2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS) | 978-1-6654-4935-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/MASS52906.2021.00037

Distributed Game-Theoretical Task Offloading for Mobile Edge Computing

En Wang[1], Pengmin Dong[1], Yuanbo Xu*[1], Dawei Li[2], Liang Wang[3], and Yongjian Yang[1]

[1]Jilin University, China
[2]Montclair State University, USA
[3]Northwestern Polytechnical University, China

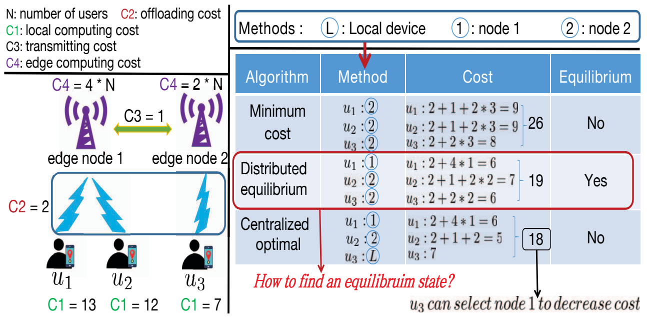
***Abstract*—Mobile Edge Computing (MEC) has been envisioned as a promising distributed computing paradigm, where mobile users offload their tasks to edge nodes to decrease the cost of energy and computation. However, most existing works only consider the congestion of wireless channels as the crucial factor influencing the strategy-making process, and ignore the impact of the offloading among edge nodes. In addition, centralized task offloading strategies result in heavy computation complexity in center nodes. Along this line, we take both the congestion of wireless channels and the offloading among multiple edge nodes into consideration to enrich users' offloading strategies. To this end, we first formulate the offloading problem as a multi-user potential game, and then propose a distributed task offloading algorithm to reach an equilibrium state which can also protect individual privacy. Specifically, in the above task offloading algorithm, we propose two subalgorithms to select users for updating strategies: Parallel User Selection Algorithm (PUS) and Single User Selection Algorithm (SUS) in order to substantially accelerate the convergence. Extensive experiments on three real-world data sets validate that the proposed algorithm achieves a Nash equilibrium and effectively decreases the total user cost which is acceptable compared to the optimal solution.**

***Index Terms*—Mobile Edge Computing, computation offloading, potential game, Nash equilibrium.**

I. Introduction

With the development of 5G [1] and other networking technologies, terminal devices have been involved with computation-intensive and latency-critical applications, including face recognition, natural language processing, interactive gaming [2]–[5] and so on. Due to hardware limitation, mobile devices are generally limited in battery life and computing resources. Mobile Edge Computing (MEC) is envisioned as a promising distributed computing paradigm to decrease the computing cost as well as enhance Quality of Service (QoS) by utilizing edge nodes' computing capacity [6]–[8].

In MEC, because the computing capacity and wireless channels of edge nodes are shared by the offloading users, the mobile users need to decide whether offloading tasks to edge nodes for computing or processing tasks utilizing their local devices, which raises the fundamental task offloading problem. Recently, some works only consider the impact of wireless channel's competition regardless of the offloading among edge nodes, which may change the offloading decision [9]. Moreover, the proposed offloading strategies are mostly



Algorithm	Method	Cost		Equilibrium
Minimum cost	u_1 : ② u_2 : ② u_3 : ②	$u_1 : 2+1+2*3=9$ $u_2 : 2+1+2*3=9$ $u_3 : 2+2*3=8$	26	No
Distributed equilibrium	u_1 : ① u_2 : ② u_3 : ②	$u_1 : 2+4*1=6$ $u_2 : 2+1+2*2=7$ $u_3 : 2+2*2=6$	19	Yes
Centralized optimal	u_1 : ① u_2 : ② u_3 : Ⓛ	$u_1 : 2+4*1=6$ $u_2 : 2+1+2=5$ $u_3 : 7$	$\boxed{18}$	No

Fig. 1: Problem description for task offloading game.

centralized [10]–[12], i.e., the edge nodes collect the users' information and make the global task offloading decisions which may result in both heavy computation complexity for the edge nodes and privacy leakage for users. Furthermore, the overall offloading decisions may be incapable of satisfying all the users when users have better options and alter the offloading strategy unilaterally.

Therefore, in this paper, we take both the offloading among edge nodes and the competition in wireless channels into consideration. Each mobile user has a home edge node, which is the nearest edge node to the user, and multiple neighbor edge nodes, which are connected to the user's home edge node via optical fiber cables. As illustrated in Fig. 1 (left part), user u_1 regards the edge node 1 as his/her home node and edge node 2 as its neighbor edge node. Due to different locations and the transmission distance of optical fiber, each edge node has some neighbor connecting nodes. Obviously, for different mobile users, the division of edge nodes may also be different. The wireless channels shown as blue arrows in Fig. 1 are shared by the offloading users. There are three methods for task computing. Apart from processing tasks utilizing their local devices, users have two more options: offloading to the home edge node, and offloading to the neighbor edge node via fibre-optical wired transmission from the home edge node. When they choose to offload the tasks to neighbor edge nodes, they must first offload tasks to their home edge nodes, and then their home edge nodes will offload tasks to corresponding neighbor edge nodes.

Moreover, we consider a distributed algorithm to let users select offloading methods instead of uploading users' infor-

*The corresponding author is Yuanbo Xu.

mation to the center edge node which can protect individual privacy. Some comparing offloading algorithms are shown in Fig. 1 (right part). An intuitive idea (Minimum cost) for each user is to assume that there is only him/her in this model and to select the method with the minimum cost. However, this leads to the highest total cost of 26. An ideal approach (Centralized optimal) achieves the least total cost of 18. Nevertheless, it does not reach an equilibrium state because u_3 can select node 2 to achieve a less cost. A trade-off (Distributed equilibrium) is our target method, where a relatively acceptable cost of 19 and an equilibrium state are achieved, and no user has the motivation to change the decision unilaterally. Hence, in this paper, *how to construct a distributed model to achieve the equilibrium while guaranteeing a good total cost* is the first challenge. Moreover, during the task offloading process, each user has individual conditions (e.g., battery condition, and the time urgency of the task). Therefore, the second challenge is *how to design a unified distributed algorithm satisfying all mobile users' personal conditions*. Finally, even if we find the distributed equilibrium state, the total cost performance is not always optimal. Hence, the third challenge is *how to guarantee a good total cost that is upper-bounded with respect to the centralized optimal solution.*

To deal with the above challenges, we first formulate the task offloading problem as a multi-user task offloading game, where each user selects a task offloading method to minimize his/her cost separately. Then, we prove that the formulated game is a potential game by constructing a global potential function. The change of each user's cost can be uniformly mapped into the change of the global potential function. Through continuing to approach the minimum value of the global potential function, we achieve an equilibrium state where each user's cost function achieves a local minimum value. Furthermore, we design a distributed game-theoretical task offloading algorithm to achieve the Nash equilibrium. For the cost function, the weighting parameters can be modified by the users based on their individual conditions. Finally, we utilize the metric of Price of Anarchy (PoA) to guarantee the upper bound of the total cost with respect to that of the centralized optimal solution.

