Wang E, Wang H, Dong PM *et al.* Distributed game-theoretical D2D-enabled task offloading in mobile edge computing. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(4): 919–941 July 2022. DOI 10.1007/s11390-022-2063-3

Distributed Game-Theoretical D2D-Enabled Task Offloading in Mobile Edge Computing

En Wang[1](王　恩), *Member*, *CCF*, Han Wang[1](王　晗), Peng-Min Dong[2](董朋民)
Yuan-Bo Xu[1,*](徐原博), *Member*, *CCF*, and Yong-Jian Yang[1](杨永健), *Member*, *CCF*

[1] *Department of Computer Science and Technology, Jilin University, Changchun 130012, China*
[2] *Department of Software, Jilin University, Changchun 130012, China*

E-mail: wangen@jlu.edu.cn; {wanghan21, dongpm19}@mails.jlu.edu.cn; {yuanbox, yyj}@jlu.edu.cn

Received December 3, 2021; accepted July 8, 2022.

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 of the existing studies only consider the congestion of wireless channels as a crucial factor affecting the strategy-making process, while ignoring the impact of offloading among edge nodes. In addition, centralized task offloading strategies result in enormous 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 and propose the Parallel User Selection Algorithm (PUS) and Single User Selection Algorithm (SUS) to substantially accelerate the convergence. More practically, we extend the users' offloading strategies to take into account idle devices and cloud services, which considers the potential computing resources at the edge. Furthermore, we construct a potential game in which each user selfishly seeks an optimal strategy to minimize its cost of latency and energy based on acceptable latency, and find the potential function to prove the existence of Nash equilibrium (NE). Additionally, we update PUS to accelerate its convergence and illustrate its performance through the experimental results of three real datasets, and the updated PUS effectively decreases the total cost and reaches Nash equilibrium.

Keywords computation offloading, potential game, Nash equilibrium, device-to-device (D2D), acceptable latency

1 Introduction

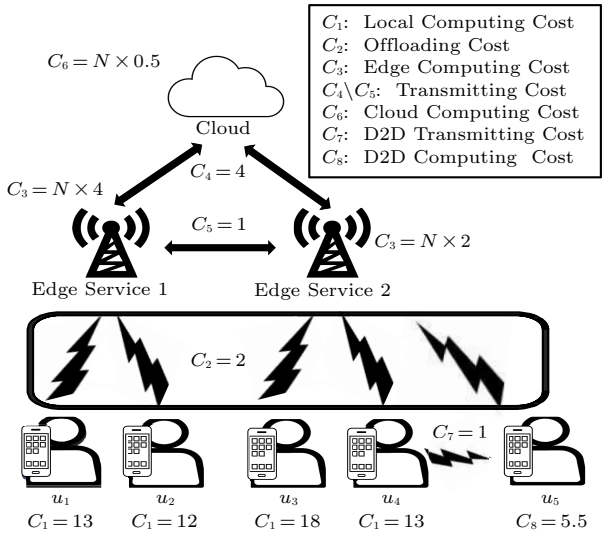
With the rapid development of 5G [1] and other network technologies, terminal devices such as face recognition, natural language processing and interactive gaming [2–5] have been involved in computing-intensive and delay-critical applications. Due to hardware limitations, the battery life and computing resources of mobile devices are usually limited. Mobile edge computing (MEC) is considered as a promising distributed computing paradigm, which uses the computing capacity [6–8] of edge nodes to decrease computing cost and improve quality of service (QoS).

In MEC, because the computing capacity and wireless channels of edge nodes and the computing capacity of cloud are shared by offloading users, the mobile users need to decide whether to offload tasks to edge nodes and the cloud, or to utilize the device-to-device (D2D) [9] technology or their local devices for processing tasks, which raises fundamental task offloading issues. Recently, some studies only consider the impact of wireless channel competition regardless of the offloading among edge nodes, which may change the offloading decision [10]. Moreover, most of the proposed task of-

Regular Paper
Special Section of MASS 2020–2021
A preliminary version of the paper was published in the Proceedings of MASS 2021.
This work was supported by the National Natural Science Foundation of China under Grant No. 62072209, the National Natural Science Foundation of China Youth Fund under Grant No. 62002123, the Key Research and Development Program of Jilin Province of China under Grant No. 20210201082GX, the Scientific and Technological Planning Project of Jilin Province of China under Grant No. JJKH20221010KJ, and the Development and Reform Commission Project of Jilin Province of China under Grant No. 2020C017-2.
*Corresponding Author
©Institute of Computing Technology, Chinese Academy of Sciences 2022

floading strategies are centralized [11, 12]. The central node needs to collect all user information and control the global task offloading strategies through calculation, which not only makes the central node have the enormous computation complexity, but also makes the user's privacy leak. From the perspective of mobile users, when they find a better way to offload tasks, the centralized strategy may not meet the needs of all users. For example, a user may be unwilling to deviate from the original offloading decision, especially when the assigned decision costs more than the former one. Nowadays, many offloading strategies do not make use of the potential computing resources of edge nodes, such as D2D technology [13], so that users' tasks are offloaded to idle mobile devices without tasks to be offloaded, which effectively decreases the total cost of users. End-edge-cloud computing [14] can not only improve resource utilization, but also enhance the quality of service. Mobile users can offload tasks to the cloud for task computing through edge service nodes or directly to the cloud through the wireless network.

Therefore, we take the offloading among edge nodes, the competition in wireless channels, cloud computing and D2D into consideration, which is a more practical situation. 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. There are some users connected through the D2D links in mobile users. Therefore, the users who have tasks can offload their tasks to idle devices which do not have tasks and each idle device is allowed to offload only one task. The tasks of mobile users can be offloaded to the cloud through their home nodes because the cloud is connected to the home edge nodes with optical fiber cables. As illustrated in Fig.1(a), user u_1 regards the edge service node 1 as his/her home node and the edge service node 2 as his/her neighbor edge node. Due to different locations and the transmission distance of optical fiber cables, each edge service node has some neighbor connecting nodes. Obviously, for different mobile users, the division of edge nodes may also be different. Both edge service node 1 and edge service node 2 have optical fiber cables connected to the cloud. Because user u_4 connects with idle mobile user u_5 through a D2D link, user u_4 can offload his/her task to user u_5 for processing. The wireless channels shown as C_2 arrows in Fig.1(a) are shared by the offloading users. Therefore, the more the mobile users using the channel to transmit data are, the lower the transmission rate of mobile users using the channel is. There are five methods for task computing including processing tasks utilizing their local devices, offloading to the home edge node, offloading to the neighbor edge node and cloud via fiber-optical wired transmission from the home edge node, and offloading to one idle device via the D2D link. When mobile users offload their tasks to neighbor edge nodes or cloud, they must first offload tasks to their home edge nodes, and then their home edge nodes will offload tasks to corresponding neighbor edge nodes or cloud.



(a)

Algorithm	Method	Cost		Equilibrium
Minimum Cost	u_1 ② u_2 ② u_3 ② u_4 ②	u_1: $2+1+4\times2=11$ u_2: $2+1+4\times2=11$ u_3: $2+4\times2=10$ u_4: $2+4\times2=10$	42	No
Centralized Optimal	u_1 ① u_2 Ⓒ u_3 ② u_4 u_5	u_1: $2+4\times1=6$ u_2: $2+4+1\times0.5=6.5$ u_3: $2+1\times2=4$ u_4: $1+5.5=6.5$	23	No
Distributed Equilibrium	u_1 Ⓒ u_2 ① u_3 ② u_4 ②	u_1: $2+4+1\times0.5=6.5$ u_2: $2+4\times1=6$ u_3: $2+2\times2=6$ u_4: $2+2\times2=6$	24.5	Yes

① Edge Service 1
② Edge Service 2
u Idle Device
Ⓛ Local Device
Ⓒ Cloud

u_4 can choose edge service 2 to decrease cost

We can find an equilibrium state

(b)

Fig.1. Problem description for task offloading. (a) Task offloading model. (b) Task offloading algorithms. \ represents "and".

In addition, we consider a distributed algorithm that allows users to choose an offloading method instead of uploading users' information to the center edge service node. Some comparing offloading algorithms are shown in Fig.1(b). An intuitive idea (minimum cost) assumes that every user thinks that there is only himself/herself and no one else in the model, and therefore every user only needs to choose an offloading scheme to minimize his/her cost. Obviously, this will lead to a higher total cost of 42. An ideal approach (centralized optimal) achieves the least total cost of 23, but the needs of all mobile users are not satisfied, i.e., user u_4 can offload his/her task to node 2 to achieve a less cost value of 6. A trade-off (distributed equilibrium) is our target method, where a relatively acceptable cost of 24.5 and an equilibrium state are achieved, and no user has the motivation to change the decision unilaterally.

Therefore, our first challenge is how to construct a distributed model, which not only gets a better total cost, but also achieves Nash equilibrium. In the process of task offloading, the requirements of user equipment for battery condition and the time urgency of the task are considered. Therefore, the second challenge is how to design a unified distributed algorithm to meet the personal conditions of all mobile users. Moreover, Fig.1(b) shows that although we can find Nash equilibrium, the total cost is not always optimal. The third challenge is to make the better total cost obtained by the algorithm have an upper bound relative to the optimal solution. In order to meet the above challenges, we first define the task offloading problem as a multi-user task offloading game where each user selects a task offloading method to minimize their own cost. Then, by constructing a global potential function, it demonstrates that the constructed game is a potential game, and thus the algorithm can achieve Nash equilibrium. The change of each user's cost can be uniformly mapped to the change of the global potential function. By constantly approaching the minimum of the global potential function, we reach an equilibrium state, in which the cost function of each user reaches a local minimum. On this basis, a distributed game theory task offloading algorithm is designed to realize Nash equilibrium. For the cost function, users can modify the weight parameters according to their specific situations. Finally, the metric of Price of Anarchy (PoA) is used to ensure the upper bound of the total cost relative to the centralized optimal solution.

In this paper, we introduce a mobile edge computing offload framework supporting D2D. As shown in Fig.1, a user can offload his/her task to the home edge server node, neighbor edge nodes, and cloud through a home base station or an idle mobile user. The main contributions are summarized as follows.

1) We propose a game theory method to solve the problem of computing offload by modeling the offload process as a multi-user task offload game considering the offload between wireless channels, edge nodes, cloud, and idle devices and meeting users' acceptable latency. We prove that the centralized optimal solution of the offloading problem is NP-hard.

2) We prove that the multi-user offloading game is a potential game, and design an algorithm to achieve Nash equilibrium. At the same time, users can modify the parameters of the cost function to meet their respective conditions and acceptable latency.

3) We further prove that our distributed algorithm can reach Nash equilibrium, and prove the upper bound of the number of update steps and the lower bound of the total cost. In addition, in order to speed up the convergence of the distributed algorithm, we update PUS and get better results in the experiment.

4) We evaluate the proposed algorithms with the D2D link and cloud on three real widely-used datasets of edge networks: Melbourne, Shanghai and Darmstadt. Experimental results show that our algorithm can reach NE in limited iterative times, and the total cost is close to the centralized optimal solution.

This paper is a journal version extended from the conference paper [15]. 1) Based on the multi-user task offload game considering the offload between wireless channels and edge nodes [15], we further extend the task offloading game to a D2D-enabled multi-user task offload game (Fig.1(a)) considering the offload including wireless channels, edge nodes, the cloud and idle devices, which takes the potential computing resources at the edge and cloud computing into consideration and is proved to be a weighted potential game in Subsection 3.8. 2) In this paper, we meet the acceptable latency of all users, that is, the completion time of a task should be within the user's acceptable latency (Section 3). 3) To decrease the convergence time and apply to the model including wireless channels, edge nodes, the cloud and idle devices, we update Distributed Game-Theoretical Task Offloading algorithm [15], Information Update algorithm [15], and PUS algorithm [15] to Algorithms 1–3 in this paper respectively. 4) We add experiments to explore the impact of the number of D2D links on total cost and offloading ratio in Subsection 5.3.3.

The remainder of the paper is organized as follows. After reviewing the related work in Section 2, we introduce the system model, the NP-hardness of the centralized problem and the potential game formulation in Section 3. Then, we propose the distributed task offloading algorithm and analyze its performance theoretically in Section 4. Finally, we conduct extensive simulations to evaluate the proposed algorithm in Section 5 and conclude the paper in Section 6.

2 Related Work

2.1 Task Offloading

The research of task offloading in MEC can be divided into centralized task offloading and distributed task offloading. For the former, Baron *et al.*[16] proposed a method of offloading multi-user tasks between multiple edges to achieve the maximum task completion rate. Jiang *et al.*[17] proposed a centralized task offloading method based on deep learning and MEC to minimize the total energy consumption. The disadvantage of the centralized task offloading method is that it cannot consider whether users are satisfied with the offloading strategy, and the computational complexity of the central node is enormous. For the latter, Hong *et al.*[18] proposed a multi-hop collaborative computing offload scheme in the edge cloud computing environment of industrial Internet of Things. Wang *et al.*[19] investigated the offloading problem and resource allocation using deep reinforcement learning. Ding *et al.*[14] proposed two types of computing architectures according to the visibility and accessibility of cloud to users: hierarchical end-edge-cloud computing and horizontal end-edge-cloud computing. Yu *et al.*[20] designed a D2D multicast computing offload framework to minimize the overall energy consumption. However, most of the existing studies do not consider the impact of offloading between edge nodes and wireless channel congestion and do not make use of the potential computing resources of edge nodes. Furthermore, we consider the impact of offloading between edge nodes, wireless channel congestion, cloud, and idle user devices, and users' acceptable latency, which is actually a more realistic scenario.

