IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 3, 2019, accepted December 12, 2019, date of publication December 19, 2019, date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2960887

Dynamic Computation Offloading Based on Graph Partitioning in Mobile Edge Computing

GUANGSHUN LI[1,2], QINGYAN LIN[1], JUNHUA WU[1], YING ZHANG[1], AND JIAHE YAN[1]
[1]School of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China
[2]Department of Computer, The Hong Kong Polytechnic University, Hong Kong

Corresponding author: Junhua Wu (shdwjh@163.com)

This work was supported by the National Natural Science Foundation of China (61672321, 61771289, and 61832012), Shandong province key research and development plan (2019GGX101050), Shandong provincial Graduate Education Innovation Program (SDYY14052 and SDYY15049), Qufu Normal University Science and Technology Project (xkj201525), Shandong province Agricultural Machinery Equipment Research and Development Innovation Project (2018YZ002), Shandong Provincial Specialized Degree Postgraduate Teaching Case Library Construction Program, and Shandong Provincial Postgraduate Education Quality Curriculum Construction Program.

ABSTRACT Mobile edge computing is a new cloud computing paradigm that utilizes small-sized edge clouds to provide real-time services to users. These mobile edge clouds (MECs) are located near users, thereby enabling users to seamlessly access applications that are running on MECs and to easily access MECs. Terminal devices can transfer tasks to MEC servers nearby to improve the quality of computing. In this paper, we study multi-user computation offloading problem for mobile-edge computing in a multichannel wireless interference environment. Then, we analyze the overhead of each mobile devices, and we propose strategies for task scheduling and offloading in a multi-user MEC system. For reducing the energy consumption, we propose a server partitioning algorithm that is based on clustering. We formulate the task offloading decision problem as a multi-user game, which always has a Nash equilibrium. The simulation results demonstrate that our scheme outperforms the traditional offloading strategy in terms of energy consumption.

INDEX TERMS Mobile edge computing, offloading decision, node clustering, optimal strategy, Nash equilibrium.

I. INTRODUCTION

The growing popularity of mobile devices, such as smart phones, tablet computers and wearable devices, is accelerating the advent of the Internet of things (IoT) and triggering a revolution of mobile applications [1]. The IoT as attracted substantial research attention; it is considered part of the Internet of the future and will comprise billions of intelligent communicating ‘things’ [2]. With the ever-increasing popularity of mobile computing technology, a wide range of computational resources and services are migrating to the mobile infrastructure and devices [3]. As an important component of strategic emerging industries, the IoT has promoted the transformation of production, life and social management methods into intelligent, refined and networked methods. If your paper is intended for a conference, please contact your conference editor concerning acceptable word processor formats for your particular conference.

The Internet of things is increasingly being applied in every field, and the number of devices that are connected to the global Internet continues to grow [4], [5]. The growing popularity of smart mobile devices has driven the development of mobile cloud computing (Mcc), and the emergence of Mcc has reduced the cost of developing mobile applications [6], [7]. Therefore, cloud computing has become the overall method of choice for centralized information storage and management, and mobile devices have become the main destinations of information [8], [9]. Through the interconnection of cloud computing and mobile devices, resources such as online applications and network infrastructure can be shared through the Internet.

As functions of cloud computing increasingly move to the edge of the network, a new trend of computing has emerged: It is estimated that tens of billions of edge computing devices will be deployed on the edge of the network [10]. In this new environment, we must manage, process, and store the large amounts of data that are generated at the edge of the network [11].

By collecting large amounts of free computing power and storage space on the edge of the network, it is possible to generate sufficient capacity for performing computationally intensive and delay-sensitive tasks on mobile devices [12].

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

This mode is called mobile edge computing (MEC). Facilitated by MEC, mobile devices can offload their tasks to the MEC servers on the edge of the network rather than utilizing the servers in the core network. This MEC paradigm can provide low latency, high bandwidth and computing agility in the computation offloading process [13], [14]. MEC is a novel paradigm that extends cloud computing capabilities and services to the edge of the network. Due to its dense geographical distribution, proximity to users, support for high mobility, and open platforms, MEC can support applications and services with reduced latency and improved QOS [15]. Thus, it is becoming an important enabler of user-centric IoT applications and services that demand real-time operations.

II. RELATED WORK

In edge computing, tasks can be regarded as resource consumers, including traditional computing tasks and noncomputing tasks. The time and overhead of these subtasks depend on the resources on which they are completed [16]. Therefore, in the fog computing environment, to satisfy the user's rapid response (performance) requirements while reducing the cost to the user, it is necessary to reasonably schedule tasks on various resources. In the past few years, in parallel to the ETSI MEC ISG initiative and to the Open Fog Consortium, MEC has emerged as a promising research area [17]. In this section, we present related works to the problem of MEC resource dimensioning.