In summary, the contributions are listed as follows:

- We first prove that it is NP-hard to find the centralized optimal solution of the task offloading problem in MEC. Instead, we formulate the distributed task offloading problem as a multi-user task offloading game, which takes both wireless channels and the offloading among edge nodes into consideration.
- We prove that the formulated multi-user game is a potential game. Furthermore, we design a distributed task offloading algorithm to reach an equilibrium state; at the same time, users could modify the parameters of the cost function to satisfy their individual conditions.
- We show that the proposed distributed task offloading algorithm can converge to a Nash equilibrium within a finite number of update steps. Moreover, we prove the upper bound of the number of update steps and the lower bound of the total costs compared to the optimal centralized solution.
- We conduct extensive simulations based on three real widely-used data sets of edge networks. The results verify that our proposed algorithm can achieve a Nash equilibrium, while achieving a total user cost close to that of the optimal solution.

The remainder of the paper is organized as follows. After reviewing the related works in Section II, we introduce the system model, the NP-hardness of the centralized problem and the potential game formulation in Section III. Then, we propose the distributed task offloading algorithm and analyze its performance theoretically in Section IV. Finally, we conduct extensive simulations to evaluate the proposed algorithm in Section V and conclude the paper in Section VI.

II. Related Works

A. Task Offloading

The research on task offloading in MEC can be classified into two categories: centralized task offloading and distributed task offloading. For the former [13], [14], Baron *et al.* [13] propose a multi-user task offloading approach among multi-edge, which can achieve the maximum task completing ratio. Jiang *et al.* [14] propose a centralized task offloading approach based on deep learning and MEC to minimize the total energy consumption. The centralized strategies always fail to consider whether a user is satisfied with the offloading decisions. For the latter [15], [16], Hong *et al.* [15] propose multi-hop cooperative computation offloading for industrial IoT–Edge–Cloud computing environments. Wang *et al.* [16] investigate the offloading problem and resource allocation utilizing deep reinforcement learning. However, most existing works fail to consider the impact of offloading among edge nodes or the congestion of wireless channels, which is actually a more realistic scenario.

B. Potential Game

Many studies have recently utilized the potential game theory to make distributed game-theoretical decisions and achieve the Nash equilibrium. Fabiani *et al.* [17] formulate the multi-vehicle driving coordination problem as a mixed-integer potential game and find an equilibrium solution. Liu *et al.* [18] formulate the multi-user computation offloading problem as a potential game in which the mobile devices make the offloading decisions in a distributed manner. Raschell *et al.* [19] propose a novel access point selection approach based on a potential game relying on software-defined networking. He *et al.* [20] investigate a app users computation offloading problem in mobile edge computing. Furthermore, they propose a potential game-theoretical approach to achieve efficient computation offloading. Wu *et al.* [21] formulate the edge user allocation problem as a potential game and propose a decentralized algorithm to serve the maximum number of users with minimum overall system cost. However, most works design a fixed and unified cost function for all the users. Hence, the corresponding potential game does not consider the diverse requirements of the users. In this paper, we propose a distributed game-theoretical approach, where mobile users can achieve different preferences by adjusting the parameters of the cost function.

III. MULTI-USER TASK OFFLOADING GAME

A. System Model

We first introduce the system model for the task offloading in MEC. In our system, there are N users denoted as $\mathcal{U} = \{u_1, u_2, ..., u_N\}$ and M edge nodes denoted as $\mathcal{K} = \{b_1, b_2, ..., b_M\}$ embedded in base stations. Each edge node is interconnected via optical fibers with not all but several nodes. Our model investigates the offloading problem in quasi-static scenarios where the users' locations and edge nodes are stable. Each user has his/her own nearest edge node called the home edge node. Correspondingly, other edge nodes connected with the home edge node are regarded as the user's neighbor edge nodes. For different users, their home edge nodes and neighbor edge nodes may be different. We assume that users' tasks are generated by their daily mobile devices such as mobile phones or pads and so on. And there are three ways of computing their tasks: 1) utilizing their own devices, 2) utilizing their home edge nodes through wireless channels, and 3) utilizing their neighbor edge nodes, where the users' tasks must be first offloaded to the home edge nodes and then sent to the neighbor edge nodes.

B. Local Device Computing Model

Let $C_l = \{c_{u_1}, c_{u_2}, ..., c_{u_N}\}$ denote the computing capacity of all users' local devices, where $c_{u_i}(i = 1, ..., N)$ means the computing capacity of user u_i's local device. If user u_i determines to compute tasks in the local device, the computing time will be formulated as follows:

$$t_{u_i} = \frac{R_i}{c_{u_i}}, \tag{1}$$

where R_i denotes the CPU cycles required to complete tasks. Furthermore, we consider energy consumption. Let P_i denote the energy consumption per CPU cycle, then the energy consumption will be given as:

$$e_{u_i} = P_i \cdot R_i. \tag{2}$$

Besides, in order to adjust users' individual conditions, we associate two positive parameters, α and β $(0 <= \alpha, \beta <= 1)$, with the time cost and energy cost, respectively, when calculating the total cost for a user. If there is little energy left in the user's battery, user can increase the value of the β to increase the cost of energy consumption. Equally, if their time is valuable, they can increase α to emphasize the impact of time costs. In brief, users can adjust the value of the two weights to adapt to the ever-changing situation. As described above, for user u_i, according to Eq. (1) and Eq. (2) the total cost of local device computing is formulated as

$$T_{l(i)} = \alpha_i t_{u_i} + \beta_i e_{u_i}. \tag{3}$$

C. Home Edge Node Computing Model

In this section, we consider the scenario where users choose their home edge nodes (i.e., the closest edge node) for completing tasks. The computing time consists of two parts: task computing time and task offloading time. Let $C_e = \{c_{b_1}, c_{b_2}, ..., c_{b_M}\}$ denote the computing capacities of all edge nodes and $c_{h(u_i)}(u_i \in \mathcal{U})$ denotes the computing capacity of user i's home edge node. In addition, $N_{h(u_i)}$ denotes the total number of users of user i's home node. Then, the task computing time is given by:

$$t_{h(u_i)} = \frac{R_i}{(\frac{c_{h(u_i)}}{N_{h(u_i)}})} = \frac{R_i N_{h(u_i)}}{c_{h(u_i)}}. \tag{4}$$