2.2 Potential Game

In recent years, many researches have used the potential game theory to make distributed game theories decisions and achieve Nash equilibrium. Fabiani and Grammatico[21] expressed the multi-vehicle driving coordination problem as a mixed-integer potential game, and obtained its equilibrium solution. Liu *et al.*[22] described the offloading problem of multi-user computing as a potential game in which mobile devices make offloading decisions in a distributed manner. Raschellá *et al.*[23] proposed an access point selection method for potential games based on the software-defined network. He *et al.*[24] studied an application user computing offload problem in mobile edge computing and proposed a potential game theory method to achieve effective computational offload. Wu *et al.*[25] defined the edge user assignment problem as a potential game and proposed a decentralized algorithm to serve the maximum number of users with the minimum total system cost. However, most studies design fixed and unified cost functions for all users. Therefore, the corresponding potential games do not take into account the diversified needs of users. In this paper, we propose a distributed game theory method, in which mobile users can achieve different preferences by adjusting the parameters of the cost function.

3 Multi-User Task Offloading Game

3.1 System Model

In this subsection, we introduce the MEC offloading framework as shown in Fig.1. In our system, there are N users denoted as $U = \{u_1, u_2, ..., u_N\}$ and S edge nodes embedded in base stations are denoted as $B = \{b_1, b_2, ..., b_S\}$. Let $L = \{l_1, l_2, ..., l_M\}(L \subset U)$ denote the set of M idle devices which provide computation support to other users who have tasks. Our model investigates the stable offloading of users' locations and edge nodes in quasi-static scenarios. Each mobile user is connected to zero or more idle devices through the D2D link, and thus the mobile user can offload his/her task to an idle device if there is a D2D link between them. Note that only one task of a mobile user is allowed to be offloaded to an idle device, i.e., if user u_4 has offloaded his/her task to user l_5, and even if there is a D2D link between user u_6 and user l_5, user u_6 cannot offload his/her task to user l_5. Each edge node is interconnected to each other through optical fiber cables, not all nodes. Moreover, all edge nodes and cloud are also connected with one another through optical fiber cables. Each user has his/her own nearest edge node called the home edge node and the edge nodes which are connected with his/her home edge node through optical fiber cables are called the user's neighbor edge

nodes. As will be readily seen, the home edge nodes and the neighbor edge nodes of different users may be different. Therefore, each user can offload his/her task to cloud or the neighbor nodes through his/her home edge node. We assume that users' tasks are generated by their daily mobile devices, such as mobile phones or tablets. According to Fig. 1, there are five ways for computing the tasks of mobile users: 1) utilizing mobile users' local computing, 2) utilizing their home edge nodes through wireless channels, 3) utilizing their neighbor edge nodes, 4) utilizing cloud computing, and 5) utilizing idle devices through the D2D link.

3.2 Local Device Computing Model

For every user $u_i \in U$, we use c_i $(i = 1, 2, ..., N)$ to denote the computing capacity of user u_i's local device. Therefore, $C = \{c_1, c_2, ..., c_N\}$ denotes the set of the computing capacities of N users' local devices. We use T_i, P_i, E_i to denote u_i's task, the energy consumption per CPU cycle, and the energy consumption per bite for offloading tasks respectively. $T_i = \{R_i, D_i, O_i, L_i\}$ denotes user u_i's task where R_i denotes the number of CPU cycles required to complete tasks T_i, D_i denotes user u_i's offloading data size, O_i denotes the output data size and L_i denotes user u_i's acceptable latency. If user u_i determines to compute task T_i in the local device, the computing time will be formulated as follows:

$$t_i^{\text{local}} = \frac{R_i}{c_i}. \tag{1}$$

Then the energy consumption for computing task T_i will be given as:

$$e_i^{\text{local}} = P_i R_i. \tag{2}$$

In order to adjust users' individual conditions, we associate two positive parameters, α_i and β_i $(\alpha_i + \beta_i = 1, \alpha_i > 0, \beta_i > 0)$, with the time cost and the energy cost of user u_i, respectively, when calculating the total cost for a user. If there is little energy left in the user u_i's battery, the user can increase the value of β_i to increase the cost of energy consumption. Equally, if u_i's time is valuable, they can increase α_i to emphasize the impact of time costs. As described above, according to (1) and (2), the total cost of u_i's local device computing can be formulated as:

$$Q_i^{\text{local}} = \alpha_i t_i^{\text{local}} + \beta_i e_i^{\text{local}}. \tag{3}$$

3.3 Home Edge Node Computing Model

We utilize $\widetilde{C} = \{\widetilde{c}_1, \widetilde{c}_2, ..., \widetilde{c}_S\}$ to denote the set of the computing capacities of all edge nodes. Let $h(u_i)(u_i \in U)$ denote the home edge node of user u_i and $N_{h(u_i)}(u_i \in U)$ denote the number of users who offload the task to u_i's home edge node. If u_i chooses the home edge node to compute the task, the computing time consists of task computing time and task offloading time. The task computing time will be formulated as follows:

$$t_{h(u_i)}^{\text{exe}} = \frac{R_i}{(\frac{\widetilde{c}_{h(u_i)}}{N_{h(u_i)}})} = \frac{R_i N_{h(u_i)}}{\widetilde{c}_{h(u_i)}}.$$

The computing power of home edge service nodes is equally shared. Meanwhile, the task offloading time is as follows:

$$t_i^{\text{off}} = \frac{D_i}{r_i} + AD_i U(i),$$

where 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. Considering that the high transmission power of the edge service nodes makes the downlink data rate high enough, we omit the downlink transmission time. As mentioned earlier, we also need to take energy consumption into consideration. The transmission energy consumption can be given as:

$$e_i^{\text{off}} = E_i D_i.$$

The total cost of u_i's home edge node computing can be formulated as:

$$Q_{h(i)} = \alpha_i (t_{h(u_i)}^{\text{exe}} + t_i^{\text{off}}) + \beta_i E_i D_i. \tag{4}$$

As same as (3), α_i and β_i $(\alpha_i + \beta_i = 1, \alpha_i > 0, \beta_i > 0)$ denote the weight of the time cost and the energy cost of user u_i respectively.

3.4 Neighbor Edge Node Computing Model

If user u_i offloads his/her task to a neighbor edge node for execution, his/her task is transmitted to the home edge node through a wireless channel in the first place, and then the home edge node transmits the task to the corresponding neighbor edge node. Obviously, compared with the home edge node computing model, the transmission time of the neighbor edge node computing model is different from the latency from the home edge node to the neighbor edge node. Therefore, we utilize B_{ab} to denote the transmission time between home edge node a and neighbor edge node b. Similarly, let $n(u_i)(u_i \in U)$ denote the neighbor edge node

selected by user u_i and $N_{n(u_i)}(u_i \in U)$ denote the number of users who offload the tasks to the neighbor edge node selected by user u_i. The task computing time will be formulated as follows:

$$t^{\text{exe}}_{n(u_i)} = \frac{R_i}{(\frac{\widetilde{c}_{n(u_i)}}{N_{n(u_i)}})} = \frac{R_i N_{n(u_i)}}{\widetilde{c}_{n(u_i)}}.$$

The computing power of every neighbor edge service node is equally shared. We might as well assume that user u_i chooses to offload task T_i to neighbor edge node b through home edge node a. Meanwhile, the task offloading time is as follows:

$$t^{\text{nei\_off}}_i = \frac{D_i}{r_i} + AD_iU(i) + B_{ab}.$$

Similarly, the transmission energy consumption can be given as:

$$e^{\text{nei\_off}}_i = E_iD_i.$$

The total cost of u_i's neighbor edge node computing can be formulated as:

$$Q_{n(i)} = \alpha_i(t^{\text{exe}}_{n(u_i)} + t^{\text{nei\_off}}_i) + \beta_i e^{\text{nei\_off}}_i. \quad (5)$$

α_i and β_i $(\alpha_i + \beta_i = 1, \alpha_i > 0, \beta_i > 0)$ denote the weight of the time cost and the energy cost of user u_i respectively.

3.5 Cloud Computing Model

In the case of cloud computing, the operation mechanism is that the user firstly transmits a task to his/her home edge node, and then the home edge node transmits it to the cloud. The computing capacity of cloud is denoted as $\hat{c}$. Obviously, compared with the home edge node computing model, the transmission time of the cloud computing model is different from the latency from the home edge node to the cloud. We utilize B^{cloud}_a to denote the transmission time between home edge node a and the cloud. The cloud computing capacity also decreases with the increasing of the number of users selecting the cloud computing. The number of users who select the cloud computing is denoted as N_{cloud}. The task computing time will be formulated as follows:

$$t^{\text{cloud\_exe}}_i = \frac{R_i}{(\frac{\hat{c}}{N_{\text{cloud}}})} = \frac{R_i N_{\text{cloud}}}{\hat{c}}. \quad (6)$$

We might as well assume that user u_i chooses to offload the task to the cloud through home edge node a. Meanwhile, the task offloading time is as follows:

$$t^{\text{cloud\_off}}_i = \frac{D_i}{r_i} + AD_iU(i) + B^{\text{cloud}}_a. \quad (7)$$

The transmission energy consumption can be formulated as:

$$e^{\text{cloud\_off}}_i = E_iD_i. \quad (8)$$

Following (6), (7) and (8), the total cost of u_i's cloud computing can be formulated as:

$$Q^{\text{cloud}}_i = \alpha_i(t^{\text{cloud\_exe}}_i + t^{\text{cloud\_off}}_i) + \beta_i e^{\text{cloud\_off}}_i. \quad (9)$$

3.6 D2D Computing Model

In D2D offloading, one D2D link is established between the user who has a task and one idle user's device. It is possible for every mobile user to connect to zero or more idle devices through the D2D links. We assume that only one task of a mobile user is allowed to be offloaded to an idle device, and thus one idle device can only compute one task at most. Because the data rates of the D2D communication are not equal and the transmission power of terminal equipment is lower than that of edge service nodes, the output latency in the transmission time is considered. We might as well assume that user u_i chooses to offload his/her task to idle device u_j. The transmitting rate from user u_i to idle device u_j is $r_{i,j}$. On the contrary, the transmitting rate from idle device u_j to user u_i is $r_{j,i}$. Obviously, $r_{i,j} \neq r_{j,i}$. Therefore, if user u_i offloads his/her task to idle device u_j, the transmission time is shown as:

$$t^{\text{d2d\_off}}_{i,j} = \frac{D_i}{r_{i,j}} + \frac{O_i}{r_{j,i}}.$$

The execution time in idle device u_j is:

$$t^{\text{exe}}_{i,j} = \frac{R_i}{c_j}.$$

The transmission energy consumption can be given as:

$$e^{\text{d2d\_off}}_{i,j} = E_iD_i.$$

Then the total cost in the D2D offloading model is given as:

$$Q^{\text{d2d}}_{i,j} = \alpha_i(t^{\text{exe}}_{i,j} + t^{\text{d2d\_off}}_{i,j}) + \beta_i e^{\text{d2d\_off}}_{i,j}. \quad (10)$$

In addition, for the convenience of readers, we summarize the symbols commonly used in this paper in Table 1.