Wang and Yuan et al. [18] describe the problem of energy-aware edge server placement as a multiobjective optimization problem and devise a particle-swarm-optimization-based energy-aware edge server placement algorithm for identifying the optimal solution. Zeng et al. [19] expressed the task scheduling problem as a mixed-integer nonlinear programming problem, overcame its high computational complexity, and proposed a computationally effective solution. Song and Gao et al. [20] used cloud atomization technology to transform physical nodes into virtual machine nodes. Based on graph partitioning theory, a load balancing algorithm for fog computing that uses dynamic graph partitioning is proposed, which dynamically balances the load, effectively allocates resources, and reduces the overhead that is caused by new nodes. Mebrek et al. [21] used the energy and the quality of service (QoS) as two important indicators of fog performance. They express the problem as a constrained optimization problem and use the evolutionary algorithm (EA) to effectively solve the scheduling problem. Chen et al. [22] describe multi-user multitask unloading as an NP-hard problem and use the separable semi-deterministic relaxation problem to identify the lower bound of the system overhead and to realize the optimal performance of the system under multiparameter conditions. Qi et al. [23] considers the user's job size, service invocation time, and service quality level, and a set of experiments are designed, deployed, and tested to validate the feasibility of our proposed approach in terms of cost optimization. The experiment results demonstrate that the CS-COM method outperforms other related methods. To improve the energy efficiency in the cloud environment, they also designed a QoS-aware VM scheduling method for energy conservation, namely, QVMS [24]. Chen and Liang et al. [25] consider a general multi-user mobile cloud computing system in which the mobile users share the communication resources while offloading tasks to the cloud. They describe the optimization problem as a nonconvex quadratic constrained quadratic program. Via the separable semidefinite relaxation and recovery of the binary unloading decision and the optimal allocation of communication resources, an effective approximate solution is proposed. Xu et al. [26] study the cloudlet placement problem in a large-scale wireless metropolitan area network. They consider placing multiple cloudlets of various computing capacities at strategic locations to reduce the latency. They show that the problem is NP-hard and propose a fast and scalable heuristic solution. Zhu et al. [27] considers a much more complex scenario in which multiple moving MDs share multiple heterogeneous MEC servers. A problem, namely, the minimum energy consumption problem, in a deadline-aware MEC system is formulated, and two approximation algorithms are proposed. The experimental results demonstrate that these two algorithms realize superior performance in terms of energy consumption.

III. THE DESCRIPTION OF MEC

Mobile edge computing (MEC) can be defined as an implementation of edge computing that introduces computational and storage capacities into the edge of a radio access network, thereby reducing the latency by moving the cloud and the service platform to the edge of the network. [28]. As illustrated in Fig. 1, the MEC server node is the core part of the mobile edge computing network and the main implementation node for the mobile edge computing. In the mobile edge computing system, the edge server is connected to the core network through a wired link, and the edge server can be a mobile device. The IoT device provides a computing task migration service, in contrast to mobile cloud computing [29]. At this time, most of the computing tasks can be processed at the edge nodes without entering the cloud core network; hence, the computing and communication loads of the cloud core network can be significantly reduced. Redundant computing resources at the edge of the network can also be fully utilized, and mobile devices and IoT devices can realize lower computational task completion delays.

IV. GRAPH-BASED ENERGY-CLUSTERING ALGORITHM

An edge computing network consists of edge devices that are deployed around the Internet of things. Tasks and resources often change dynamically because the physical nodes in the edge calculation often join or exit nodes. Therefore, to solve the task allocation and scheduling problems in edge computing, we must also consider the dynamic changes of the resources and tasks [30]. MCC resources are abstracted into services with specified functions and parameters. Due to the differences among the heterogeneous resources, in the process of task scheduling, suitable resources should be

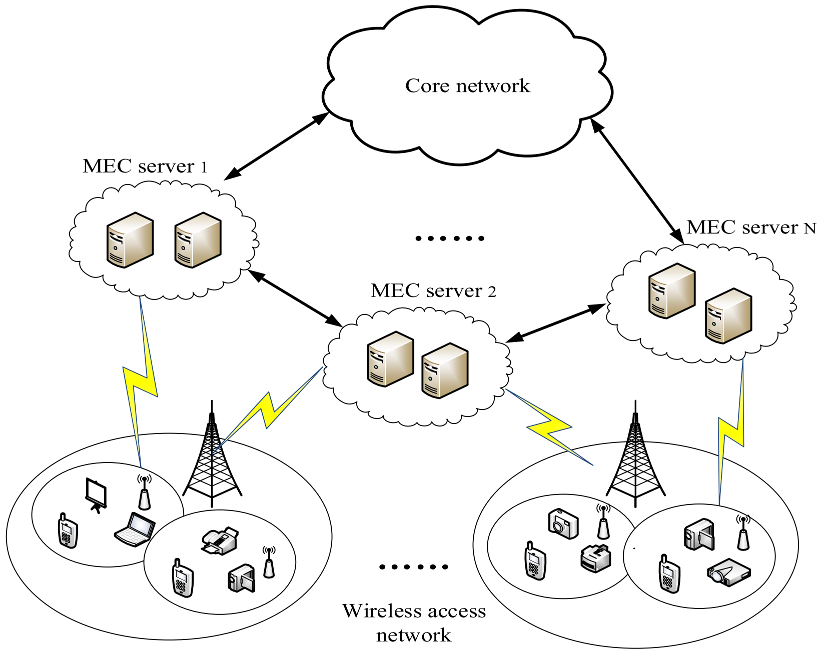


FIGURE 1. MEC deployment architecture diagram.

selected for each task according to the global optimization objectives.

As illustrated in Figure 1, the mobile edge computing network consists of three layers of architecture: a data center level, an edge server level and a user device level. We focus on the edge server level and the user device level.