Meanwhile, the task offloading time is as follows:

$$t_{u_i}^{off} = \frac{k_i}{r_i} + Ak_iU(i), \tag{5}$$

where k_i denotes user u_i's offloading data size, r_i denotes the transmission rate, $U(i)$ denotes the number of users that share user u_i selected channel, and A is a congestion parameter. As mentioned earlier, we also need to take energy consumption into consideration. Let E_i denote user u_i's data transmission power. The energy consumption is described as:

$$e_{u_i}^{off} = E_i \cdot k_i. \tag{6}$$

And for user u_i, the total cost of home edge node computation is derived as:

$$T_{h(i)} = \alpha_i(t_{h(u_i)} + t_{u_i}^{off}) + \beta_i E_i t_{u_i}^{off}. \tag{7}$$

According to the method of Eq. (3), here the user u_i's parameters, α_i and β_i (both are positive numbers) denote the weights of computing time and energy consumption.

D. Neighbor Edge Node Computing Model

In the case of neighbor edge node computing, the operation mechanism is that first users offload their tasks to the home edge node through wireless channels, then the home edge node will transmit the tasks to the neighbor edge node specified by users in advance. Compared with the home edge node computing model, the neighbor edge node computing model adds an extra variable, which is the transmission time between home edge node a and neighbor edge node b denoted as B_{ab}. For user u_i, the task computing time is as follows:

$$t_{n(u_i)} = \frac{R_i}{(\frac{c_{n(u_i)}}{N_{n(u_i)}})} = \frac{R_i N_{n(u_i)}}{c_{n(u_i)}}, \tag{8}$$

where $c_{n(u_i)}$ denotes neighbor edge node's computing capacity selected by user u_i and $N_{n(u_i)}$ denotes the neighbor edge node's total number of users selected by user u_i. We might as well assume that user u_i chooses offloading tasks to neighbor edge node b through home edge node a, then the task offloading time is derived as:

$$t_{u_i}^{nei\_off} = \frac{k_i}{r_i} + Ak_iU(i) + B_{ab}. \tag{9}$$

Let E_i denote user u_i's data transmission power. The energy consumption for offloading task is derived as:

$$e_{u_i}^{nei\_off} = E_i \cdot k_i. \tag{10}$$

As mentioned above, we set positive parameters, α_i and β_i to adjust different individual performance, the total cost for performing the tasks is described as:

$$T_{n(i)} = \alpha_i(t_{n(u_i)} + t_{u_i}^{nei\_off}) + \beta_i e_{u_i}^{nei\_off}. \tag{11}$$

E. NP-hardness of the Centralized Problem

First, we consider the centralized optimization problem of minimizing the total costs of all users. Mathematically, given all users' strategies $\mathbf{s} = (s_i, s_{-i})$ (i.e., s_i denotes the user u_i's strategy and s_{-i} denotes others' strategies), the problem can be formulated as follows:

$$\min_{\mathbf{s}} \sum_{u_i \in \mathcal{U}} T_i(\mathbf{s}),$$
$$\text{subject to } s_i \in S_i, \forall u_i \in \mathcal{U}, \tag{12}$$

where S_i means the optional strategies of user u_i and T_i is user u_i's total cost. Then, we try to prove that finding the optimal solution of the formulated centralized optimization problem is quite difficult, as shown in Theorem 1.

Theorem 1. *The problem of finding the optimal solution to minimize the total cost in a centralized manner is NP-hard.*

Proof. The main idea is to change the perspective of the problem in order to tally with the maximization version of Generalized Assignment Problem (GAP) [22] which is NP-hard. The problem of GAP is defined as follows:

Input: there are n items and m knapsacks, where every item has a different profit and size when assigned to different knapsacks and each knapsack has its own capacity. For example, assigning item i to knapsack j, its size and profit will be $s_{i,j}$ and $p_{i,j}$, separately.

Output: The assignment of items to knapsacks which will reach the optimal total profits without exceeding the capacity limit of the knapsacks.

In our problem, the worst situation is processing tasks at local devices. Thus, we regard the cost saving utilizing edge nodes compared with local computation as the task's profit. In that way, the profit of item i is defined as $c_i - e_i$, where c_i denotes the cost of local device computation and e_i denotes the cost of edge node computation. When the user number of wireless channel is large enough that $e_i = c_i$, the user number at this time is the task's striction of item i. And the size of task i is regarded as the size of item, the capacity of each channel is tasks' strictions. Now that the maximization version of GAP is NP-hard, our problem is also NP-hard. □

F. Potential Game Formulation

In this section, we try to utilize distributed task offloading method and formulate our model as a potential game [23], [24]. First, we will introduce some definitions.

Definition 1. (Nash equilibrium) *A strategy profile* $\hat{\mathbf{s}} = \{\hat{s}_1, \hat{s}_2, ..., \hat{s}_N\}$ *is a Nash equilibrium for our multi-user task offloading game if and only if*

$$T_i(\hat{s}_i, \hat{s}_{-i}) = min\ T_i(s_i, \hat{s}_{-i})\ \forall u_i \in \mathcal{U}, \forall s_i \in S_i. \tag{13}$$

It is obvious that no user has motivation to decrease the task completing costs by altering the strategy unilaterally in the state of Nash equilibrium.

Definition 2. (Weighted potential game) *A game is a weighted potential game if and only if there exists a potential function* $\delta(\boldsymbol{s})$ *for* $\forall i \in \mathcal{U}$ *satisfying:*

$$T_i(s_i, s\_i) - T_i(s_i', s_{-i}) = \mu_i(\delta(s_i, s_{-i}) - \delta(s_i', s_{-i}))$$
$$\forall s_i, \forall s_i' \in S_i,\ \forall s_{-i} \in S_{-i}, \tag{14}$$

where μ_i $(i = 1, ..., N)$ constitutes a vector of positive numbers.