Table 1. Description of Major Notations

Variable	Description
N, U, u_i	Number of all users, set of all users, and index of user i, respectively
S, B, b_s	Number of all edge nodes, set of all edge nodes, and index of edge node s, respectively
M, L, l_m	Number of all idle devices, set of all idle devices, and index of idle device, respectively
c_i, C	Computing capacity of user u_i's local device and set of users' computing capacity, respectively
T_i, P_i, E_i	u_i's task, energy consumption per CPU cycle and energy consumption per bite for offloading task respectively
R_i, D_i	Number of CPU cycles required to complete tasks T_i and offloading data size of T_i, respectively
O_i, L_i	Output data size of T_i and acceptable latency of u_i, respectively
t_i^{local}, e_i^{local}, Q_i^{local}	Computing time, energy consumption, and total cost of u_i's local device computing respectively
α_i, β_i	Weight parameters of time cost and energy cost respectively
$\widetilde{c}_s$, $\widetilde{C}$	Computing capacity of edge node b_s and set of edge nodes' computing capacities, respectively
$h(u_i)$	Home edge node of user u_i
$N_{h(u_i)}$	Number of users who offload the task to $h(u_i)$
r_i, $U(i)$	Transmission rate of u_i, and number of users that share the channel selected by user u_i, respectively
$t_{h(u_i)}^{\text{exe}}$, t_i^{off}	Computing time and task offloading time for home edge node computing of u_i, respectively
e_i^{off}	Transmission energy consumption for home edge node computing of u_i
$Q_{h(i)}$	Total cost for home edge node computing of u_i
B_{ab}	Transmission time between home edge node a and neighbor edge node b
$n(u_i)$	Neighbor edge node of user u_i
$N_{n(u_i)}$	Number of users who offload the task to $n(u_i)$
$t_{n(u_i)}^{\text{exe}}$	Computing time for neighbor edge node computing of u_i
$t_i^{\text{nei\_off}}$	Task offloading time for neighbor edge node computing of u_i
$e_i^{\text{nei\_off}}$, $Q_{n(i)}$	Transmission energy consumption and total cost for neighbor edge node computing of u_i, respectively
$\hat{c}$, N_{cloud}	Computing capacity of cloud and number of users who select cloud computing, respectively
B_a^{cloud}	Transmission time between home edge node a and the cloud
$t_i^{\text{cloud\_exe}}$	Computing time for cloud computing of u_i
$t_i^{\text{cloud\_off}}$	Task offloading time for cloud computing of u_i
$e_i^{\text{cloud\_off}}$, Q_i^{cloud}	Transmission energy consumption and total cost for cloud computing of u_i, respectively
$r_{i,j}$	Transmitting rate from user u_i to idle device u_j
$t_{i,j}^{\text{exe}}$	Computing time when user u_i offloads the task to idle device u_j
$t_{i,j}^{\text{d2d\_off}}$	Task offloading time when user u_i offloads the task to idle device u_j
$e_{i,j}^{\text{d2d\_off}}$, $Q_{i,j}^{\text{d2d}}$	Transmission energy consumption and total cost when user u_i offloads the task to idle device u_j, respectively
A	Congestion parameter
$\hat{u}$	Set of users who send requests to update offloading strategies
$\boldsymbol{s}$, s_i, $\boldsymbol{s}_{-i}$	$\boldsymbol{s} = (s_i, \boldsymbol{s}_{-i})$, user u_i's strategy, and others' strategies, respectively

3.7 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 $\boldsymbol{s} = (s_i, \boldsymbol{s}_{-i})$ (i.e., s_i denotes user u_i's strategy and $\boldsymbol{s}_{-i}$ denotes others' strategies), the problem can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & \textstyle\sum_{u_i \in U} Q_i(\boldsymbol{s}) \\ \text{subject to} \quad & s_i \in S_i, \forall u_i \in U, \\ & t_i^{\text{total}} < L_i, \end{aligned}$$

where S_i means the optional strategies of user u_i, Q_i is user u_i's total cost when user u_i chooses the strategy s_i, and t_i^{total} means that the total real time when the task is completed cannot exceed user u_i's acceptable latency L_i. 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) [26] 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 $p_{i,j}$, respectively.

Output. The assignment of items to knapsacks which will reach the optimal total profits without ex-

ceeding 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, cloud computing or D2D offloading and e_i denotes the cost of edge node computation, cloud computing or D2D offloading. When the user number of a wireless channel is large enough so that $e_i = c_i$, the number of users at this time is the task's striction of item i. The size of task i is regarded as the size of item, and the capacity of each channel is tasks' striction. Now that the maximization version of GAP is NP-hard, our problem is also NP-hard. □

3.8 Potential Game Formulation

In this section, we first introduce some definitions, and then utilize the distributed task offloading method to express our model as a potential game [27, 28].

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

$$Q_i(\hat{s}_i, \hat{\boldsymbol{s}}_{-i}) = \min\ Q_i(s_i, \hat{\boldsymbol{s}}_{-i})\ \forall u_i \in U, \forall s_i \in S_i.$$

Obviously, in a Nash equilibrium state, when a user changes to any other strategy, the user's cost will increase, and therefore the user has no motivation to reduce the task completion cost by unilaterally changing the strategy.

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 $\forall i \in U$ satisfying*:

$$\begin{aligned} &Q_i(s_i, \boldsymbol{s}_{-i}) - Q_i(\acute{s}_i, \boldsymbol{s}_{-i}) \\ &= \mu_i(\delta(s_i, \boldsymbol{s}_{-i}) - \delta(\acute{s}_i, \boldsymbol{s}_{-i})) \forall s_i, \forall \acute{s}_i \in S_i,\ \forall \boldsymbol{s}_{-i} \in \boldsymbol{S}_{-i}, \end{aligned}$$

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, and 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:

$$\begin{aligned} &\delta(\boldsymbol{s}) \\ &= \sum_{n\in\gamma}\sum_{j=0}^{C(n)} Aj + \sum_{b\in\mathcal{B}}\sum_{j=0}^{|N_b|} q\frac{j}{\tilde{c}_b} + \sum_{j=0}^{|N_c|} q\frac{j}{\hat{c}} + \\ &\sum_{i\in\mathcal{U}} X_i I(a_i, 0) + \sum_{i\in\mathcal{U}} Y_i I(a_i, 1) + \sum_{i\in\mathcal{U}} Z_i I(a_i, 2) + \\ &\sum_{i\in\mathcal{U}} W_i I(a_i, 3) + \sum_{i\in\mathcal{U}} V_i I(a_i, 4), \end{aligned} \tag{11}$$

where γ is the wireless channel set, $q = R_i/D_i$ denotes the CPU cycles required to process per data size, $X_i = q(\frac{1}{c_i} + \frac{\beta_i P_i}{\alpha_i})$, $Y_i = \frac{\beta_i E_i}{\alpha_i} + \frac{1}{r_i}$, $Z_i = \frac{\beta_i E_i}{\alpha_i} + \frac{1}{r_i} + \frac{B_{cd}}{D_i}$, $W_i = \frac{1}{r_{i,j}} + \frac{O_i}{D_i r_{j,i}} + \frac{q}{c_j} + \frac{\beta_i E_i}{\alpha_i}$, $V_i = \frac{1}{r_i} + \frac{\beta_i E_i}{\alpha_i} + \frac{B_e^{\text{cloud}}}{D_i}$ and $I(a_i, m)$ is an indicator function defined as:

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

Here, $m = 0$ means that users process tasks locally, $m = 1$ means that users choose to offload their tasks to home edge nodes, $m = 2$ means that users choose to offload their tasks to neighbor edge nodes, $m = 3$ means that users choose to offload their tasks to idle devices through the D2D link and $m = 4$ means that users choose to offload their tasks to the cloud through home edge nodes. Moreover, a_i denotes user u_i's strategy assigned from the set $\{0, 1, 2, 3, 4\}$.

We define the original strategy profile of user u_i and the other users as $\boldsymbol{s} = (s_i, \boldsymbol{s}_{-i})$. When user u_i changes the strategy into $\acute{s}_i$ while the others keep stable, the strategy profile becomes $\boldsymbol{s} = (\acute{s}_i, \boldsymbol{s}_{-i})$. Considering all the user u_i's strategy changing situations, we will discuss the following 13 cases: 1) from the local device to the home edge node; 2) from the local device to the neighbor edge node; 3) from the home edge node to a 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; 6) from the local device to D2D offloading; 7) from D2D offloading to the home edge node; 8) from D2D offloading to a neighbor edge node; 9) from D2D offloading i to D2D offloading j; 10) from D2D offloading to the cloud; 11) from the local device to the cloud; 12) from a home edge node to the cloud; and 13) from a neighbor edge node to the cloud. Obviously, reversing the order of each case does not affect the result.

For case 1, we assume that user u_i offloads his/her task to home edge node $h_{(u_i)}$. According to (3) and (4), we have:

$$\begin{aligned}
&Q_i(\acute{s}_i) - Q_i(s_i)\\
&= Q_{h(i)} - Q_i^{\text{local}}\\
&= \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + AD_iU(i)\right) + \beta_i E_i D_i\right) -\\
&\quad \left(\alpha_i \frac{R_i}{c_i} + \beta_i P_i R_i\right)\\
&= \alpha_i D_i\left(\frac{1}{r_i} + q\frac{N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + AU(i) -\right.\\
&\quad \left. q\left(\frac{1}{c_i} + \frac{\beta_i P_i}{\alpha_i}\right)\right)\\
&= \alpha_i D_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)).
\end{aligned}$$

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:

$$\begin{aligned}
&Q_i(\acute{s}_i) - Q_i(s_i)\\
&= Q_{n(i)} - Q_i^{\text{local}}\\
&= \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{n(u_i)}}{\widetilde{c}_{n(u_i)}} + B_{cd} + AD_iU(i)\right) +\right.\\
&\quad \left.\beta_i E_i D_i\right) - \left(\alpha_i \frac{R_i}{c_i} + \beta_i P_i R_i\right)\\
&= \alpha_i D_i\left(\frac{1}{r_i} + q\frac{N_{n(u_i)}}{\widetilde{c}_{n(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + \frac{B_{cd}}{D_i} +\right.\\
&\quad \left. AU(i) - q\left(\frac{1}{c_i} + \frac{\beta_i P_i}{\alpha_i}\right)\right)\\
&= \alpha_i D_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)).
\end{aligned}$$

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

$$\begin{aligned}
&Q_i(\acute{s}_i) - Q_i(s_i)\\
&= Q_{n(i)} - Q_{h(i)}\\
&= \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{n(u_i)}}{\widetilde{c}_{n(u_i)}} + B_{cd} + AD_iU(i)\right) +\right.\\
&\quad \left.\beta_i E_i D_i\right) - \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + AD_iU(i)\right) +\right.\\
&\quad \left.\beta_i E_i D_i\right)\\
&= \alpha_i D_i\left(\frac{1}{r_i} + q\frac{N_{n(u_i)}}{\widetilde{c}_{n(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + \frac{B_{cd}}{D_i} -\right.\\
&\quad \left.\left(\frac{1}{r_i} + q\frac{N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i}\right)\right)\\
&= \alpha_i k_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)).
\end{aligned}$$

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

$$\begin{aligned}
&Q_i(\acute{s}_i) - Q_i(s_i)\\
&= \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + AD_iU(i_b)\right) + \beta_i E_i D_i\right) -\\
&\quad \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + Ak_iU(i_a)\right) + \beta_i E_i D_i\right)\\
&= \alpha_i D_i\left(\frac{1}{r_i} + q\frac{N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + AU(i_b) -\right.\\
&\quad \left.\left(\frac{1}{r_i} + q\frac{N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + AU(i_a)\right)\right)\\
&= \alpha_i D_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)).
\end{aligned}$$

For case 5, by the similar argument in case 4, the formula also easily holds in the situation of neighbor edge nodes.

For case 6, we might as well suppose that user u_i chooses to offload his/her task to idle user u_j. Then we have:

$$\begin{aligned}
&Q_i(\acute{s}_i) - Q_i(s_i)\\
&= Q_{i,j}^{\text{d2d}} - Q_i^{\text{local}}\\
&= \left(\alpha_i\left(\frac{R_i}{c_j} + \frac{D_i}{r_{i,j}} + \frac{O_i}{r_{j,i}}\right) + \beta_i E_i D_i\right) -\\
&\quad \left(\alpha_i \frac{R_i}{c_i} + \beta_i P_i R_i\right)\\
&= \alpha_i D_i\left(\left(\frac{q}{c_j} + \frac{1}{r_{i,j}} + \frac{O_i}{D_i r_{j,i}} + \frac{\beta_i E_i}{\alpha}\right) -\right.\\
&\quad \left. q\left(\frac{1}{c_i} + \frac{\beta_i P_i}{\alpha_i}\right)\right)\\
&= \alpha_i D_i(\delta(\acute{s}_i) - \delta(s_i)) = \mu(\delta(\acute{s}_i) - \delta(s_i)).
\end{aligned}$$

For case 7, we assume that user u_i's offloading object changes from idle device u_j to u_i's home edge node. Then we have:

$$\begin{aligned}
&Q_i(\acute{s}_i) - Q_i(s_i)\\
&= Q_{h(i)} - Q_{i,j}^{\text{d2d}}\\
&= \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + AD_iU(i)\right) + \beta_i E_i D_i\right) -\\
&\quad \left(\alpha_i\left(\frac{R_i}{c_j} + \frac{D_i}{r_{i,j}} + \frac{O_i}{r_{j,i}}\right) + \beta_i E_i D_i\right)\\
&= \alpha_i D_i\left(\left(\frac{1}{r_i} + q\frac{N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i}\right) - \left(\frac{q}{c_j} + \frac{1}{r_{i,j}} +\right.\right.\\
&\quad \left.\left.\frac{O_i}{D_i r_{j,i}} + \frac{\beta_i E_i}{\alpha}\right)\right)
\end{aligned}$$

$$= \alpha_i D_i(\delta(s_i') - \delta(s_i)) = \mu(\delta(s_i') - \delta(s_i)).$$

For case 8, we assume that user u_i's offloading object changes from idle device u_j to u_i's neighbor edge node b and u_i's home edge node is a. Then we have:

$$\begin{aligned}
&Q_i(s_i') - Q_i(s_i)\\
&= Q_{n(i)} - Q_{i,j}^{\mathrm{d2d}}\\
&= \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{n(u_i)}}{\widetilde{c}_{n(u_i)}} + B_{ab} + AD_iU(i)\right) + \right.\\
&\quad \left.\beta_i E_i D_i\right) - \left(\alpha_i\left(\frac{R_i}{c_j} + \frac{D_i}{r_{i,j}} + \frac{O_i}{r_{j,i}}\right) + \beta_i E_i D_i\right)\\
&= \alpha_i D_i\left(\left(\frac{1}{r_i} + q\frac{N_{n(u_i)}}{\widetilde{c}_{n(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + \frac{B_{cd}}{D_i}\right) - \right.\\
&\quad \left.\left(\frac{q}{c_j} + \frac{1}{r_{i,j}} + \frac{O_i}{D_i r_{j,i}} + \frac{\beta_i E_i}{\alpha}\right)\right)\\
&= \alpha_i D_i(\delta(s_i') - \delta(s_i))\\
&= \mu(\delta(s_i') - \delta(s_i)).
\end{aligned}$$