Now, we introduce the system model of the mobile edge computing system. We assume that there are mobile devices in which each user has a computationally intensive task to complete. Each device can connect directly to its neighbors via wifi. We use a tuple $\langle res_i, E_i, N_i \rangle$ to represent a mobile device node, in which res_i is the computing resource of node i, E_i is the energy of node i, N_i is the number of neighboring nodes of node i, and the neighboring nodes can communicate with node i. We use a tuple $\langle B_i, K_i, L_i \rangle$ to represent task I, for $I \in N$, in which B_i is the input data size (in bits), K_i is the amount of computation that is required for the task, and L_i is the maximum acceptable delay for the task, where for a real-time task, we set $L_i = 0$, for delay-insensitive jobs, we set $L_i = \infty$, and otherwise we set it to the actual maximum delay. Important notations that are used in this paper are summarized in Table 1.

In this paper, the network topology diagram that is composed of the physical nodes of the edge computing network are abstracted into an undirected weighted connected graph, where

$$V = \{v_1, v_2, v_3, \ldots, v_i, \ldots, v_n\} \tag{1}$$

is a vertex set, in which v_i is a mobile device and n is the number of devices;

$$E = \left\{e_{ij} | i, j \in [1, 2 \cdots n]\right\} \tag{2}$$

is an edge set, in which e_{ij} is the communication link between edge nodes v_j and v_j; and τ_{ij} is the weighted sum of the communications between nodes v_i and v_j.

TABLE 1. Notations.

symbol	definition
v_i	mobile device i
e_{ij}	the communication link between edge nodes
τ_{ij}	the weighted sum of the communications between nodes
$M_{\max}$	the cluster maximum weight
$res_{\max}$	the cluster maximum resource
a_i, b_i, c_i	measurements of the factors
T_m^n, T_i^m	the transmission delay and execution time
E_i, E_{trans}	the computing energy consumption and the energy consumption during transmission
α_l, λ_l	weighting factors
E_n^{edge}, $E_{n,trans}^c$, $E_{n,com}^c$	the total energy consumption, the transmission energy consumption and the calculation energy consumption

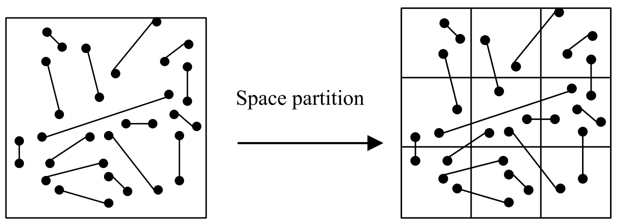


FIGURE 2. Graph of nodes in the area.

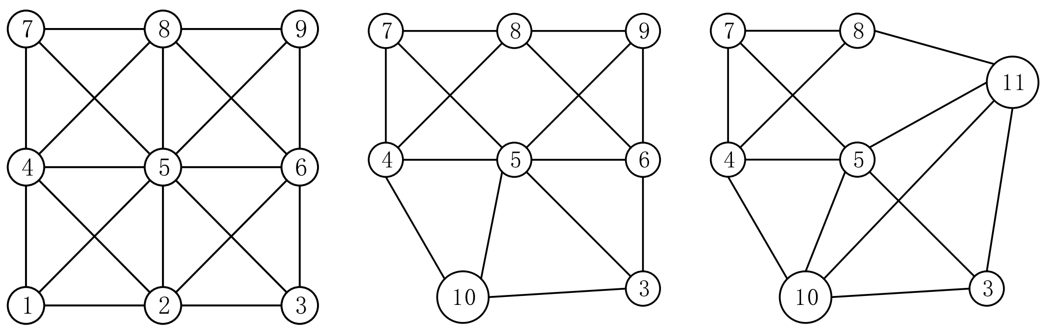


FIGURE 3. Adjacencies of the nodes $G_a = (C, E_a)$.

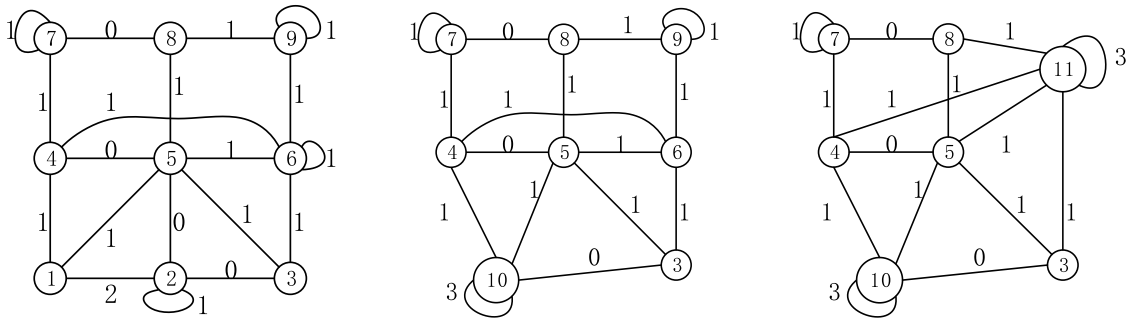


FIGURE 4. Interaction graph $G_{int} = (C, E_{int})$.

We studied Bouet's [31] MEC-based deployment algorithm. From a network system standpoint, we consider a mobile device deployment, as presented in Figure 2. The first graph represents the region where the node is located, and the second graph represents the discretized distribution of the nodes after space partitioning. Now, we will use the node set as the starting point for node partitioning.