Now we introduce the two significant properties of a potential game: (1) *the existence of Nash equilibrium* : there is always at least one Nash equilibrium in the potential game, (2) *Finite improvement property:* the potential game always converges to a Nash equilibrium in a finite number of decision steps which can decrease their costs, irrespective of the initial strategy profile or the users' updating order.

Next, in Theorem 2, we will prove that our multi-user task offloading game is a weighted potential game.

Theorem 2. *The multi-user task offloading game is a weighted potential game and has at least one Nash equilibrium and the finite improvement property.*

Proof. We first construct the potential function as follows:

$$\delta(\mathbf{s}) = \sum_{n \in \gamma} \sum_{j=0}^{C(n)} Aj + \sum_{b \in \mathcal{K}} \sum_{j=0}^{|N_b|} q \frac{j}{c_b} + \sum_{i \in \mathcal{U}} W_i I(a_i, 0) +$$
$$\sum_{i \in \mathcal{U}} V_i I(a_i, 1) + \sum_{i \in \mathcal{U}} Z_i I(a_i, 2), \tag{15}$$

where γ is the wireless channel set, $q = R_i/k_i$ denotes the CPU cycles processing per data size, $W_i = q(\frac{1}{c_{u_i}} + \frac{\beta_i P_i}{\alpha_i})$, $V_i = \frac{\beta_i E_i}{\alpha_i} + \frac{1}{r_i}$, $Z_i = \frac{\beta_i E_i}{\alpha_i} + \frac{1}{r_i} + \frac{B_{cd}}{k_i}$, and $I(a_i, m)$ is an indicator function defined as:

$$I(a_i, m) = \begin{cases} 0, & \text{if } a_i \neq m \\ 1, & \text{if } a_i = m \end{cases} \tag{16}$$

Here, $m = 0$ means that users process tasks locally, $m = 1$ means that users choose offloading to home edge nodes, and $m = 2$ means that users choose offloading to neighbor edge nodes. Moreover, a_i denotes user u_i's strategy assigned from the set $\{0, 1, 2\}$.

We define the original strategy profile of user u_i and other users as $\mathbf{s} = (s_i, s_{-i})$. When user u_i changes the strategy into s_i' while others keep stable, the strategy profile becomes $\mathbf{s} = (s_i', s_{-i})$. Considering all the user u_i's strategy changing situations, we will discuss the following five cases: 1) From local device to home edge node; 2) From local device to neighbor edge node; 3) From home edge node to neighbor edge node; 4) From channel a to channel b of the home edge node; 5) From neighbor edge node a to neighbor edge node b. Obviously, reversing the order of each case does not affect the result. For case 1):

Algorithm 1 Distributed Game-Theoretical Task Offloading Algorithm for user $i \in \mathcal{U}$

```
1: Input $\alpha_i$, $\beta_i$ according to the battery condition and the time urgency of the task.
2: Initialize $s_i(0) = r$ by selecting a method $r \in S_i$.
3: Report $s_i(0)$ $(u_i \in \mathcal{U})$ to their home nodes.
4: Receive the condition of users in each home edge node's channels and the total user number of each neighbor edge node.
5: Calculate the three costs $T_{l(i)}$, $T_{h(i)}$, $T_{n(i)}$ performing the task in local device, home edge node and neighbor edge node.
6: repeat for each decision slot $t$
7:     Obtain the condition of users in home edge node's each channel and the total user number of each neighbor edge node.
8:     Compute the better option $\triangle_i(t)$.
9:     if $\triangle_i(t) \neq \varnothing$ and $\triangle_i(t) \neq s_i(t-1)$ then
10:        Send the request to the edge node for competing the opportunity of updating decision.
11:        if Win the opportunity then
12:            Update the offloading decision $s_i(t) = \triangle_i(t)$.
13:            Report $s_i(t)$ to the edge node.
14:        else
15:            Maintain the last time slot decision $s_i(t) = s_i(t-1)$.
16: until The termination message is received.
```

$$T_i(\acute{s}_i) - T_i(s_i) = (\alpha_i(\frac{k_i}{r_i} + \frac{R_i N_{h(u_i)}}{c_{h(u_i)}} + Ak_iU(i)) + \beta_i E_i k_i) - (\alpha_i \frac{R_i}{c_{u_i}} + \beta_i P_i R_i)$$
$$= \alpha_i k_i (\frac{1}{r_i} + q\frac{N_{h(u_i)}}{c_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + AU(i) - q(\frac{1}{c_{u_i}} + \frac{\beta_i P_i}{\alpha_i}))$$
$$= \alpha_i k_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)). \quad (17)$$

For case 2), We might as well suppose user u_i's home edge node is c and his/her offloading neighbor edge node is d. Then we have:

$$T_i(\acute{s}_i) - T_i(s_i) = (\alpha_i(\frac{k_i}{r_i} + \frac{R_i N_{n(u_i)}}{c_{n(u_i)}} + B_{cd} + Ak_iU(i)) + \beta_i E_i k_i) - (\alpha_i \frac{R_i}{c_{u_i}} + \beta_i P_i R_i))$$
$$= \alpha_i k_i (\frac{1}{r_i} + q\frac{N_{n(u_i)}}{c_{n(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + \frac{B_{cd}}{k_i} + AU(i) - q(\frac{1}{c_{u_i}} + \frac{\beta_i P_i}{\alpha_i}))$$
$$= \alpha_i k_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)). \quad (18)$$

For case 3), as mentioned above, we also suppose that edge node c and d are user u_i's home node and selected neighbor edge node.