For case 9, we assume that user u_i's offloading object changes from idle device u_j to another idle device u_k. Then we have:

$$\begin{aligned}
&Q_i(s_i') - Q_i(s_i)\\
&= Q_{i,k}^{\mathrm{d2d}} - Q_{i,j}^{\mathrm{d2d}}\\
&= \left(\left(\alpha_i\left(\frac{R_i}{c_k} + \frac{D_i}{r_{i,k}} + \frac{O_i}{r_{k,i}}\right) + \beta_i E_i D_i\right) + \beta_i E_i D_i\right) -\\
&\quad \left(\alpha_i\left(\frac{R_i}{c_j} + \frac{D_i}{r_{i,j}} + \frac{O_i}{r_{j,i}}\right) + \beta_i E_i D_i\right)\\
&= \alpha_i D_i\left(\left(\frac{q}{c_k} + \frac{1}{r_{i,k}} + \frac{O_i}{D_i r_{k,i}} + \frac{\beta_i E_i}{\alpha}\right) - \right.\\
&\quad \left.\left(\frac{q}{c_j} + \frac{1}{r_{i,j}} + \frac{O_i}{D_i r_{j,i}} + \frac{\beta_i E_i}{\alpha}\right)\right)\\
&= \alpha_i D_i(\delta(s_i') - \delta(s_i))\\
&= \mu(\delta(s_i') - \delta(s_i)).
\end{aligned}$$

For case 10, we assume that user u_i's offloading object changes from idle device u_j to cloud. Then we have:

$$\begin{aligned}
&Q_i(s_i') - Q_i(s_i)\\
&= Q_i^{\mathrm{cloud}} - Q_{i,j}^{\mathrm{d2d}}\\
&= \left(\alpha_i\left(\frac{R_i N_{\mathrm{cloud}}}{\hat{c}} + \frac{D_i}{r_i} + AD_iU(i) + B_a^{\mathrm{cloud}}\right) + \right.\\
&\quad \left.\beta_i E_i D_i\right) - \left(\alpha_i\left(\frac{R_i}{c_j} + \frac{D_i}{r_{i,j}} + \frac{O_i}{r_{j,i}}\right) + \beta_i E_i D_i\right)\\
&= \alpha_i D_i\left(\left(\frac{qN_{\mathrm{cloud}}}{\hat{c}} + \frac{1}{r_i} + AU(i) + \frac{B_a^{\mathrm{cloud}}}{D_i}\right) - \right.\\
&\quad \left.\left(\frac{q}{c_j} + \frac{1}{r_{i,j}} + \frac{O_i}{D_i r_{j,i}} + \frac{\beta_i E_i}{\alpha}\right)\right)\\
&= \alpha_i D_i(\delta(s_i') - \delta(s_i))\\
&= \mu(\delta(s_i') - \delta(s_i)).
\end{aligned}$$

For case 11,

$$\begin{aligned}
&Q_i(s_i') - Q_i(s_i)\\
&= Q_i^{\mathrm{cloud}} - Q_i^{\mathrm{local}}\\
&= \left(\alpha_i\left(\frac{R_i N_{\mathrm{cloud}}}{\hat{c}} + \frac{D_i}{r_i} + AD_iU(i) + B_a^{\mathrm{cloud}}\right) + \right.\\
&\quad \left.\beta_i E_i D_i\right) - \left(\alpha_i\frac{R_i}{c_i} + \beta_i P_i R_i\right)\\
&= \alpha_i D_i\left(\left(\frac{qN_{\mathrm{cloud}}}{\hat{c}} + \frac{1}{r_i} + AU(i) + \frac{B_a^{\mathrm{cloud}}}{D_i}\right) - \right.\\
&\quad \left.q\left(\frac{1}{c_i} + \frac{\beta_i P_i}{\alpha_i}\right)\right)\\
&= \alpha_i D_i(\delta(s_i') - \delta(s_i))\\
&= \mu(\delta(s_i') - \delta(s_i)).
\end{aligned}$$

For case 12, we assume that user u_i's offloading object changes from u_i's home edge node a to cloud. Then we have:

$$\begin{aligned}
&Q_i(s_i') - Q_i(s_i)\\
&= Q_i^{\mathrm{cloud}} - Q_{h(i)}\\
&= \left(\alpha_i\left(\frac{R_i N_{\mathrm{cloud}}}{\hat{c}} + \frac{D_i}{r_i} + AD_iU(i) + B_a^{\mathrm{cloud}}\right) + \right.\\
&\quad \left.\beta_i E_i D_i\right) - \left(\alpha_i\left(\frac{D_i}{r_i} + \frac{R_i N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + \right.\right.\\
&\quad \left.\left.AD_iU(i)) + \beta_i E_i D_i\right)\right)\\
&= \alpha_i D_i\left(\left(\frac{qN_{\mathrm{cloud}}}{\hat{c}} + \frac{1}{r_i} + AU(i) + \frac{B_a^{\mathrm{cloud}}}{D_i}\right) - \right.\\
&\quad \left.\left(\frac{1}{r_i} + q\frac{N_{h(u_i)}}{\widetilde{c}_{h(u_i)}} + \frac{\beta_i E_i}{\alpha_i}\right)\right)\\
&= \alpha_i D_i(\delta(s_i') - \delta(s_i))\\
&= \mu(\delta(s_i') - \delta(s_i)).
\end{aligned}$$

For case 13, we assume that user u_i's offloading object changes from neighbor edge node b to the cloud and u_i's home edge node is a. Then we have:

$$\begin{aligned}
&Q_i(s_i') - Q_i(s_i)\\
&= Q_i^{\mathrm{cloud}} - Q_{n(i)}\\
&= \left(\alpha_i\left(\frac{R_i N_{\mathrm{cloud}}}{\hat{c}} + \frac{D_i}{r_i} + AD_iU(i) + B_a^{\mathrm{cloud}}\right) + \right.
\end{aligned}$$

$$\beta_i E_i D_i\Big) - \Big(\alpha_i\Big(\frac{D_i}{r_i} + \frac{R_i N_{n(u_i)}}{\widetilde{c}_{n(u_i)}} + B_{cd} + AD_iU(i)\Big) + \beta_i E_i D_i\Big)$$

$$= \alpha_i D_i\Big(\Big(\frac{qN_{\text{cloud}}}{\hat{c}} + \frac{1}{r_i} + AU(i) + \frac{B_a^{\text{cloud}}}{D_i}\Big) - \Big(\frac{1}{r_i} + q\frac{N_{n(u_i)}}{\widetilde{c}_{n(u_i)}} + \frac{\beta_i E_i}{\alpha_i} + \frac{B_{cd}}{D_i}\Big)\Big)$$

$$= \alpha_i D_i(\delta(\acute{s_i}) - \delta(s_i))$$

$$= \mu(\delta(\acute{s_i}) - \delta(s_i)). \qquad \square$$

4 Algorithm Design

In this section, we will update our original algorithms (Distributed Game-Theoretical Task Offloading algorithm [15], Information Update algorithm [15], and PUS [15]) to take D2D offloading and cloud computing into consideration. In order to get a better total cost and meet all mobile users, Algorithm 1 is a distributed game-theoretical D2D-enabled task offloading algorithm for mobile users; Algorithm 2 is an information update algorithm for edge nodes. When Algorithm 2 terminates, a Nash equilibrium will be reached and we can get a better total cost. Because there are local computing, home edge nodes, neighbor edges, idle devices and the cloud in our current model, we update Algorithm 3 (PUS) to accelerate the convergence of Algorithm 2.

Algorithm 1. Distributed Game-Theoretical D2D-Enabled Task Offloading Algorithm for User $i \in U$

1: Input α_i, β_i, L_i
2: Set $s_i(0) = local$
3: Report $s_i(0)(u_i \in U)$ to h_{u_i}
4: Receive $N_{b_s}(b_s \in B)$, N_{cloud}, $U(i)(u_i \in U)$ and L from h_{u_i}
5: Calculate Q_i^{local}, $Q_{h(i)}$, $Q_{n(i)}$, Q_i^{cloud}, $Q_{i,j}^{\text{d2d}}$ based on (3), (4), (5), (9), (10) respectively
6: **repeat** for each decision slot t
7: Obtain $N_{b_s}(b_s \in B)$, N_{cloud}, $U(i)$ and L
8: Compute the better strategy $\triangle_i(t)$ under $t_i^{\text{total}} < L_i$ based on (3), (4), (5), (9), (10)
9: **if** $\triangle_i(t) \neq \emptyset$ and $\triangle_i(t) \neq s_i(t-1)$ **then**
10: Send update request to h_{u_i}
11: **if** Win the opportunity **then**
12: Update $s_i(t) = \triangle_i(t)$
13: Report $s_i(t)$ to the edge node
14: **else**
15: Maintain $s_i(t) = s_i(t-1)$
16: **until** the termination message is received

Algorithm 2. Information Update Algorithm for Edge Nodes

1: Receive $s_i(0)$ $(u_i \in U)$
2: Exchange $N_{b_s}(b_s \in B)$, N_{cloud}, $U(i)(u_i \in U)$ and L between edge nodes and cloud
3: **repeat** for each decision slot t
4: Receive $\hat{u}$
5: **if** $\hat{u} \neq \emptyset$ **then**
6: Select a user u_i randomly from $\hat{u}$
7: Inform the user u_i to update the decision
8: Receive $s_i(t)(u_i \in U)$ and update $N_{b_s}(b_s \in B)$, N_{cloud} and $U(i)(u_i \in U)$
9: **until** $\hat{u} = \emptyset$
10: Send the termination message to all users

Algorithm 3. Parallel User Selection Algorithm

Input: U', B
1: Initialize $\mu = \emptyset$
2: **for all** $j \in B$ **do**
3: $l' = \emptyset$, $e' = \emptyset$
4: **for all** $i \in U'$ **do**
5: **if** $s_i(t) =$ local and $i \in j$ **then** $l' \leftarrow l' \cup i$
6: **else if** $s_i(t) =$ idle device and $s_i(t-1) =$ home edge node, neighbor edge node or cloud **then** $l' \leftarrow l' \cup i$
7: **else if** $s_i(t) = j$ **then** $e' \leftarrow e' \cup i$
8: **else if** $s_i(t) =$ cloud and $h(i) = j$ **then** $e' \leftarrow e' \cup i$
9: Randomly select a user $m \in l'$ and $\mu \leftarrow \mu \cup m$
10: Randomly select a user $n \in e'$ and $\mu \leftarrow \mu \cup n$
11: $U' \leftarrow U' - m - n$
12: **for all** $k \in L$ **do**
13: $I' = \emptyset$
14: **for all** $i \in U'$ **do**
15: **if** $s_i(t) = u_k$ and $s_i(t-1)$= local or idle device **then** $I' \leftarrow I' \cup i$
16: Randomly select a user $p \in I'$ and $\mu \leftarrow \mu \cup p$
17: $U' \leftarrow U' - p$
18: **return** μ

4.1 Distributed Task D2D-Enabled 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 Algorithm 1 is to select a group of mobile users by updating the task offloading decision in each decision slot to reduce their cost and adjusting their strategy to meet their acceptable latency.

In the initialization phase (line 1) of Algorithm 1, mobile users first input two weighted parameters of the battery condition and the time urgency of the tasks and their acceptable latency, and then they initialize their strategies of computing at their local devices (line 2). $s_i(t)$ denotes the offloading strategy of user u_i in the slot t. In the information exchange stage (lines 3–5),

each user reports his/her offloading strategy ($s_i(0)$) to his/her home edge node, and gets the number of tasks offloaded to each edge and the cloud (N_{b_s} and N_{cloud} respectively), the number of users that share the channel selected by each user($U(i)(u_i \in U)$) and the messages of idle devices (L). In the calculation phase (lines 6–16), each user receives the number of tasks offloaded to each edge node and the cloud, the number of users that share the channel selected by each user, and the messages of idle devices. In this way, every user can calculate a better response strategy under the condition that the total time does not exceed the acceptable latency (lines 7 and 8). Because some users change their offloading strategies to offload their tasks to some edge service nodes or the cloud, the execution time of other users' tasks may exceed their own acceptable latency. The timeout users can find the other offloading strategy of the minimum cost under the condition of meeting the acceptable latency. Obviously, users will always find a strategy to complete tasks in their acceptable time and the worst-case scenario is when the user chooses local computing, because a user's acceptable time will not be less than the time of local computing. If a user finds a better offloading strategy which is different from the chosen offloading strategy, the user sends a request to the home edge node applying for updating the decision (lines 9–15). If a user is selected, he/she updates the decision in the next slot (lines 11–13). Other users will keep the decision in the previous decision slot (lines 14 and 15). The process repeats until users receive the termination message (line 16) (i.e., no user sends the updating request to nodes). Then the algorithm converges to a Nash equilibrium. Please note that our algorithm is a task offloading algorithm, and thus idle devices do not need to execute it.