Figure 3 presents the adjacencies of the nodes. For instance, in a square grid, a node (a grid cell) has up to 8 adjacent nodes. A node can communicate with the other eight nodes. In a collection, nodes can self-loop. Figure 4 presents the interactions (the communications or the traffic) between the nodes in the area.

We assume that the considered area has been discretized into N cells. Let G denote the set of the N cells. C is a partition of the set G. The objective is to cluster the area cells. A cluster has the largest resource limit $res_{\max}$ and it also has a maximum capacity in terms of the traffic or communications per unit of time that can be processed at its MEC server, which is denoted as $M_{\max}$.

(1) First, we identify the edge with the highest weight and determine the amount of resources for each node. The highest weight between node i and node j is less than the cluster maximum weight $M_{\max}$, and the sum of the resources of node i and node j is less than the cluster maximum resource $res_{\max}$.

(2) Then, we cluster node i and node j, and we update the graphs G_a and G_{int} with a new node C_{ij} that represents their clustering. The neighboring nodes of the formed cluster C_{ij} are the neighboring nodes of node i and node j. The weight of the link between the new node and its neighbor is determined by the sum of the link weights between the former node i and its neighbors and between the former node j and its neighbors. The weight $\tau_{c,ij}$ of the new cluster corresponds to the sum of the two former self-loops plus the weight between node i and node j.

$$\tau_{c,ij} = \tau_{ij} + \tau_{ji} + \tau_{ii} + \tau_{jj}$$
$$\tau_{c,ij} \leq M_{\max}$$

By construction, the number of nodes (clusters) is reduced in each iteration. These iterations are conducted until there are no more changes, namely, until the local minimum of the mobile device cluster interaction is attained.

In the above system model, the cluster nodes select cluster heads via clustering, and the cluster heads represent the cluster nodes for sending and receiving information. A node can utilize many strategies for executing jobs. In this paper, we mainly consider the job execution quality a_i, the device energy b_i, and the device cost c_i, and their respective weights are expressed in binary form. The node cluster selects the job strategy to be executed according to its own performance. For example, when the performance of the node cluster is described as $a_i = 1$, $b_i = 0$, and $c_i = 0$, then this node cluster selects jobs that require high-quality completion. If a node cluster's performance is described by $a_i = 0$, $b_i = 1$, and $c_i = 0$, then this node cluster selects a job with less energy consumption; for the same reason, if a node's performance is described by $a_i = 0$, $b_i = 0$, and $c_i = 1$, then this node chooses to have its own job performed by other nodes. A multitarget strategy can be implemented by setting various values of i.

V. SYSTEM MODEL AND PROBLEM DESCRIPTION

A. SYSTEM MODEL

1) LOCAL COMPUTING MODEL

Each mobile device i can execute its own task locally. Suppose that the computational capability of a mobile device is F_i. The communication delay T_m^n mainly consists of the transmission delay of tasks that are transmitted through wireless network and the execution time T_i^m of computing tasks at nodes. When a task is executed at a local layer, the execution time T_i^m is expressed as

$$T_i^m = \frac{K_i}{F_i} \tag{3}$$

The energy consumption of task i at the local device is

$$E_i^m = c_n T_i^m \tag{4}$$

where c_n is the CPU power of the local mobile device. Therefore, the transmission delay can be expressed as

$$T_m^n = \frac{K_i}{w\log(1+\frac{p_m g_{mn}}{\omega_0})} \tag{5}$$

The total time delay must be less than the maximum acceptable delay for the task: $T_i^m + T_i^n \leq L_i$.

If the task must be transmitted to another device for execution, the device chooses a channel, and the energy consumption during transmission is $E_{trans}^{mn} = r_m^n \times K_i$, while the total energy consumption is

$$E_i^{mn} = E_{trans} + E_i^n \tag{6}$$

where E_{trans} is the energy that is consumed by the transmission task process and E_i^n is the energy that is consumed by the other device in executing the task. x_{ij} is equal to 1 if the task is transferred to another device for processing.

Therefore, the total overhead is

$$\begin{aligned} \min \sum_{i=1}^{n} \Bigg[& \sum_{j=1}^{n} \alpha_l \left[E_i^m x_{ij} + E_i^{mn}(1-x_{ij})\right] \\ & + \lambda_l \left[T_i^m x_{ij} + T_i^n(1-x_{ij})\right] \Bigg] \\ s.t\ \ & i \in (1,n), j \in (1,n) \\ & T_i^m + T_i^n \leq L_i \end{aligned} \tag{7}$$

Hence, the total overhead of local execution on mobile devices can be expressed as $Z_n^l = \alpha_l E_n + \lambda_l T_n$ where α_l and λ_l are two weighting factors that correspond to the weights of the time consumption and the consumption in the decision-making process, respectively. To meet the specified demands of mobile devices, mobile devices can choose their weight factors. In the decision-making process, if a mobile device is in a low-battery state, to conserve more energy, it would choose a larger value of λ_l and put more weight on energy consumption. If a mobile device is running a delay-sensitive application, to reduce the time delay, the device would choose a larger value of α_l and put more weight on the execution time. In practice, suitable weights that capture a user's valuations on computational energy and time can be determined by applying the multiattribute utility approach in multiple-criteria decision-making theory [32].