Algorithm 2 Information Update Algorithm for the Edge Nodes.

```
1: Receive $s_i(0)$ $(u_i \in \mathcal{U})$ from each user.
2: For each edge node, send its user conditions of channels to home users and its users number to corresponding neighbor edge nodes.
3: repeat for each decision slot t.
4:     Receive the updating request from the users and let $\hat{u}$ denote the set of users who send the request.
5:     if $\hat{u} \neq \varnothing$ then
6:         Select a user randomly from $\hat{u}$.
7:         Inform the user to update the decision
8:         Receive $s_i(t)$ from user $u_i \in \mathcal{U}$ and update the user condition of each edge node's channel.
9: until No request is received from the user.
10: Send the termination message to all users.
```

$$T_i(\acute{s}_i) - T_i(s_i) = (\alpha_i(\frac{k_i}{r_i} + \frac{R_i N_{n(u_i)}}{c_{n(u_i)}} + B_{cd} + Ak_iU(i)) + \beta_i E_i k_i) - (\alpha_i(\frac{k_i}{r_i} + \frac{R_i N_{h(u_i)}}{c_{h(u_i)}} + Ak_iU(i)) + \beta_i E_i k_i)$$
$$= \alpha_i k_i (\frac{1}{r_i} + q\frac{N_{n(u_i)}}{c_{n(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + \frac{B_{cd}}{k_i} - (\frac{1}{r_i} + q\frac{N_{h(u_i)}}{c_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i}))$$
$$= \alpha_i k_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)). \quad (19)$$

For case 4), since the number of mobile users among different wireless channels is different, the user number are different. We use $C(i_a)$ and $C(i_b)$ to denote the user number before and after strategy changing.

$$T_i(\acute{s}_i) - T_i(s_i) = (\alpha_i(\frac{k_i}{r_i} + \frac{R_i N_{h(u_i)}}{c_{h(u_i)}} + Ak_iU(i_b)) + \beta_i E_i k_i) - (\alpha_i(\frac{k_i}{r_i} + \frac{R_i N_{h(u_i)}}{c_{h(u_i)}} + Ak_iU(i_a)) + \beta_i E_i k_i)$$
$$= \alpha_i k_i (\frac{1}{r_i} + q\frac{N_{h(u_i)}}{c_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + AU(i_b) - (\frac{1}{r_i} + q\frac{N_{h(u_i)}}{c_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + AU(i_a)))$$
$$= \alpha_i k_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)). \quad (20)$$

For case 5), by the similar argument in case 4), it is easily to prove that in situation of neighobr edge nodes, the formula also holds. □

Finally, Theorem 2 is proved.

IV. ALGORITHM DESIGN

In this section, we introduce our algorithms. Algorithm 1 is the distributed game-theoretical task offloading algorithm for the mobile users; Algorithm 2 is the information update algorithm for the edge nodes. When the algorithm terminates, a Nash equilibrium will be reached.

A. *Distributed Task Offloading Algorithm*

Theorem 2 guarantees that the multi-user task offloading game will converge to a Nash equilibrium within a finite number of decision slots. The main idea of the algorithm is utilizing the finite improvement property and selecting a set of mobile users to decrease their costs by updating the task offloading decisions in each decision slot.

In the initialization phase (line 1) of Algorithm 1, mobile users first input the weighted parameters of battery condition and the time urgency of the tasks and a random offloading strategy. In the calculation phase (lines 6-16), each user receives the number of users of each edge node and channel. In this way, every user could calculate the better response strategy (line 7-8); then, the user sends a request to the home edge node applying for updating the decision. If the user is selected, he/she updates the decision in the next slot. Other users will keep the decision in the previous decision slot (lines 9-15). The process repeats until users receive the termination message (line 16) (i.e., no users send the updating request to nodes). Then the algorithm converges to a Nash equilibrium.

B. Information Update Algorithm

The information update algorithm updates the number of users in each channel and edge node. In addition, this algorithm selects users to update the decision in next decision slot. In the initialization phase, after receiving the initial decisions from all users (line 3), the algorithm updates the number of users in each channel and edge node and sends it to users. Next, when receiving users' updating requests, the algorithm will choose one user to update his/her strategy (lines 4-8). The algorithm terminates until no request is received, then it sends the termination messages to all users (lines 9-10).

Furthermore, we introduce the following two user selection algorithms, Parallel User Selection Algorithm (PUS) and Single User Selection Algorithm (SUS). SUS randomly selects only one user from the set of users that send the updating requests and allows the user to update the decision in next decision slot. To decrease the convergence time, we further propose PUS , which is inspired by the idea that some users whose strategies cover no overlapping channels and edge nodes could simultaneously update offloading strategy in the same decision slot. The detailed description is as follows. As shown in Algorithm 3, the inputs are $\mathcal{U}'$, and $\mathcal{K}$. Specifically, $\mathcal{U}'$ is the set of users sending the updating requests. In the initialization phase (line 1), we set the selected updating users set as empty. Then we traverse each edge node to find if there exists users whose updating strategy consists this node (lines 2-6). If user's strategy is local device and his/her home node is the traversing node, then the user is added to set l' (line 5). Meanwhile, if the updating strategy is the traversing edge node no matter whether it is his/her home edge node or not, she/he will be added to set e'(line 6). We will randomly select two users belonging to set l' and e' respectively and add them to set μ (lines 7-8), deleting the chosen users from the set $\mathcal{U}'$ by the way. Finally, when all the edge nodes are traversed, the algorithm will return the selected users set μ.

Algorithm 3 Parallel User Selection Algorithm.

```
Input: $\mathcal{U}'$, $\mathcal{K}$.
1: Initialize $\mu = \varnothing$.
2: for all $j \in \mathcal{K}$ do
3:     $l' = \varnothing$ $e' = \varnothing$
4:     for all $i \in \mathcal{U}'$ do
5:         if $s_i(t) = Local$ and $i \in j$ then $l' \leftarrow l' \cup i$.
6:         else if $s_i(t) = j$ then $e' \leftarrow e' \cup i$.
7:     Randomly select a user $m \in l'$ and $\mu \leftarrow \mu \cup m$
8:     Randomly select a user $n \in e'$ and $\mu \leftarrow \mu \cup n$
9:     $\mathcal{U}' \leftarrow \mathcal{U}' - m - n$
   return $\mu$.
```

C. Convergence Analysis

According to Theorem 2, the proposed distributed task offloading algorithm will converge to a Nash equilibrium within a finite number of update iterations. We then analyze the upper bound of the number of iterations for convergence.

Theorem 3. *The number of decision slots D for convergence of the distributed task offloading algorithm satisfies the following equation.*

$$D < \frac{|\mathcal{U}|\alpha_{max}k_{max}}{\triangle P_{min}}(q(\frac{1}{C_l^{min}} + \frac{\beta_{max}P_{max}}{\alpha_{min}}) - \frac{Aj}{2}(\frac{|\mathcal{U}|}{|\gamma|}+1) - \frac{q}{2C_e^{max}}(\frac{|\mathcal{U}|}{|\mathcal{K}|}+1) - (\frac{\beta_{min}E_{min}+\alpha_{max}}{\alpha_{max}r_{max}})). \tag{21}$$

Proof. We consider the situation where only a user $u_i \in \mathcal{U}$ changes strategy from s_i to $\acute{s}_i$. When users change strategy, the value of potential function will decrease correspondingly. When an equilibrium state is reached, the best case is that the capacity of edge node is large enough for all the users to offload their tasks to it. Then according to Eq. (14), we have the following equation:

$$\delta(\mathbf{s}) \geq Aj\frac{|\mathcal{U}|}{2}(\frac{|\mathcal{U}|}{|\gamma|}+1) + \frac{q|\mathcal{U}|}{2C_e^{max}}(\frac{|\mathcal{U}|}{|\mathcal{K}|}+1) + |\mathcal{U}|(\frac{\beta_{min}E_{min}+\alpha_{max}}{\alpha_{max}r_{max}}), \tag{22}$$

where C_e^{max} means the maximum computing capacity among all edge nodes.