4.2 Information Update Algorithm

As shown in Algorithm 2, after receiving the initial decision from all users (line 1), we update the information of the algorithm to update the number of tasks offloaded to each edge node and the cloud, the number of users that share the channel selected by each user, and the messages idle devices (i.e., whether other users have chosen to offload the task to the idle device) (line 2). In the initialization phase, after receiving the initial decisions from all users (line 3), the algorithm updates the number of tasks offloaded to each edge node and the cloud, the number of users that share the channel selected by each user, and the state of each idle device (i.e., whether the task of another user has been offloaded to this idle device), and then sends them to users, because an idle device only can be offloaded one task. Next, when receiving users' updating requests, we let $\hat{u}$ denote the set of users who send the request (line 4) and the algorithm will choose one user to update his/her strategy (lines 5–8). Every user will not choose a non-idle device to offload his/her task. The algorithm terminates until no request is received (line 9), and then it sends the termination messages to all users (line 10).

Furthermore, we introduce the following two user selection algorithms, Parallel User Selection Algorithm (PUS) and Single User Selection Algorithm (SUS)[15]. SUS randomly selects only one user from the set of users who send the updating requests and allows the user to update the decision in next decision slot. To decrease the convergence time, we propose PUS which only takes all local devices, edge nodes and channels into consideration[15]. The performance of PUS can be shown in our experiments. In this paper, we update PUS to take all local devices, edge nodes, the cloud, idle devices and channels into consideration based on the same theory that some users whose strategies cover no overlapping channels, edge nodes, and the cloud could simultaneously update offloading strategy in the same decision slot.

The detailed description of Algorithm is as follows. As shown in Algorithm 3, the inputs are U' and B. Specifically, U' is the set of users sending the updating requests. In the initialization phase (line 1), we make set μ composed of users who are allowed to update their offloading strategies an empty set. Then we traverse each edge node to find if there exist users whose updating strategies involve this node (lines 2–15). If a 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). If a user changes the strategy to an idle device and the user's original strategy is the home edge node, a neighbor edge node, or the cloud, the user also is added to set l' (line 6). Meanwhile, if the user's updating strategy is the traversing edge node no matter whether the edge node is his/her home edge node or not, he/she will be added to set e' (line 7). However, if the edge node is his/her home edge and the user's updating strategy is the cloud, he/she will also be added to set e' (line 8). We will randomly select two users belonging to set l' and e' respectively and add them to set μ (lines 9 and 10), deleting the chosen users from the set U' by the way. Then if the user's original strategy is local computing or idle device and the updating strat-

egy is another idle device, the strategies are bound to cover no overlapping channels, but one idle device can only be offloaded one task. Therefore, he/she will be added to set I' (line 15). Similarly, we will randomly select a user belonging to set I' and add him/her to set μ (lines 9 and 10) for every idle device. Finally, when all the edge nodes and idle devices are traversed, the algorithm will return the selected users set μ. As mentioned above, Algorithm 3 considers all kinds of update requests.

4.3 Convergence Analysis

According to Theorem 2, the proposed distributed game-theoretical D2D-enabled task offloading algorithm (Algorithm 1) 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 the convergence of the distributed task offloading algorithm satisfies the following equation.*

$$D < \frac{N\alpha_{\max}D_{\max}}{\triangle P_{\min}}\left(q\left(\frac{1}{c_{\text{local}}^{\min}}+\frac{\beta_{\max}P_{\max}}{\alpha_{\min}}\right)-\frac{Aj}{2}\left(\frac{N}{|\gamma|}+1\right)-\frac{q}{2\hat{c}}\left(\frac{N}{S+1}+1\right)-\left(\frac{\beta_{\min}E_{\min}+\alpha_{\max}}{\alpha_{\max}}\right)\right).$$

Proof. We consider the situation where only a user $u_i \in U$ changes the strategy from s_i to $\acute{s}_i$. When a user changes the strategy, the value of potential function will decrease correspondingly. According to (11), $Y_i < Z_i$ and $Y_i < V_i$ are obvious. Because $r_{i,j} < r_i$, we can get $Y_i < W_i$. When an equilibrium state is reached, the best case is that the capacity of edge nodes is large enough for all the users to offload their tasks to edge nodes. Then according to (11), we have the following equation:

$$\delta(\boldsymbol{s}) \geqslant Aj\frac{N}{2}\left(\frac{N}{|\gamma|}+1\right)+\frac{qN}{2\hat{c}}\left(\frac{N}{S+1}+1\right)+N\left(\frac{\beta_{\min}E_{\min}+\alpha_{\max}}{\alpha_{\max}r_{\max}}\right). \tag{12}$$

$\alpha_{\max}$ and $\beta_{\min}$ denote the maximum weight of time latency and the minimum weight of energy for all users respectively. $E_{\min}$ is the minimum energy consumption of all users' transmission tasks and $r_{\max}$ represents the maximum data transmission rate of the channels of the home edge nodes. On the contrary, the worst case is that mobile users can only choose to process locally. Therefore, we have:

$$\delta(\boldsymbol{s}) \leqslant qN\left(\frac{1}{c_{\text{local}}^{\min}}+\frac{\beta_{\max}P_{\max}}{\alpha_{\min}}\right), \tag{13}$$

where $c_{\text{local}}^{\min}$ denotes the minimum computing capacity of all local devices. Accordingly, $P_{\max}$ denotes the maximum energy consumption per bite of local computing in all users.

According to (12) and (13), when user u_i changes the strategy from s_i to $\acute{s}_i$, we have:

$$\begin{aligned}&\delta(\boldsymbol{s})-\delta(\acute{\boldsymbol{s}})\\ &<\left(qN\left(\frac{1}{c_{\text{local}}^{\min}}+\frac{\beta_{\max}P_{\max}}{\alpha_{\min}}\right)\right)-\left(Aj\frac{N}{2}\left(\frac{N}{|\gamma|}+1\right)+\frac{qN}{2\hat{c}}\left(\frac{N}{S+1}+1\right)+|U|\left(\frac{\beta_{\min}E_{\min}+\alpha_{\max}}{\alpha_{\max}r_{\max}}\right)\right).\end{aligned}$$

Then, we have the following equation:

$$D < \frac{N\alpha_{\max}D_{\max}}{\triangle P_{\min}}\left(q\left(\frac{1}{c_{\text{local}}^{\min}}+\frac{\beta_{\max}P_{\max}}{\alpha_{\min}}\right)-\frac{Aj}{2}\left(\frac{N}{|\gamma|}+1\right)-\frac{q}{2\hat{c}}\left(\frac{N}{S+1}+1\right)-\left(\frac{\beta_{\min}E_{\min}+\alpha_{\max}}{\alpha_{\max}}\right)\right).$$

□

4.4 Theoretical Analysis

We then analyze the performance of the proposed distributed D2D-enabled 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 $\boldsymbol{S}'$ be the set of strategy profiles that can achieve Nash equilibrium of the multi-user task offloading game and $\boldsymbol{s}^*$ denote the centralized optimal strategy. PoA is defined as follows:

$$PoA = \max_{\boldsymbol{s}\in\boldsymbol{S}'}\sum\nolimits_{u_i\in U} Q_i(\boldsymbol{s}) / \sum\nolimits_{u_i\in U} Q_i(\boldsymbol{s}^*).$$

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

$$1 \leqslant PoA \leqslant \frac{t_{\text{local}}^{\max}+e_{\text{local}}^{\max}}{\alpha_{\min}(t_{\text{e}}^{\min}+t_{\text{off}}^{\min})+\beta_{\min}E_{\min}},$$

where $t_{\text{local}}^{\max}$ and $e_{\text{local}}^{\max}$ mean the maximum time and the energy cost among all users' local devices, respectively, and $t_{\text{e}}^{\min}$ and $t_{\text{off}}^{\min}$ 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 the computing task at their local devices. Therefore, when our model reaches a Nash

equilibrium, the total cost of our model is always less than the total cost when all the mobile users choose local computing, which will be derived as:

$$\max_{\boldsymbol{s}\in\boldsymbol{S}'}\sum\nolimits_{u_i\in U} Q_i(\boldsymbol{s}) = N(t_{\text{local}}^{\max} + e_{\text{local}}^{\max}).$$

On the other hand, if all home nodes can afford the time and energy cost of mobile users, according to (4), the total cost will be minimum which means:

$$\sum\nolimits_{u_i\in U} Q_i(\boldsymbol{s}^*) \geqslant N(\alpha_{\min}(t_{\text{e}}^{\min} + t_{\text{off}}^{\min}) + \beta_{\min} E_{\min}).$$

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

$$\frac{t_{\text{local}}^{\max} + e_{\text{local}}^{\max}}{\alpha_{\min}(t_{\text{e}}^{\min} + t_{\text{off}}^{\min}) + \beta_{\min} E_{\min}} \geqslant PoA \geqslant 1. \qquad \square$$

5 Performance Evaluation

5.1 Datasets & Settings

The following three real-world datasets are used for the evaluation.

• *Melbourne*[29,30]. It 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*[31,32]. It contains more than 7.2 million records of accessing the Internet through 3 233 base stations from 9 481 mobile phones for six months. Each base station denotes an edge node in Shanghai, China. Fig.2(b) shows the distribution of base stations.

• *Darmstadt*[33]. It 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 devices process natural language processing applications whose input data size D_i distributes in $[100, 200]$ KB and output data size be a half of the input data[34]. The computing capacities of users' local devices and edge nodes are a Poisson distribution in the ranges of $[1, 2.5]$ GHz and $[10, 12]$ GHz, respectively, and the computing capacity of the cloud is 15 GHz. The number of idle devices accounts for 30% of the total number of mobile users. The energy consumption per CPU cycle P_i is 1. As for the offloading model, we define users' transmission rate r_i as:

$$r_i = w \log_2\left(1 + \frac{\lambda_i P_i}{\sigma^2 D_{(i,j)}^2}\right), \tag{14}$$

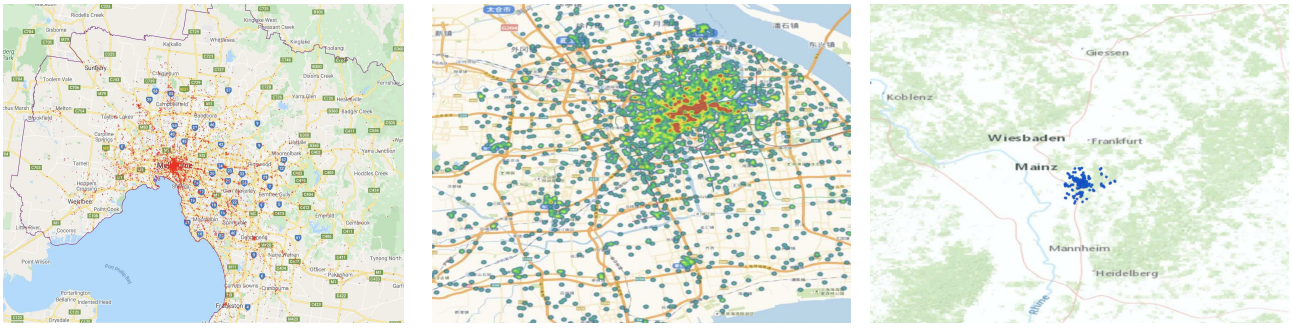
where $D_{(i,j)}$ denotes the distance between user u_i and edge node j or the distance between user u_i and idle device u_j, σ means the path loss factor which is set as 2, and λ denotes the background noise. And the channel bandwidth w is 15 MHz. The transmission rate between edge nodes is set 15 MB/s so that the delay of them will draw from a uniform distribution across $[0.1, 0.2]$. The transmission rate from the home edge node to the cloud is set 10 MB/s. Ultimately, every user's acceptable latency is determined by the local execution time.

5.2 Comparison Algorithms & Metrics

In this subsection, in order to compare the impact of different ways for users to select new offloading strategies and different ways for edge nodes to select users from users who send the updating requests on task offloading algorithms, we also utilize the following algorithms to experiment with the two task offloading models with and without D2D and cloud offloading.

• *Distributed Game-Theoretical D2D-Enable Task Offloading* (*DGTO*). It is the proposed algorithm that utilizes the SUS[15] to randomly select a user from the users who send the updating requests and allows the user to select a better offloading method to minimize his/her cost.

• *Multi-User Update Offloading* (*MUUO*). It is the proposed algorithm that utilizes the PUS algorithm to



(a) (b) (c)

Fig.2. Presentation on real-world datasets. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

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*) [35]. 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*) [35]. BUAU inspects all users and selects the user who minimizes the value of the potential function to update the strategy and allows the user to select a better offloading method to minimize his/her cost in the next decision slot.

• *Centralized Optimal Task Offloading* (*COTO*) [35]. It is a centralized optimal approach to minimizing the total cost of all users. Specifically, we conduct 100 experiments to obtain the best parameters set and use the simulated annealing algorithm for each experiment.

• *Random Task Offloading* (*RTO*) [35]. Each mobile user randomly selects an offloading method from the available policy set and RTO randomly selects a user from the users who send the update request.

5.3 Numerical Results

5.3.1 *Convergence for Nash Equilibrium*

We first verify the convergence for PUS [15] and updated PUS (Algorithm 3 in this paper), as shown in Fig.3 and Fig.4 respectively. Specifically we randomly select 20 users in each real dataset respectively and observe the dynamics of the costs in 20 decision slots without D2D and cloud offloading, as shown in Fig.3. Then we randomly select 30 users in each real dataset and there are 10 idle devices. The convergence for the proposed distributed algorithm with D2D and cloud offloading is shown in Fig.4. Obviously, after the algorithm starts, the cost of all mobile users changes with the decision update and can converge to a stable point. At this time, it reaches a Nash equilibrium state.