$$\begin{cases} \alpha_l & +\lambda_l = 1 \\ 0 & \leq \alpha_l \leq 1 \\ 0 & \leq \lambda_l \leq 1 \end{cases} \tag{8}$$

2) MEC COMPUTING MODEL

$$\varphi_{ij} = \begin{cases} 0 & \text{Task excute at local mobile device} \\ 1 & \text{Task excute at MEC device} \end{cases} \tag{9}$$

If $\varphi_{ij} = 1$, tasks must be uninstalled from the MEC device for execution, and the total energy consumption E_n^{edge} can be divided into the transmission energy consumption $E_{n,trans}^c$ and the calculation energy consumption $E_{n,com}^c$:

$$E_n^{edge} = E_{n,trans}^{edge} + E_{n,com}^{edge} \tag{10}$$

$$E_{n,trans}^{edge} = P_i \bullet T_{i,trans}^e, \forall i \in N \tag{11}$$

P is the unit energy consumption when the mobile device accesses the channel, and $E_{n,com}^c$ is the energy that is consumed in task execution at the MEC layer.

$$E_{n,com}^{edge} = T_{i,com}^e \bullet \varepsilon_e \tag{12}$$

F_c is the computing power of the edge cloud server, and ε_e is the energy consumption of the edge cloud server per unit time. The communication delay mainly consists of the transmission delay $T_{i,trans}^e$ and the execution time $T_{i,com}^e$ of the computing tasks.

$$T_{i,trans}^e = \frac{K_i}{r_n} \tag{13}$$

$$T_{i,com}^e = \frac{K_i}{F_c} \tag{14}$$

Therefore, the total overhead of MEC is

$$\min \sum_{i=1}^{n} \alpha_c \left[E_{n,trans}^{edge} + E_{n,com}^{edge}\right] + \lambda_c \left[T_{n,trans}^e + T_{n,com}^e\right] \tag{15}$$

Hence, the total overhead of MEC server execution can be expressed as

$$Z_n^c = \alpha_c E_n^{edge} + \lambda_c T_n^{edge} \tag{16}$$

B. NASH EQUILIBRIUM

Based on the computation and communication models in Section 5.1, we conclude that if too many mobile devices choose the same channel for task unloading, the interference among them will become highly severe, thereby resulting in a lower data rate between each mobile device and the base station and a higher time cost in uploading task data. Spending too much time on uploading tasks can lead to higher energy consumption of the mobile devices. In this case, performing the tasks locally without task offloading would be more beneficial.

When mobile device m considers offloading its task to mobile device n, all task scheduling decisions of these devices in ϕ_n ($\phi_n \geq 0$) affect device i. Therefore, the task scheduling decision

$$\phi_{-n} = \{\phi_1, \phi_2, \ldots, \phi_{n-1}, \phi_{n+1}, \ldots \phi_N\} \tag{17}$$

represents the unloading decisions of other users except user i. If the unloading decision of user n is specified, the multi-user task unloading decision can be expressed as

$$\Gamma = \left(N, \{\phi_n\}_{n\in N}, \{Z_n\}_{n\in N}\right) \tag{18}$$

The objective of our decision-making process in this paper is to minimize the overhead of the task scheduling and unloading process. The objective function can be expressed as:

$$\min_{\phi_n\in\{0,1\}} Z_i\left(\phi_n, \phi_{-n}\right), \quad \forall n \in N \tag{19}$$

$$Z_n(\phi_n^*, \phi_{-n}^*) \leq Z_n(\phi_n, \phi_{-n}^*), \quad \forall \phi_n \in A_n,\ n \in N \tag{20}$$

From the literature [33], it is concluded that there is a Nash equilibrium in the multi-user computing task unloading decision game, and the Nash equilibrium can be attained via finitely many iterations.

$$\Delta_n(t) \left\{ \tilde{\phi} : \tilde{\phi} = \arg\min_{\phi\in A_n} Z_n\left(\phi_n, \phi_{-n}\right) \ and\ Z_n\left(\phi^*, \phi_{-n}(t)\right) < Z_n\left(\phi_n(t), \phi_{-n}(t)\right) \right\} \tag{21}$$

If the calculated $\Delta_n(t)$ is not empty, the mobile device has not reached the Nash equilibrium state.

We consider a node cluster of N participants, where each participant schedules tasks according to a strategy. When other participants utilize optimal strategies, the participants will not change their own strategies any further. When all participants stop changing their strategies, the system as a whole reaches a state of equilibrium. The Nash equilibrium is closely related to the strategies that are adopted by the nodes.

Each cluster node determines a_i, b_i, and c_i according to its own scheduling policy. If the scheduling strategy for each cluster is feasible for the system as a whole, then the status quo is maintained. Following the execution of a job, changes are made in the system. If the strategies are not feasible for the whole system, the function is adjusted to adjust the strategy of the partial node cluster so that the system transitions to a new equilibrium state. The system can support the scheduling policy of each node cluster if the following conditions are satisfied.

We define a threshold for the opportunistic consumption of a mobile device, and only mobile devices for which the opportunistic consumptions are less than the threshold could provide the resource, which aims at avoiding resource consumption when there are not sufficient resources in the mobile device. The threshold is set as $res(i) = 20$.

