloading does not bring performance improvement. The comparison in [29] ignores the waiting delay in the MEC server, so the average overall load is higher than the proposed algorithm.

6. Conclusion

In this paper, a new MEC system based on device-to-device communication is designed and modeled, including a communication model, task model, and calculation model. Then, this paper constructs a utility function that takes the unloading strategy and power as constraints, maximizes the number of CPU cycles, and minimizes energy consumption, and then we use game theory to solve the unloading problem. Mobile users choose a computing offloading solution that suits their situation based on factors such as the number of computing offload tasks, transmission rate, and performance of idle devices.

In addition, there are more areas of concern and in-depth research on task offloading strategies in mobile edge computing. In the future, in-depth research can be conducted from the following aspects:

- (1) To simplify the offloading and caching model, the scenarios considered are all single MEC servers. In

the future, more application scenarios should consider the multiserver multiuser scenario and increase the number of servers and users, so that the system model is more complete.

- (2) Information security is not considered in the transmission of massive data. If it is attacked by a network, it will have a great impact on the unloading of computing. Therefore, we will consider encrypting the data in a later stage to ensure the reliability of information.

Data Availability

No data were used to support this study.

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 Program for Innovative Research Team (in Science and Technology) in University of Henan Province (17IRTSTHN009).

References

- [1] Y. Chen, M. Zhao, H. Xia, X. Jin, Z. Wang, and Z. Yu, "A method for extracting high-quality core data from edge computing nodes," *Mathematical Problems in Engineering*, vol. 2019, Article ID 3834846, 10 pages, 2019.
- [2] 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. 9, pp. 193–202, 2019.
- [3] W. Li, X. You, Y. Jiang, J. Yang, and L. Hu, "Opportunistic computing offloading in edge clouds," *Journal of Parallel and Distributed Computing*, vol. 123, no. 1, pp. 69–76, 2019.
- [4] Q. Zhu, "Stabilization of stochastic nonlinear delay systems with exogenous disturbances and the event-triggered feedback control," *IEEE Transactions on Automatic Control*, vol. 64, no. 9, pp. 3764–3771, 2019.
- [5] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.
- [6] W. Hu, Q. Zhu, and H. R. Karimi, "Some improved razumikhin stability criteria for impulsive stochastic delay differential systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 5207–5213, 2019.
- [7] T.-D. Nguyen, E.-N. Huh, and M. Jo, "Decentralized and revised content-centric networking-based service deployment and discovery platform in mobile edge computing for IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4162–4175, 2019.
- [8] 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.
- [9] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic computation offloading in edge computing for Internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4242–4251, 2019.
- [10] C.-C. Chang, W.-K. Lee, Y. Liu, B.-M. Goi, and R. C.-W. Phan, "Signature gateway: offloading signature generation to IoT gateway accelerated by GPU," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4448–4461, 2019.
- [11] S. Yang, C. Xu, X. Qiu, and D. O. Wu, "Diffusion kalman filter with quantized information exchange in distributed mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4423–4435, 2019.
- [12] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4188–4200, 2019.
- [13] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, and W. Dou, "Become: blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4187–4195, 2020.
- [14] T. Wan, Y. Karimi, M. Stanacevic, and E. Salman, "AC computing methodology for RF-powered IoT devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1017–1028, 2019.
- [15] X. Cheng, F. Lyu, W. Quan et al., "Space/aerial-assisted computing offloading for IoT applications: a learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.
- [16] F. Zhou and R. Q. Hu, "Computation efficiency maximization in wireless-powered mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3170–3184, 2020.
- [17] S. S. Gill, P. Garraghan, and R. Buyya, "Router: fog enabled cloud based intelligent resource management approach for smart home IoT devices," *Journal of Systems and Software*, vol. 154, no. 8, pp. 125–138, 2019.
- [18] S. Shen, Y. Han, X. Wang et al., "Computation offloading with multiple agents in edge-computing-supported IoT," *ACM Transactions on Sensor Networks*, vol. 16, no. 1, pp. 1–27, 2019.
- [19] F. Zhou, Y. Wu, R. Q. Hu et al., "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.
- [20] W. Wu, F. Zhou, R. Q. Hu, and B. Wang, "Energy-efficient resource allocation for secure noma-enabled mobile edge computing networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 493–505, 2020.
- [21] H. Guo and J. Liu, "UAV-enhanced intelligent offloading for internet of things at the edge," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2737–2746, 2020.
- [22] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 726–738, 2019.
- [23] X. Hu, K. K. Wong, and K. Yang, "Wireless powered cooperation-assisted edge computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2375–2388, 2019.
- [24] Y. Wu, L. P. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 3, pp. 392–407, 2019.
- [25] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing

- using a deep sequential model based on reinforcement learning," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 64–69, 2019.
- [26] C. Wu, Y. Zhang, and Y. Deng, "Toward fast and distributed computation migration system for edge computing in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10041–10052, 2019.
- [27] J. A. Cordero and W. Lou, "Take your time, get it closer: content dissemination within mobile pedestrian crowds," *Wireless Networks*, vol. 25, no. 6, pp. 3385–3403, 2019.
- [28] S. Wan, X. Li, Y. Xue, W. Lin, and X. Xu, "Efficient computation offloading for Internet of Vehicles in edge computing-assisted 5G networks," *The Journal of Supercomputing*, vol. 76, no. 4, pp. 2518–2547, 2020.
- [29] Y. He, L. Ma, R. Zhou, C. Huang, and Z. Li, "Online task allocation in mobile cloud computing with budget constraints," *Computer Networks*, vol. 151, no. 3, pp. 42–51, 2019.
- [30] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2019.
- [31] H. Wei, H. Luo, Y. Sun, and M. S. Obaidat, "Cache-aware computation offloading in IoT systems," *IEEE Systems Journal*, vol. 14, no. 1, pp. 61–72, 2020.
- [32] Q. Li, J. Hou, S. Meng, and H. Long, "Glide: a game theory and data-driven mimicking linkage intrusion detection for edge computing networks," *Complexity*, vol. 2020, Article ID 7136160, 18 pages, 2020.
- [33] Z. Xiao, X. Dai, H. Jiang et al., "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2038–2052, 2020.
- [34] 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.
- [35] A. Khalili, S. Zarandi, and M. Rasti, "Joint resource allocation and offloading decision in mobile edge computing," *IEEE Communications Letters*, vol. 23, no. 4, pp. 684–687, 2019.
- [36] S. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Energy efficiency and delay tradeoff for wireless powered mobile-edge computing systems with multi-access schemes," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1855–1867, 2020.
- [37] L. Kuang, S. Tu, Y. Zhang et al., "Providing privacy preserving in next POI recommendation for Mobile edge computing," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–11, 2020.
- [38] F. Zhang, J. Ge, C. Wong et al., "Online learning offloading framework for heterogeneous mobile edge computing system," *Journal of Parallel and Distributed Computing*, vol. 128, pp. 167–183, 2019.
- [39] R. Xie, X. Jia, and K. Wu, "Adaptive online decision method for initial congestion window in 5G mobile edge computing using deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 389–403, 2020.
- [40] Y. Wei, Z. Wang, D. Guo, and F. Yu, "Deep q-learning based computation offloading strategy for mobile edge computing," *Computers, Materials & Continua*, vol. 59, no. 1, pp. 89–104, 2019.