Then we investigate the number of decision slots for convergence with respect to the number of users as shown in Fig.5 [15] and Fig.6. No matter whether there are D2D and cloud or not, these algorithms rank as follows: MUUO < BUAU < DGTO < BRUO. The reason is that MUUO selects multiple users to update

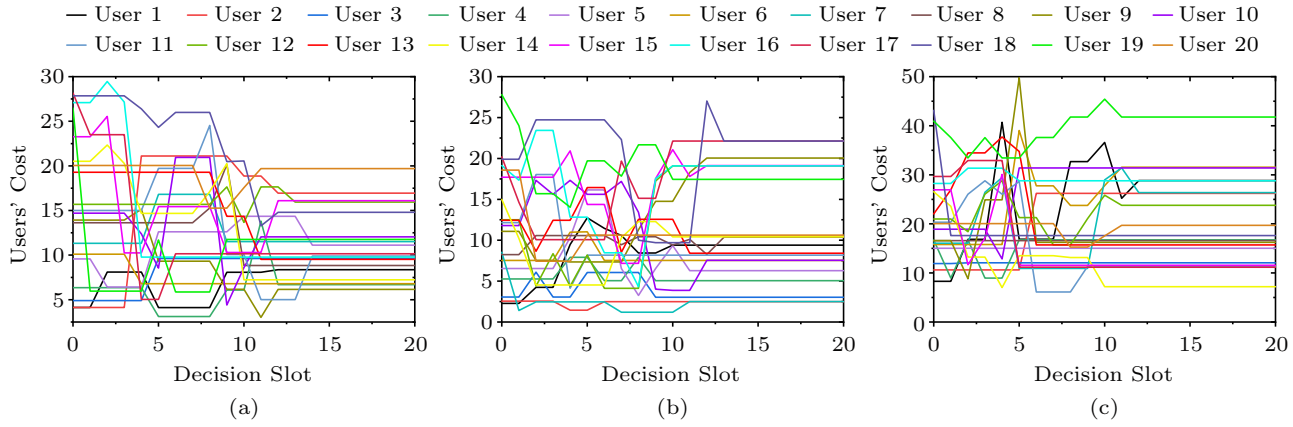


Fig.3. User cost vs decision slot without D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

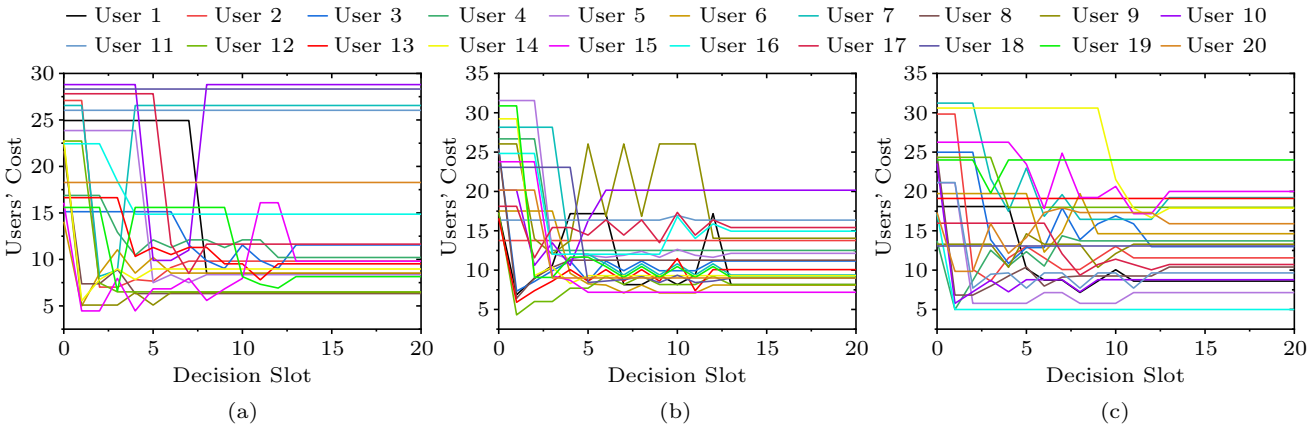


Fig.4. User cost vs decision slot with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

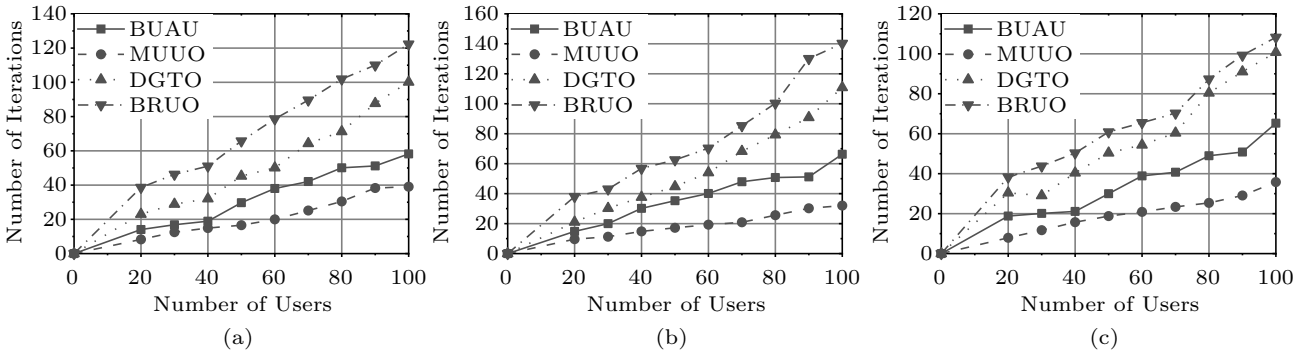


Fig.5. Decision slot vs number of users without D2D and cloud offloading [15]. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

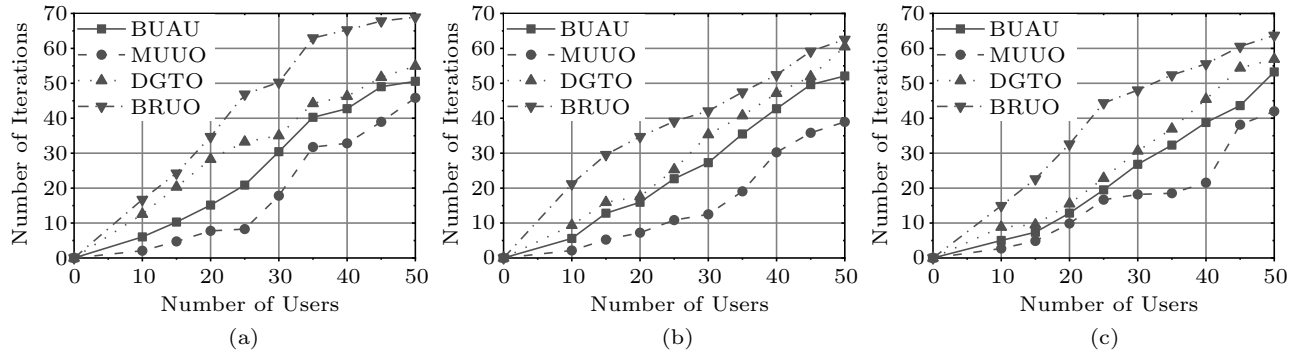


Fig.6. Decision slot vs number of users with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

their decisions in parallel, while BUAU selects only one user who minimizes the potential function in each decision slot. What is more, DGTO and BRUO randomly select a user to update the decision, but allow users to choose the best and better offloading strategies to minimize users' costs respectively. Obviously, MUUO will converge the fastest. On the contrary, BRUO will be the slowest to converge. We know that COTO and RTO do not necessarily reach Nash equilibrium in the end, and thus it is meaningless to explore COTO and RTO here.

5.3.2 *Total Cost*

As shown in Fig.7 [15] and Fig.8, we investigate the trend of the total cost with respect to the number of users. As shown in Fig.9 [15] and Fig.10, we evaluate the influence of the number of CPU cycles per data size on total cost without and with D2D and cloud offloading respectively. In Fig.11 [15] and Fig.12, we investigate the trend of total cost with respect to the channel bandwidth without and with D2D and cloud offloading respectively. We repeat the three simulations 500 times respectively which all rank as follows: COTO < BUAU < MUUO < DGTO < BRUO < RTO.

According to (5) and (14), we know that the total cost has a positive correlation with the number of users and with the number of CPU cycles per data size, and has a negative correlation with the channel bandwidth. In addition, the simulations indicate that our algorithm is acceptable compared with the optimal solution no matter whether there are cloud computing and D2D offloading or not. COTO is the centralized optimal approach to minimize the total cost of all users, and thus it gets the minimum. BUAU, MUUO and DGTO allow the user to select a better offloading method to minimize his/her cost. Compared with MUUO and DGTO, BUAU selects the user who minimizes the value of the potential function. MUUO utilizes the Parallel User Selection algorithm (Algorithm 3) but DGTO randomly selects a user from the users who send the updating requests. Although BRUO also randomly selects a user from the users who send the updating requests, BRUO allows the user to randomly select a strategy which is better than the current offloading method compared with DGTO. Therefore, the total cost value of BRUO is greater than that of DGTO. Because RTO allows users to select an offloading method from the available policy set completely randomly and also selects a user from the users who send the update request completely randomly, RTO gets the maximum total cost value. Af-

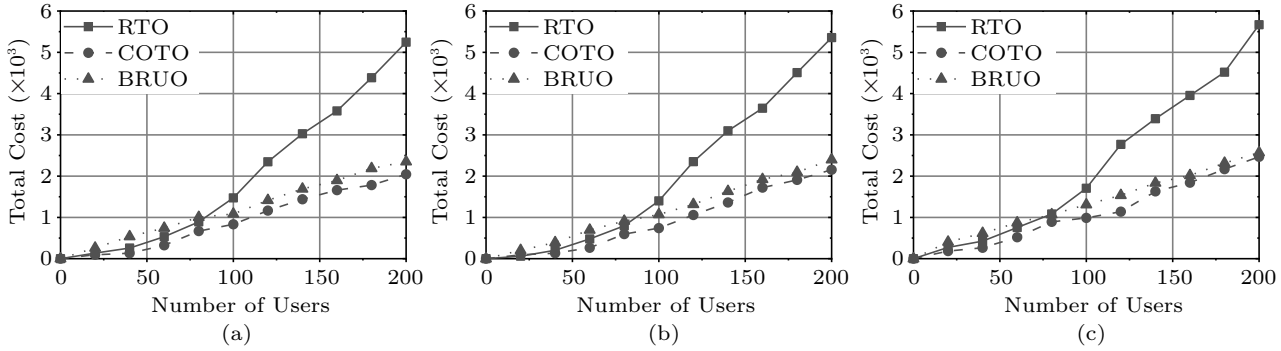


Fig.7. Total cost vs number of users without D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

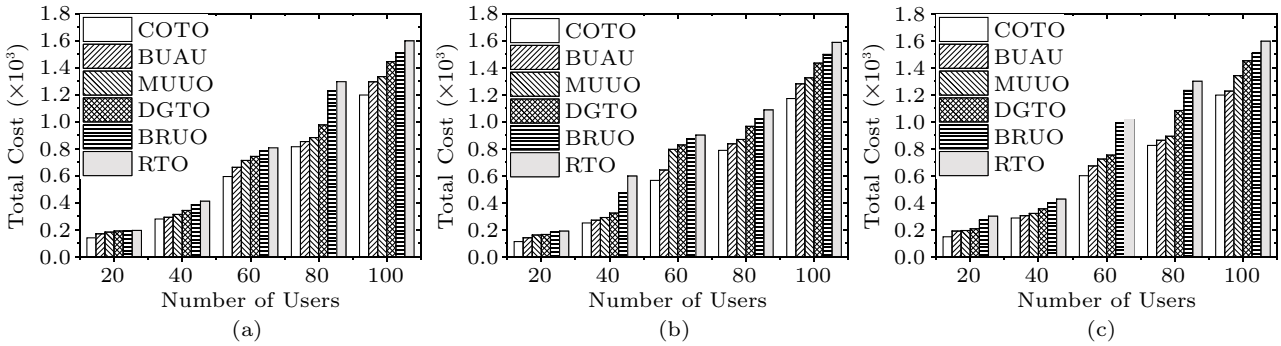


Fig.8. Total cost vs number of users with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

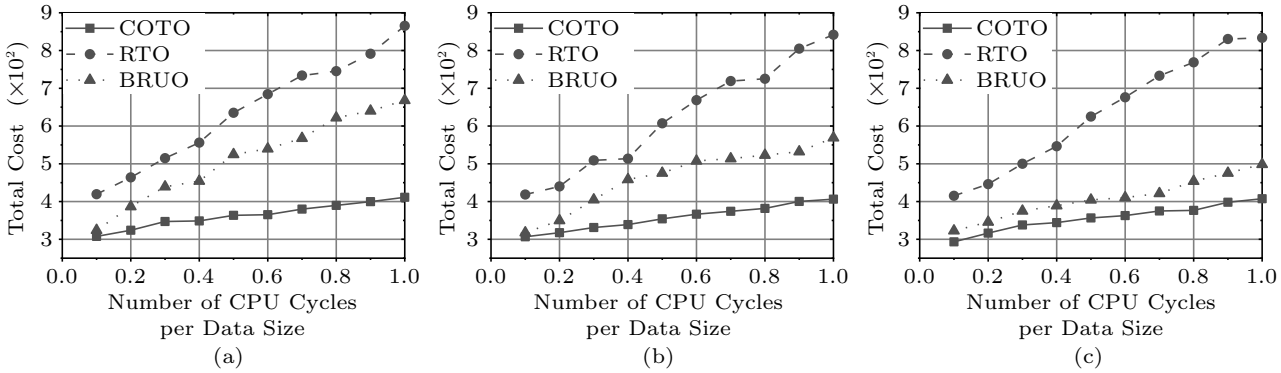


Fig.9. Total cost vs number of CPU cycles per data size without D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