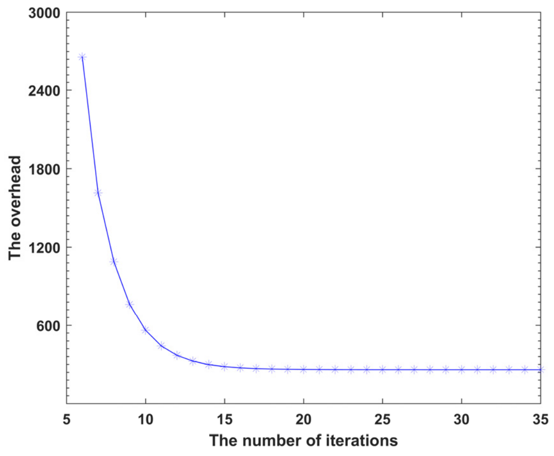


FIGURE 5. **The overhead of reaching Nash equilibrium.**

The total cost of the unloading and uninstallation decisions must satisfy the following restrictions:

$$Z_n\left(\phi_n, \phi_{-n}\right) = \begin{cases} Z_n^l & \phi_n = 0 \\ Z_n^{of} & \phi_n = 1 \end{cases} \tag{22}$$

1) Each mobile device obtains its own parameter values, each mobile device makes the computing decision, and each mobile device in the decision slot receives the parameters of its neighboring mobile devices. At each time t, each device customizes the decision according to formulas (7) and (15).
2) Formula (22) is used to determine whether the Nash equilibrium has been reached.
3) If all the costs reach the minimum value at this time, terminate the iteration process; otherwise, at the next time, namely, t + 1, return to (1) and continue the iteration process.

VI. EVALUATION AND ANALYSIS

In this section, we use the MATLAB software simulation method to evaluate the performance of the proposed unloading strategy. We set up 100 mobile devices in the network, and each device has a task to be processed. Finally, we divide the devices into 10 clusters for the game.

In this simulation, the parameter of the task is set as $B_i = 5MB$, the calculated value of the task is $K_i \sim U(30, 60)MI$, and the time delay constraint is $L_{\max} \sim U(5, 10)s$. The computing power of the mobile devices is $F_n = 5MI/s$, with $(\alpha_l, \lambda_l) = (0.5, 0.5)$. The computing power of the MEC devices is $F_c = 10MI/s$. The parameter of the Wi-Fi wireless channel is set as $r = 10MB/S$.

According to Figure 5, by increasing the number of iterations, a stable state can realized; Hence, the algorithm can reach the Nash equilibrium state within a limited time.

In this paper, we propose a scheme of energy consumption. We compare the energy consumption of our method with those of local computing without task scheduling and edge

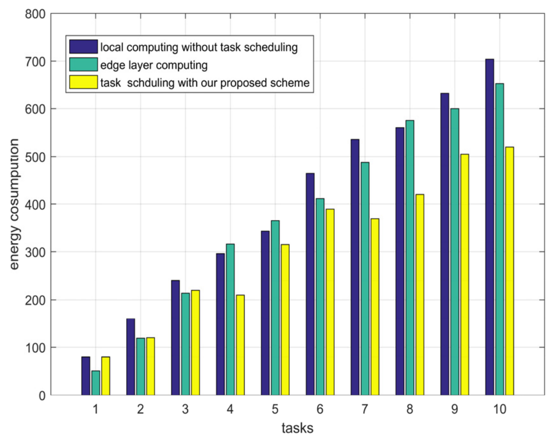


FIGURE 6. The consumption of task owner.

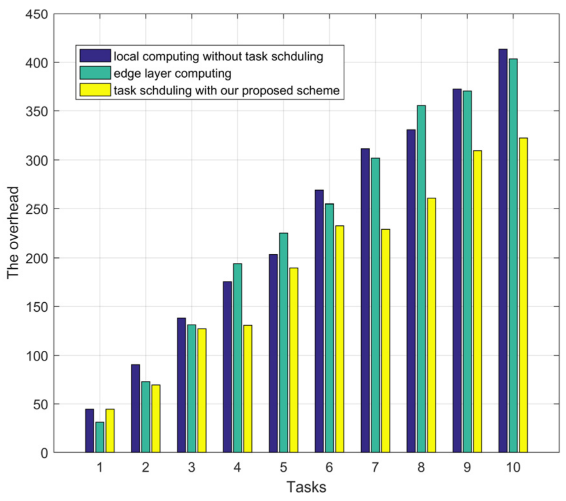


FIGURE 7. The overhead for mobile device.

cloud computing of all tasks [34]. The experimental results are presented in Figure 6.

The experimental results demonstrate that as the number of tasks increases, the local computing task scheduling of our proposed scheme consumes less energy than local computing without task scheduling and edge cloud computing of all tasks. When $I \leq 3$ hours, our proposed solution and all tasks of the edge layer calculation do not differ substantially in terms of energy consumption. When $I > 3$, due to the increased workload, all transmissions to the edge layer will consume a large amount of transmission energy, while the local device is idle. At this time, the game-theory-based scheduling scheme performs better.