On the contrary, the worst case is that mobile users can only choose processing locally. So we have:

$$\delta(\mathbf{s}) \leq q|\mathcal{U}|(\frac{1}{C_l^{min}} + \frac{\beta_{max}P_{max}}{\alpha_{min}}), \tag{23}$$

where C_l^{min} denotes the minimum computing capacity of all local devices.

According to Eq. (22) and Eq. (23), when user u_i changes strategy from s_i to $\acute{s}_i$, we have:

$$\delta(\mathbf{s}) - \delta(\acute{\mathbf{s}}) < q|\mathcal{U}|(\frac{1}{C_l^{min}} + \frac{\beta_{max}P_{max}}{\alpha_{min}}) - Aj\frac{|\mathcal{U}|}{2}(\frac{|\mathcal{U}|}{|\gamma|}+1) - \frac{q|\mathcal{U}|}{2C_e^{max}}(\frac{|\mathcal{U}|}{|\mathcal{K}|}+1) - |\mathcal{U}|(\frac{\beta_{min}E_{min}+\alpha_{max}}{\alpha_{max}}) \tag{24}$$

Then, we have the following equation:

$$D < \frac{|\mathcal{U}|\alpha_{max}k_{max}}{\triangle P_{min}}(q(\frac{1}{C_l^{min}} + \frac{\beta_{max}P_{max}}{\alpha_{min}}) - \frac{Aj}{2}(\frac{|\mathcal{U}|}{|\gamma|}+1) - \frac{q}{2C_e^{max}}(\frac{|\mathcal{U}|}{|\mathcal{K}|}+1) - (\frac{\beta_{min}E_{min}+\alpha_{max}}{\alpha_{max}})). \tag{25}$$

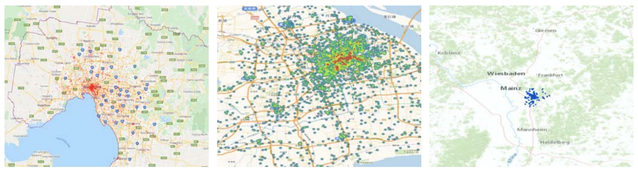

(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 2: The presentation on real world data sets.

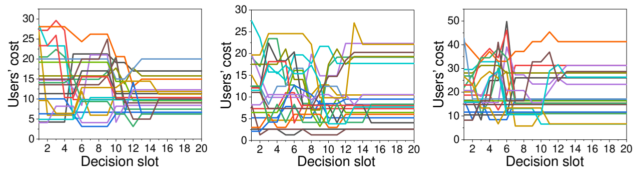

(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 3: User cost vs. decision slot.

Hence, Theorem 3 is proved. □

D. Theoretical Analysis

We then analyze the performance of the proposed distributed task offloading algorithm by analyzing Price of Anarchy (PoA). PoA is a metric measured by the ratio of the total cost of all users in the worst case of Nash equilibrium to the minimum total cost of the optimal strategy. Let S' be the set of strategy profile that can achieve Nash equilibrium of the multi-user task offloading game and $\mathbf{s}^*$ denote the centralized optimal strategy. PoA is defined as follows:

$$PoA = \max_{\mathbf{s}\in S'} \sum\nolimits_{u_i\in\mathcal{U}} T_i(\mathbf{s}) / \sum\nolimits_{u_i\in\mathcal{U}} T_i(\mathbf{s}^*). \tag{26}$$

Theorem 4. *For the multi-user task offloading game, the PoA of the metric of the overall costs satisfies that*

$$\frac{t_l^{max} + e_l^{max}}{t_e^{min} + t_{off}^{min}(1+E^{min})} \geq PoA \geq 1, \tag{27}$$

where t_l^{max} and e_l^{max} mean the maximum time and energy cost among all users' local devices, and t_e^{min} and t_{off}^{min} denote the minimum computing time and the minimum task offloading time among all edge nodes, respectively.

Proof. In our multi-edge conditions, for any user, the worst strategy is computing task at their local devices. Therefore, the total cost of our model when reaches a Nash equilibrium is always less than the total cost when all the mobile users choose to compute locally, which will be derived as:

$$\sum\nolimits_{u_i\in\mathcal{U}} T_i(\mathbf{s}) = |\mathcal{U}|(t_l^{max} + e_l^{max}). \tag{28}$$

On the other hand, if all the home nodes can afford the time and energy cost of native users, the total cost will be minimum which means:

$$\sum\nolimits_{u_i\in\mathcal{U}} T_i(\mathbf{s}^*) \geq |\mathcal{U}|(t_e^{min} + t_{off}^{min}(1+E^{min})). \tag{29}$$

In conclusion, according to the above description, the following equation holds:

$$\frac{t_l^{max} + e_l^{max}}{t_e^{min} + t_{off}^{min}(1+E^{min})} \geq PoA \geq 1. \tag{30}$$

Hence, Theorem 4 is proved. □

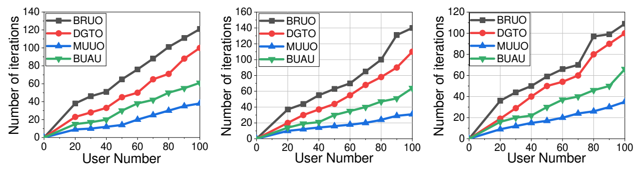

(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 4: Decision slot vs. user number.

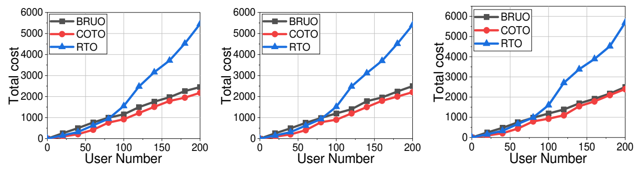

(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 5: Total cost vs. user number.