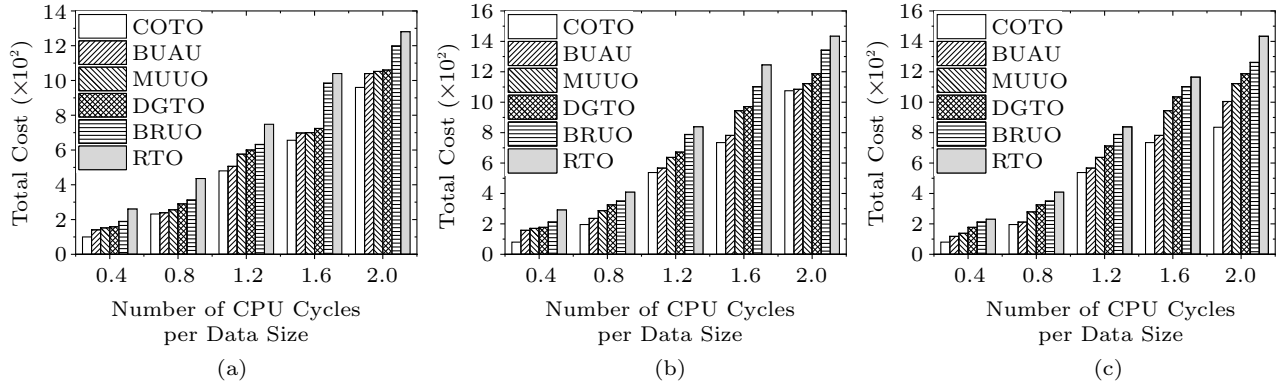


Fig.10. Total cost vs number of CPU cycles per data size with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

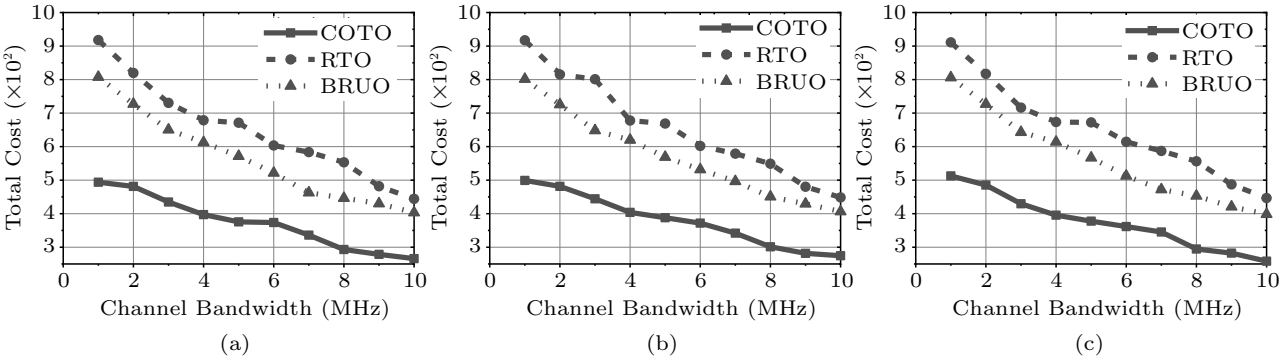


Fig.11. Total cost vs channel bandwidth without D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

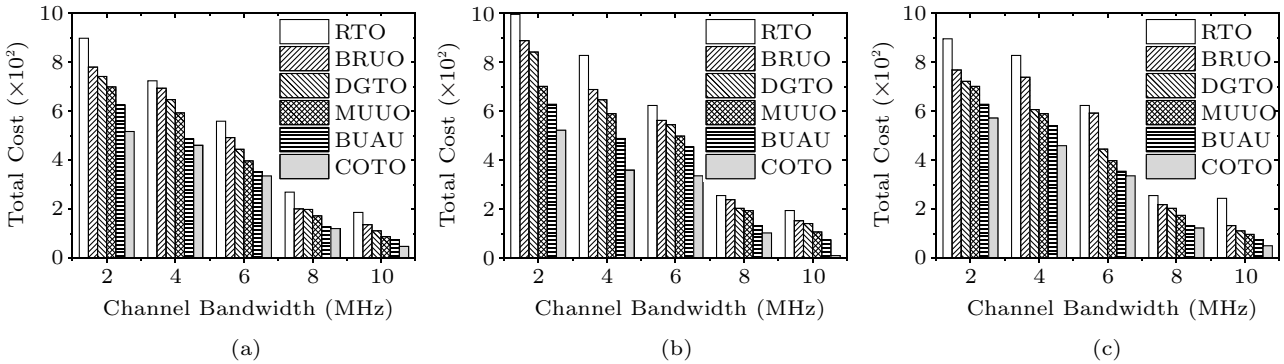


Fig.12. Total cost vs channel bandwidth with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

ter the above analysis, the rank of total cost (COTO < BUAU < MUUO < DGTO < BRUO < RTO) is reasonable.

Fig.13 shows the dynamics of Jain's fairness index with the growth of the number of users [15]. We repeat the simulations 500 times. Jain's fairness index [36] is used to measure the fairness of the user's costs, which is defined as $\frac{(\sum_{i\in U} Q_i(\boldsymbol{s}))^2}{|U|\sum_{i\in U} Q_i(\boldsymbol{s})^2}$. Most notably, Jain's fairness depends on how evenly distributed the cost of each user is. The simulation results show that the proposed DGTO achieves a higher Jain's fairness index than COTO and RTO, as DGTO can reach a Nash equilibrium of the multi-user game.

5.3.3 *Algorithm Parameters*

In Fig.14 and Fig.15 [15], we evaluate the influence of the channel bandwidth and the number of CPU cycles per data size on the tasks offloading ratio without D2D and cloud offloading. Accordingly, the influence of the channel bandwidth and the number of CPU cycles per data size on the tasks offloading ratio with D2D and cloud offloading is shown in Fig.16 and Fig.17 respectively. We repeat each simulation 500 times which is conducted among 60 mobile users. Interestingly, the offloading ratio increases with the increase of the channel bandwidth and the number of CPU cycles per data size. When the channel bandwidth gets wider, the transmission costs between edge nodes and users are lower so that more users choose to offload tasks. And when tasks are more complex, users are more likely to offload tasks to edge nodes and cloud.

In Fig.18, we show the impact of users' data size distribution on the total cost without D2D and cloud offloading. 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 the same as the rank of their users' total data size.

In Fig.19 and Fig.20, we evaluate the influence of the number of D2D links. The D2D rate denotes the ratio of the number of D2D links to the number of all users. Because the increase of the D2D rate enhances the users' chance to choose a better strategy with less cost, the total cost decreases with the increase of the D2D rate, which is easily detected. Therefore, the number of users offloading tasks to idle devices will increase with the increase of D2D rate.

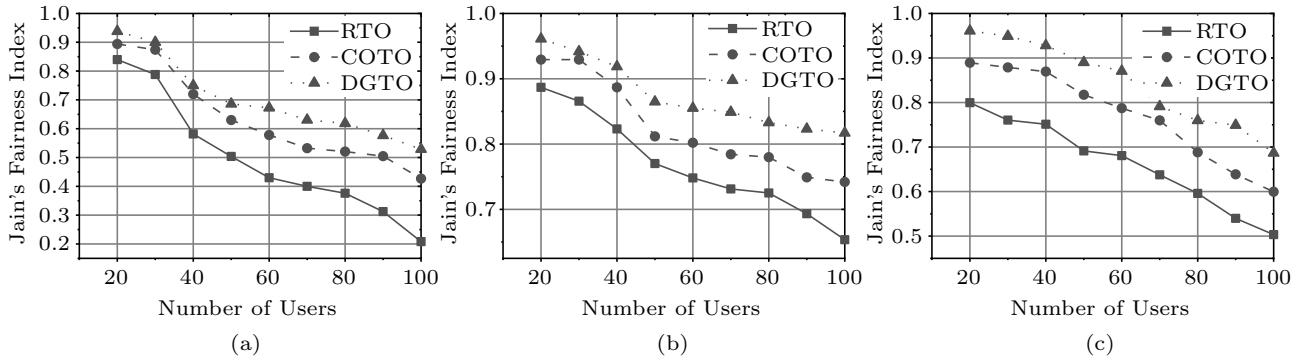


Fig.13. Jain's fairness index vs number of users. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

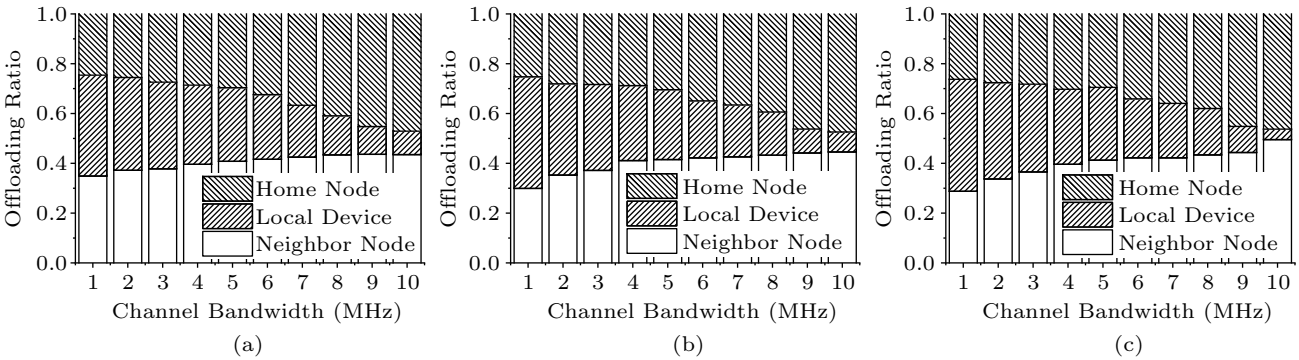


Fig.14. Offloading ratio vs channel bandwidth without D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

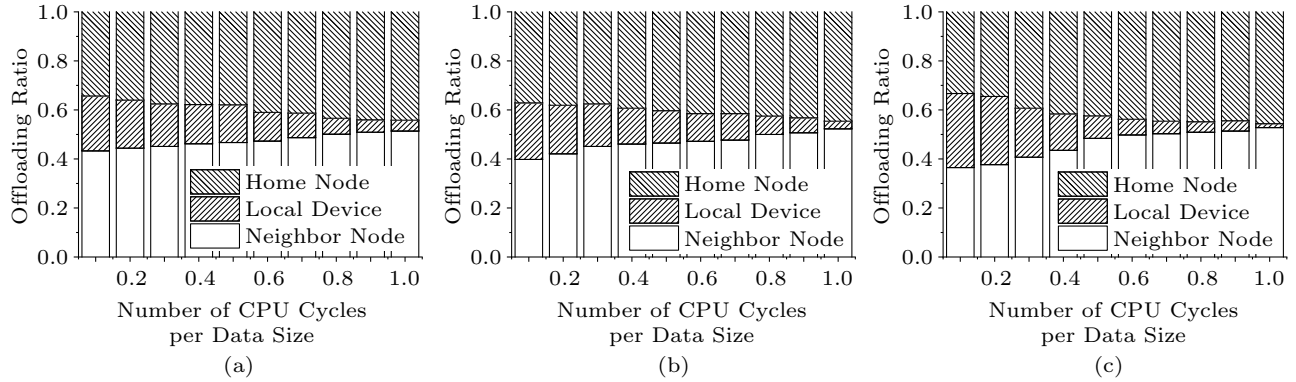


Fig.15. Offloading ratio vs number of CPU cycles per data size without D2D and cloud offloading [15]. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

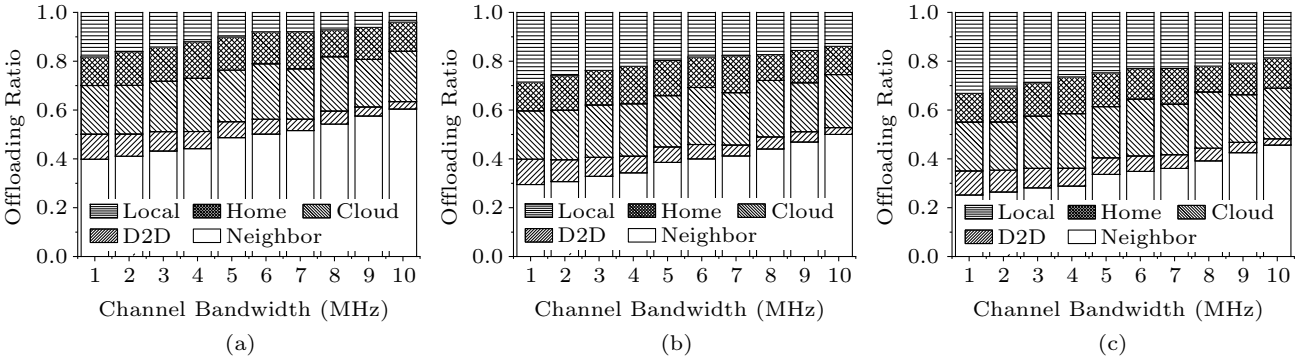


Fig.16. Offloading ratio vs channel bandwidth with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