From Figure 7, we can see that the local computing without task scheduling has the largest overhead. All the other schemes reduce the overall overhead of the system relative to it, indicating that offloading tasks to the cloud for execution can bring obvious benefits to users. We can also see from the figure, when the number of tasks is small, there is a little difference of the total overhead between our schemes and

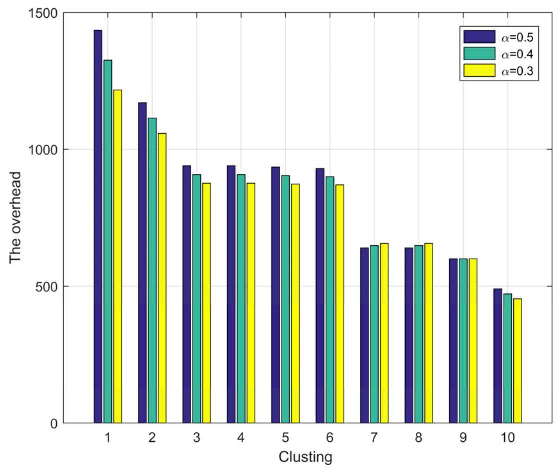


FIGURE 8. The overhead for cluster.

the edge cloud computing. Due to the fast processing speed of the edge cloud, the time delay is small, but the energy consumption is large. However, our schemes has a large time delay and low energy consumption. When the number of tasks is low, the total cost of the two solutions is basically the same. When the number of tasks increases gradually, all tasks may be queued and the communication load is too high when they are offloaded to the edge cloud. This leads to high latency, and the energy consumption of uploading tasks to the edge layer is also increasing at the same time. Our schemes has higher performance.

Figure 8 shows the overhead of the cluster. As the number of device clusters increases, we can see that the total overhead is gradually decreasing. The device cluster reduces the connection between the task and each mobile device during the scheduling process. The cluster head represents the cluster interacting with neighbor clusters and clouds, which effectively reduces the transmission energy consumption of each mobile device. In this paper, we select different values for the weight parameters α. The experimental results show that when α is gradually larger, the proportion of energy consumption in total overhead increases, and in our method, it is more suitable for calculating tasks with low delay requirements.

Figure 9 shows the energy consumption relationship between clusters and the number of tasks. As can be seen from the figure, given a certain number of tasks, when the number of clusters is 1 and 5, the energy consumption is greater than other Numbers. Therefore, in the process of clustering, we need to select appropriate parameters to achieve the optimal clustering effect.

VII. CONCLUSION

This paper analyzes the task scheduling problem based on self-organized edge computing, and proposes a graph-based server region clustering algorithm. The game scheduling based task scheduling mechanism mainly considers energy consumption and aims to minimize mobile devices. In this paper, the communication model and the computational

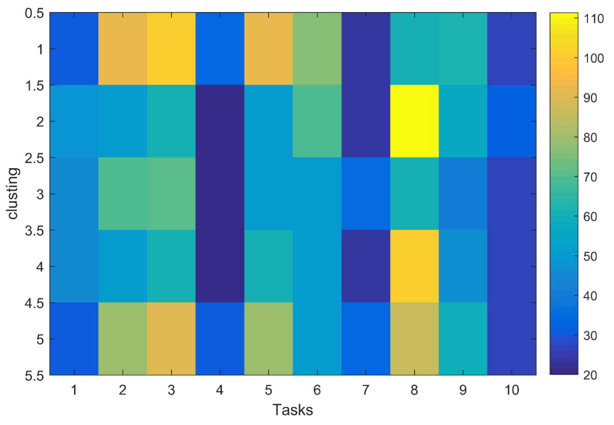


FIGURE 9. The energy consumption.

model are proposed firstly. The graph-based server clustering algorithm is proposed, and the algorithm is applied to the edge computing. Then the task unloading problem is analyzed, which effectively reduces the edge computing task scheduling energy consumption.

For the future work, we are considering extending the task scheduling model with task priority, in which case, the tasks with high priority should be performed first. Second, our analysis ignores the e possibility of task dropping.

REFERENCES

[1] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[2] X. Wang, W. Wang, L. T. Yang, S. Liao, D. Yin, and M. J. Deen, "A distributed HOSVD method with its incremental computation for big data in cyber-physical-social systems," *IEEE Trans. Computat. Social Syst.*, vol. 5, no. 2, pp. 481–492, Jun. 2018.

[3] H. Liu, H. Kou, C. Yan, and L. Qi, "Link prediction in paper citation network to construct paper correlation graph," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, 2019, Art. no. 233, doi: 10.1186/s13638-019-1561-7.

[4] G. Li, J. Yan, L. Chen, J. Wu, Q. Lin, and Y. Zhang, "Energy consumption optimization with a delay threshold in cloud-fog cooperation computing," *IEEE Access*, vol. 7, pp. 159688–159697, 2019.

[5] X. Wang, L. T. Wang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 80–85, Nov. 2017.

[6] H. Li, G. Shou, Y. Hu, and Z. Guo, "Mobile edge computing: Progress and challenges," in *Proc. 4th IEEE Int. Conf. Mobile Cloud Comput., Services, Eng. (MobileCloud)*, Mar./Apr. 2016, pp. 83–84

[7] Y. Yu, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges," *China Commun.*, vol. 13, no. Supplement2, pp. 89–99, 2016.

[8] L. Q. Wenwen Gong and Y. Xu, "Privacy-aware multidimensional mobile service quality prediction and recommendation in distributed fog environment," *Wireless Commun. Mobile Comput.*, vol. 2018, Apr. 2018, Art. no. 3075849.