V. PERFORMANCE EVALUATION

A. Data Sets & Settings

The following three real-world data sets are used for the evaluation:

- *Melbourne* [25], [26] contains all cellular base stations GPS data and all unique user locations in Australia. Fig. 2 (a) shows the distribution of edge nodes and users.
- *Shanghai* [27], [28] contains more than 7.2 million records of accessing the Internet through 3,233 base stations from 9,481 mobile phones for six months. Fig. 2 (b) shows the distribution of base stations. Each base station denotes an edge node in Shanghai, China.
- *Darmstadt* [29] contains the GPS data of the base stations in Darmstadt, Germany whose distributions are shown as blue dots in Fig. 2 (c).

We let users' mobile device process natural language processing application whose data size k_i distributes in $[100, 200]$ KB. And the computing capacity of users' local devices and edge nodes are a Poisson distribution in the ranges of $[1, 2.5]$ GHz and $[100, 120]$ GHz, respectively. The energy consumption per CPU cycle P_i is 1. As for offloading model, we define users' transmission rate r_i as:

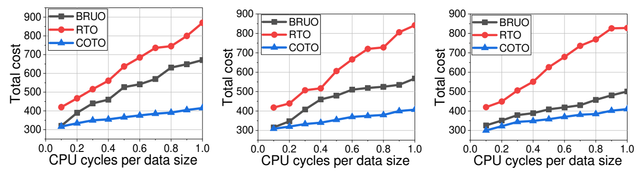
$$r_i = w \log_2\left(1 + \frac{\lambda_i P_i}{\sigma^2 D^2_{(i,j)}}\right), \tag{31}$$

where $D_{(i,j)}$ denotes the distance between user u_i and edge node j, σ means the path loss factor which is set as 2, and λ denotes the background noise. And the channel bandwith W is 15 MHz. The transmission rate between edge nodes is set 15 MB/s so that the delay of them will draw from a uniform distribution across $[0.1, 0.2]$ s.

B. Comparison Algorithms & Metrics

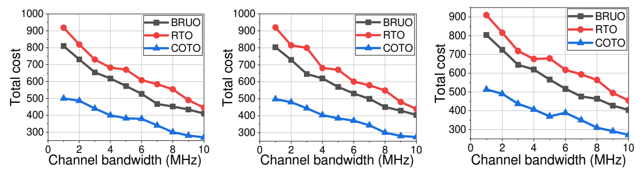
We use the following algorithms in the simulations.

- Distributed Game-theoretical Task Offloading (DGTO): The proposed algorithm that utilizes SUU algorithm to randomly select a user from the users who send the updating requests and allows the user to select better offloading method to minimize his/her cost.



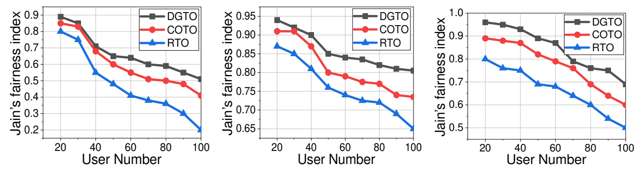
(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 6: Total cost vs. CPU cycles per data size.



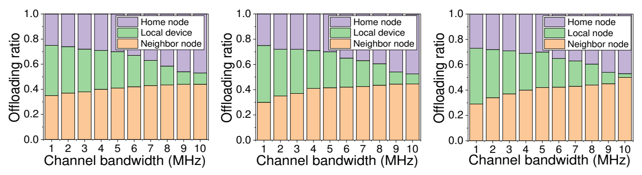
(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 7: Total cost vs. Channel bandwidth.



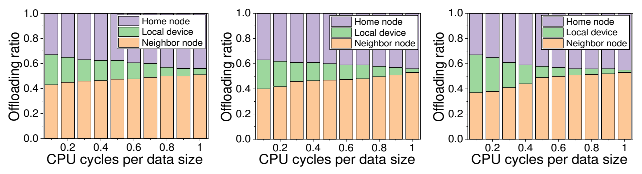
(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 8: Jain's fairness index vs. User number.



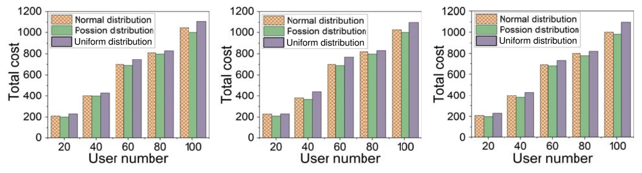
(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 9: Offloading ratio vs. Channel bandwidth.



(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 10: Offloading ratio vs. CPU cycles per data size.



(a) Melbourne (b) Shanghai (c) Darmstadt

Fig. 11: Total cost vs. Data distribution.

- Multi-User Update Offloading (MUUO): The proposed algorithm that utilizes PUS algorithm to select a set of users from the users who send the updating requests and allows the selected users to update the offloading decisions in the next decision slot.
- Better Response Update Offloading (BRUO): BRUO randomly selects a user from the users who send the update requests and allows the user to randomly select a strategy which is better than the current offloading method.
- Best Update of All Users (BUAU): BUAU inspects all users and selects the user who minimizes the value of the potential function to update the strategy in the next decision slot.
- Centralized Optimal Task Offloading (COTO): COTO is the centralized optimal approach to minimize the total cost of all users. Specifically, we use simulated annealing algorithm which is through one hundred experiments to obtain the best parameters set.
- Random Task Offloading (RTO): Each mobile user randomly selects a offloading method from the available policy set.

C. Numerical Results

1) Convergence for Nash equilibrium: We first verify the convergence for the proposed distributed algorithm, as shown in Fig. 3. Specifically we randomly select 20 users in each real data set, respectively and observe the dynamics of the costs in 20 decision slots. It is obvious that the costs of the mobile users change with the decision updates in the beginning and can converge to a stable point which reaches the state of Nash equilibrium. In Fig. 4, we investigate the number of decision slots for convergence with respect to the number of users. These algorithms rank as follows: MUUO<BUAU<DGTO<BRUO. The reason is that MUUO selects multiple users to update their decisions in parallel, while BUAU selects only one user who minimizes the potential function in each decision slot. What's more, DGTO and BRUO randomly select a user to update the decision with the best and better response update manner respectively.