[9] G. Li, Y. Liu, J. Wu, D. Lin, and S. Zhao, "Methods of resource scheduling based on optimized fuzzy clustering in fog computing," *Sensors*, vol. 19, no. 9, p. 2122, 2019.

[10] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[11] L. Jin, S. Li, J. Yu, and J. He, "Robot manipulator control using neural networks: A survey," *Neurocomputing*, vol. 285, pp. 23–34, Apr. 2018.

[12] S. Shahzadi, M. Iqbal, T. Dagiuklas, and Z. U. Qayyum, "Multi-access edge computing: Open issues, challenges and future perspective," *J. Cloud Comput.*, vol. 6, no. 1, p. 30, 2017.

[13] K. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[14] Y. Xu and L. Qi, "Simhash-based similar neighbor finding for scalable and privacy-preserving service recommendation," *Complexity*, vol. 10603, no. 11, pp. 113–122, 2017.

[15] L. T. Yang, X. Wang, X. Chen, L. Wang, R. Ranjan, X. Chen, and M. J. Deen, "A multi-order distributed HOSVD with its incremental computing for big services in cyber-physical-social systems," *IEEE Trans. Big Data*, to be published.

[16] L. Qi, X. Zhang, W. Dou, and Q. Ni, "A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2616–2624, Nov. 2017.

[17] L. T. Yang, X. Wang, X. Chen, J. Han, and J. Feng, "A tensor computation and optimization model for cyber-physical-social big data," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 4, pp. 326–339, Dec. 2019.

[18] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. T. Zhou, "MEETS: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 4076–4087, Oct. 2018.

[19] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.

[20] N. Song, C. Gong, X. An, and Q. Zhan, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Commun.*, vol. 13, no. 3, pp. 156–164, 2016.

[21] A. Mebrek, L. Merghem-Boulahia, and M. Esseghir, "Efficient green solution for a balanced energy consumption and delay in the IoT-fog-cloud computing," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, Oct./Nov. 2017, pp. 1–4.

[22] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[23] L. Qi, X. Zhang, W. Dou, C. Hu, C. Yang, and J. Chen, "A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment," *Future Gener. Comput. Syst.*, vol. 88, pp. 636–643, Nov. 2018.

[24] Q. H. L. Qi, F. Chen, W. Dou, S. Wan, X. Zhang, and X. Xu, "Finding all you need: Web APIs recommendation in Web of things through keywords search," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 5, pp. 1063–1072, Oct. 2019.

[25] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE Int. Conf. Commun.*, May 2016, pp. 1–6.

[26] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.

[27] T. Zhu, T. Shi, Z. Cai, X. Zhou, and J. Li, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4854–4866, Jun. 2019.

[28] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[29] Q. V. Pham, T. Leanh, N. H. Tran, and C. S. Hong, "Decentralized computation offloading and resource allocation in heterogeneous networks with mobile edge computing," 2018.

[30] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.

[31] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A geo-clustering approach," *IEEE Trans. Netw. Service Manag.*, vol. 2, no. 2, pp. 787–796, Jun. 2018.

[32] J. Wallenius, J. S. Dyer, P. C. Fishburn, R. E. Steuer, S. Zionts, and K. Deb, "Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead," *Manage. Sci.*, vol. 54, no. 7, pp. 1336–1349, Jul. 2008.

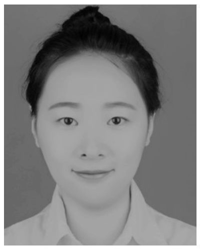
[33] L. Tianze, W. Muqing, Z. Min, and L. Wenxing, "An overhead-optimizing task scheduling strategy for ad-hoc based mobile edge computing," *IEEE Access*, vol. 5, pp. 5609–5622, 2017.

[34] J. Zheng, Y. Cai, W. Yuan, and X. S. Shen, "Stochastic computation offloading game for mobile cloud computing," in *Proc. IEEE/CIC Int. Conf. Commun. China*, Jul. 2016, pp. 1–6.

GUANGSHUN LI received the Ph.D. degree from Harbin Engineering University, China, in 2008. He is currently an Associate Professor with the School of Information Science and Engineering, Qufu Normal University, China. He is a Visiting Scholar with The Hong Kong Polytechnic University, in the second half year of 2019. He has already published more than 50 articles. His research interests include wireless networks, the IoT, and Big data.

QINGYAN LIN received the bachelor's degree in computer science and technology from Qufu Normal University, China, in 2017, where she is currently pursuing the M.Sc. degree. Her current research interests include mobile edge computing, task scheduling, and resource allocation.

JUNHUA WU received the Ph.D. degree from Harbin Engineering University, China, in 2009. She is currently an Associate Professor with the School of Information Science and Engineering, Qufu Normal University, China. She has already published more than 40 articles. Her research interests include wireless networks, edge computing, and the IoT.



YING ZHANG received the bachelor's degree in computer science and technology from Qufu Normal University, China, in 2017, where she is currently pursuing the M.Sc. degree. Her current research interests include the Internet of Vehicles, edge computing, Stackelberg game, wireless resource allocation, heterogeneous computing systems, game theory, and mobile computing.

JIAHE YAN received the bachelor's degree in computer science and technology from Qufu Normal University, China, in 2017, where she is currently pursuing the M.Sc. degree. Her current research interests include mobile edge computing, task scheduling, and the IoT.