2) Costs, fairness: As shown in Fig. 5, 6 and 7, we investigate the trend of total cost with respect to the number of users, CPU cycles per data size and channel bandwidth. We repeat the three simulations 500 times respectively which all rank as follows: COTO<BRUO<RTO. According to Eq. (11) and Eq. (31), we know the total cost is the positive correlation with the user number and CPU cycles per data size, and the negative correlation with the channel bandwidth. In addition, the simulations indicate that our algorithm is acceptable compared with the optimal solution.

Fig. 8 shows the dynamics of Jain's fairness index with the growth of the number of users. We repeat the simulations 500 times. Jain's fairness index [30] is used to measure the fairness of the user's costs, which is defined as $\frac{(\sum_{i\in\mathcal{U}} P_i(\mathbf{s}))^2}{|\mathcal{U}|\sum_{i\in\mathcal{U}} P_i(\mathbf{s})^2}$. It is worth noting that the fairness depends on how evenly distributed the cost of each user is. The simulation results show that the proposed DGTO achieves the highest Jain's fairness index among COTO and RTO, as DGTO can reach a Nash equilibrium of multi-user game.

3) The influence of algorithm parameters: In Fig. 9 and 10, we evaluate the influence of the channel bandwidth and CPU cycles per data size on the tasks offloading ratio. We repeat each simulation 500 times which is conducted among

60 mobile users. It is interesting to find out that the offloading ratio increases with the increase of channel bandwidth and CPU cycles per data size. When channel bandwidth gets wider, the transmission costs between edge nodes and users are lower so that more users choose offloading tasks. And when tasks are more complex, users are more likely to offload tasks to edge nodes.

In Fig. 11, we research the impact of users' data size distribution on the total cost. We take normal distribution, poisson distribution and uniform distribution into consideration. As we predicted, the simulation results show that the costs of these data distributions are in this order: uniform distribution > normal distribution > poisson distribution which is same as the rank of their users' total data size.

VI. CONCLUSION

In this paper, we investigate the multi-user task offloading problem, where we take both the offloading among edge nodes and the competition in wireless channels into consideration. We first prove that the centralized optimization problem is NP-hard and formulate the task offloading problem as a multi-user potential game, which always has a Nash equilibrium and the finite improvement property. Then, we propose a distributed task offloading algorithm to help users select the offloading method that could achieve a Nash equilibrium. Users can modify the parameters of the cost function to satisfy their individual conditions. We analyze the convergence and performance of the proposed algorithm theoretically. Finally, the simulation results based on three real data sets show that the proposed approach achieves a Nash equilibrium while achieving a total user cost close to that of the optimal solution.

REFERENCES

[1] S. Kim, E. Visotsky, P. Moorut, K. Bechta, A. Ghosh, and C. Dietrich, "Coexistence of 5g with the incumbents in the 28 and 70 ghz bands," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1254–1268, 2017.

[2] C. Ding and D. Tao, "Trunk-branch ensemble convolutional neural networks for video-based face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 1002–1014, 2018.

[3] Z. Zong and C. Hong, "On application of natural language processing in machine translation," in *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, 2018, pp. 506–510.

[4] H. Meng and D. Wang, "Robust design for game-based instruction using interactive whiteboards," in *2012 IEEE Fourth International Conference On Digital Game And Intelligent Toy Enhanced Learning*, 2012, pp. 250–253.

[5] Z. Zhang, D. Weng, H. Jiang, Y. Liu, and Y. Wang, "Inverse augmented reality: A virtual agent's perspective," in *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2018, pp. 154–157.

[6] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[7] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[8] Z. Niu, Y. Wu, J. Gong, and Z. Yang, "Cell zooming for cost-efficient green cellular networks," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 74–79, 2010.

[9] K. Kumar and Y. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[10] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, 2020.

[11] A. Zhou, S. Wang, S. Wan, and L. Qi, "Lmm: latency-aware micro-service mashup in mobile edge computing environment," *Neural Computing and Applications*, pp. 1–15, 2020.

[12] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Risk-aware application placement in mobile edge computing systems: A learning-based optimization approach," in *2020 IEEE International Conference on Edge Computing (EDGE)*, 2020, pp. 83–90.

[13] B. Baron, P. Spathis, H. Rivano, M. D. de Amorim, Y. Viniotis, and M. H. Ammar, "Centrally controlled mass data offloading using vehicular traffic," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 401–415, 2017.

[14] F. Jiang, R. Ma, C. Sun, and Z. Gu, "Dueling deep q-network learning based computing offloading scheme for f-ran," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–6.

[15] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial iot–edge–cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[16] Y. Wang, H. Ge, A. Feng, W. Li, L. Liu, and H. Jiang, "Computation offloading strategy based on deep reinforcement learning in cloud-assisted mobile edge computing," in *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, 2020, pp. 108–113.

[17] F. Fabiani and S. Grammatico, "Multi-vehicle automated driving as a generalized mixed-integer potential game," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1064–1073, 2020.

[18] H. Liu, H. Jia, J. Chen, X. Ge, Y. Li, L. Tian, and J. Shi, "Computing resource allocation of mobile edge computing networks based on potential game theory," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, 2018, pp. 693–699.

[19] A. Raschellà, F. Bouhafs, M. Mackay, Q. Shi, and M. Canales, "Ap selection algorithm based on a potential game for large ieee 802.11 wlans," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2018)*, 2018.

[20] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.

[21] B. Wu, J. Zeng, L. Ge, Y. Tang, and X. Su, "A game-theoretical approach for energy-efficient resource allocation in mec network," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[22] T. Zhu, J. Li, Z. Cai, Y. Li, and H. Gao, "Computation scheduling for wireless powered mobile edge computing networks," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 596–605.

[23] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0899825696900445

[24] *Algorithmic Game Theory*. Cambridge University Press, 2007.

[25] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 281–294, 2021.

[26] B. Li, Q. He, G. Cui, X. Xia, F. Chen, H. Jin, and Y. Yang, "Read: Robustness-oriented edge application deployment in edge computing environment," *IEEE Transactions on Services Computing*, pp. 1–1, 2020.

[27] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distributed Comput.*, vol. 127, pp. 160–168, 2019.

[28] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Software: Practice and Experience*, vol. 50, pp. 489 – 502, 2020.

[29] J. Gedeon, J. Krisztinkovics, C. Meurisch, M. Stein, L. Wang, and M. Mühlhäuser, "A multi-cloudlet infrastructure for future smart cities: An empirical study," in *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, pp. 19–24.

[30] R. Jain, D. M. Chiu, and H. WR, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *ACM Transactions on Computer Systems*, 1984.