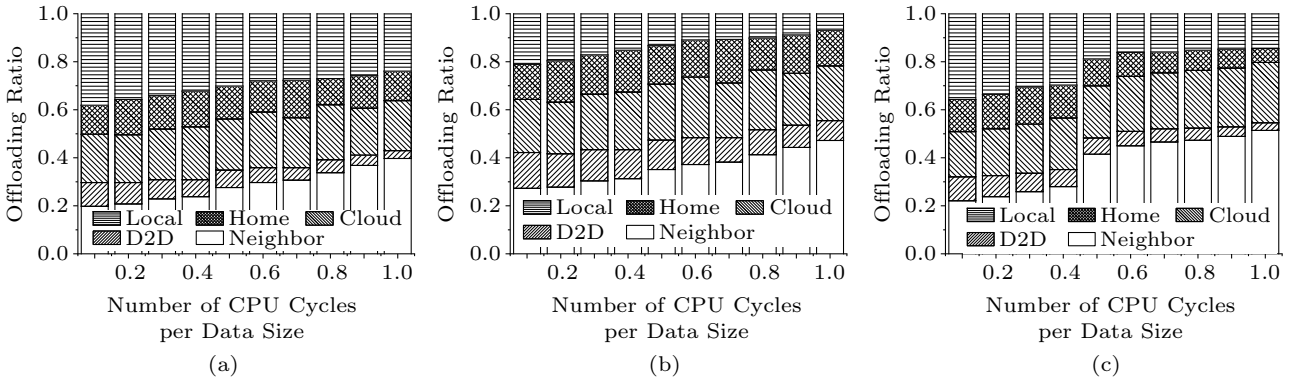


Fig.17. Offloading ratio vs number of CPU cycles per data size with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

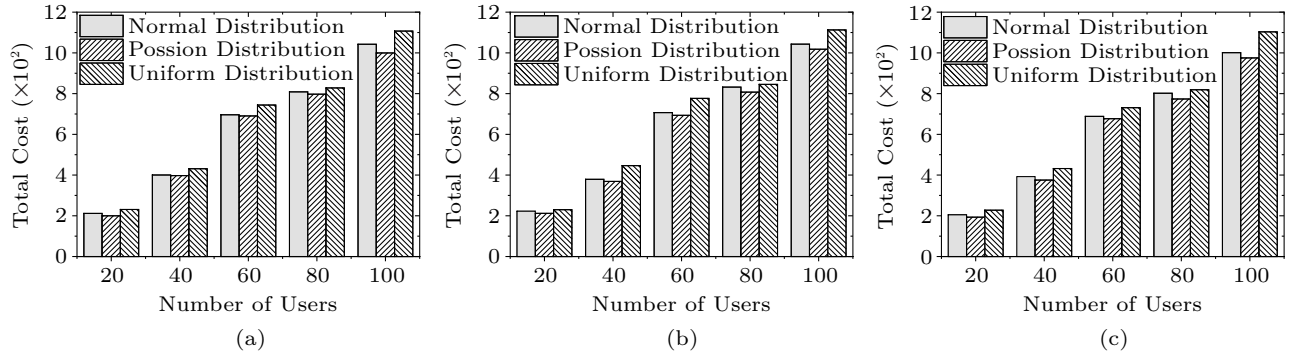


Fig.18. Total cost vs data distribution without D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

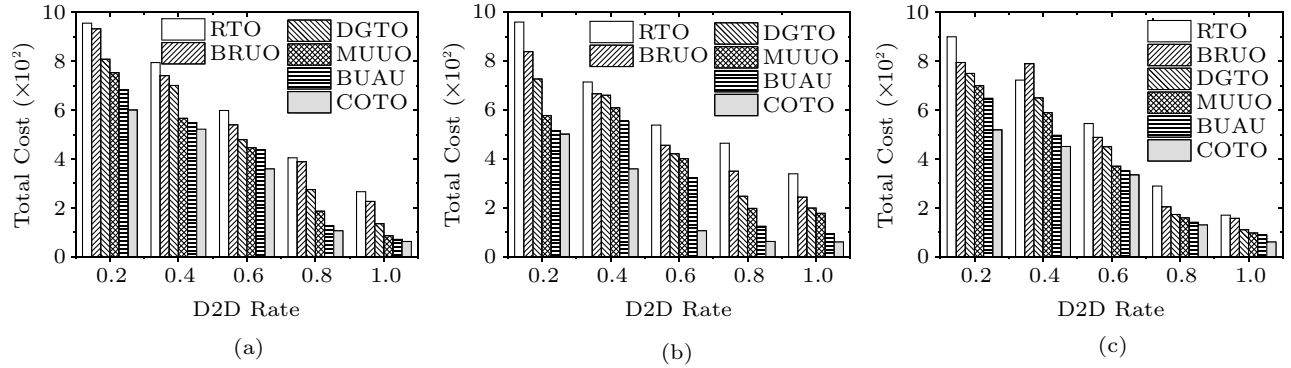


Fig.19. Total cost vs D2D rate with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

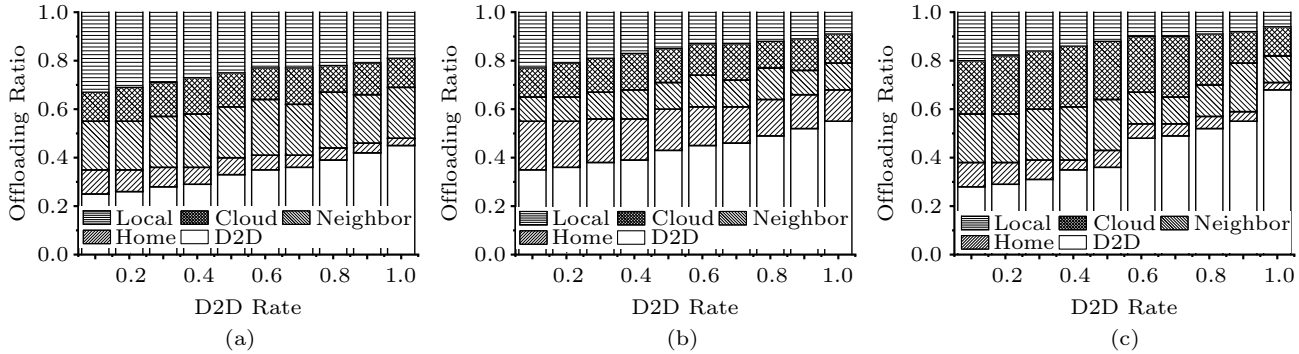


Fig.20. Offloading ratio vs D2D rate with D2D and cloud offloading. (a) Melbourne. (b) Shanghai. (c) Darmstadt.

6 Conlusions

In this paper, we extended the multi-user task offloading problem to consider the task offloading including home and neighbor edge nodes, local devices, the potential computing resources at the edge with D2D offloading and cloud computing. Simultaneously, we took into account the competition in wireless channels and users' acceptable latency. We proposed the Parallel User Selection algorithm (PUS) to accelerate the convergence of the offloading method that can achieve Nash equilibrium. Based on three real datasets, we found that among all methods in Subsection 5.2, our proposed Multi-User Update Offloading (MUUO) algorithm, which utilized PUS (Algorithm 3 in this paper) to select a set of users from the users who send the updating requests, could reach Nash equilibrium the fastest while achieving the users' total cost close to that of the optimal solution.

In the future, we will explore an incentive mechanism to ensure that all mobile users can energetically furnish their idle devices to help other users to offload their tasks. Since scalability also is a significant focus, we will also explore more potential computing resources to decrease the offloading cost in Mobile Edge Computing.

References

[1] Kim S, Visotsky E, Moorut P, Bechta K, Ghosh A, Dietrich C. Coexistence of 5G with the incumbents in the 28 and 70 GHz bands. *IEEE Journal on Selected Areas in Communications*, 2017, 35(6): 1254-1268. DOI: 10.1109/JSAC.2017.2687238.

[2] Ding C, Tao D. Trunk-branch ensemble convolutional neural networks for video-based face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017, 40(4): 1002-1014. DOI: 10.1109/TPAMI.2017.2700390.

[3] Zong Z, Hong C. On application of natural language processing in machine translation. In *Proc. the 3rd International Conference on Mechanical, Control and Computer Engineering*, Sept. 2018, pp.506-510. DOI: 10.1109/ICMCCE.2018.00112.

[4] Meng H J, Wang D C. Robust design for game-based instruction using interactive whiteboards. In *Proc. the 4th IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, Mar. 2012, pp.250-253. DOI: 10.1109/DIGITEL.2012.66.

[5] Zhang Z, Weng D, Jiang H, Liu Y, Wang Y. Inverse augmented reality: A virtual agent's perspective. arXiv:1808.03413, 2018. https://arxiv.org/abs/1808.03413-v1, Aug. 2021.

[6] Abbas N, Yan Z, Taherkordi A, Skeie T. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 2017, 5(1): 450-465. DOI: 10.1109/JIOT.2017.2750180.

[7] Satyanarayanan M. The emergence of edge computing. *Computer*, 2017, 50(1): 30-39. DOI: 10.1109/MC.2017.9.

[8] Niu Z, Wu Y, Gong J, Yang Z. Cell zooming for cost-efficient green cellular networks. *IEEE Communications Magazine*, 2010, 48(11): 74-79. DOI: 10.1109/MCOM.2010.5621970.

[9] Asadi A, Wang Q, Mancuso V. A survey on device-to-device communication in cellular networks. *IEEE Communications Surveys and Tutorials*, 2014, 16(4): 1801-1819. DOI: 10.1109/COMST.2014.2319555.

[10] Kumar K, Lu Y H. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 2010, 43(4): 51-56. DOI: 10.1109/MC.2010.98.

[11] Chen Y, Li Z, Yang B, Nai K, Li K. A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing. *Future Generation Computer Systems*, 2020, 108: 273-287. DOI: 10.1016/j.future.2020.02.045.

[12] Zhou A, Wang S, Wan S, Qi L. LMM: Latency-aware microservice mashup in mobile edge computing environment. *Neural Computing and Applications*, 2020, 32(19): 15411-15425. DOI: 10.1007/s00521-019-04693-w.

[13] Yang Y, Long C, Wu J, Peng S, Li B. D2D-enabled mobile-edge computation offloading for multi-user IoT network. *IEEE Internet of Things Journal*, 2021, 8(16): 12490-12504. DOI: 10.1109/JIOT.2021.3068722.

[14] Ding Y, Li K, Liu C, Li K. A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 2022, 33(6): 1503-1519. DOI: 10.1109/TPDS.2021.3112604.

[15] Wang E, Dong P, Xu Y, Li D, Wang L, Yang Y. Distributed game-theoretical task offloading for mobile edge computing. In *Proc. the 18th IEEE International Conference on Mobile Ad Hoc and Smart Systems* (*MASS*), Oct. 2021, pp. 216-224. DOI: 10.1109/MASS52906.2021.00037.

[16] Baron B, Spathis P, Rivano H, De Amorim M D, Viniotis Y, Ammar M H. Centrally controlled mass data offloading using vehicular traffic. *IEEE Transactions on Network and Service Management*, 2017, 14(2): 401-415. DOI: 10.1109/TNSM.2017.2672878.

[17] Jiang F, Ma R, Sun C, Gu Z. Dueling Deep Q-Network learning based computing offloading scheme for F-RAN. In *Proc. the 31st IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, Aug. 31-Sept.3, 2020. DOI: 10.1109/PIMRC48278.2020.9217355.

[18] Hong Z, Chen W, Huang H, Guo S, Zheng Z. Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments. *IEEE Transactions on Parallel and Distributed Systems*, 2019, 30(12): 2759-2774. DOI: 10.1109/TPDS.2019.2926979.

[19] Wang Y, Ge H, Feng A, Li W, Liu L, Jiang H. Computation offloading strategy based on deep reinforcement learning in cloud-assisted mobile edge computing. In *Proc. the 5th IEEE International Conference on Cloud Computing and Big Data Analytics*, Apr. 2020, pp.108-113. DOI: 10.1109/ICCCBDA49378.2020.9095689.

[20] Yu S, Langar R, Wang X. A D2D-multicast based computation offloading framework for interactive applications. In *Proc. the 2016 IEEE Global Communications Conference*, Dec. 2016. DOI: 10.1109/GLOCOM.2016.7841490.

[21] Fabiani F, Grammatico S. Multi-vehicle automated driving as a generalized mixed-integer potential game. *IEEE Transactions on Intelligent Transportation Systems*, 2019, 21(3): 1064-1073. DOI: 10.1109/TITS.2019.2901505.

[22] Liu H, Jia H, Chen J, Ge X, Li Y, Tian L, Shi J. Computing resource allocation of mobile edge computing networks based on potential game theory. arXiv:1901.00233, 2019. https://arxiv.org/abs/1901.00233, Jan. 2022.

[23] Raschellá A, Bouhafs F, Mackay M, Shi Q, Ortin J, Gallego J. R, Canales M. AP selection algorithm based on a potential game for large IEEE 802.11 WLANs. In *Proc. the 2018 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2018. DOI: 10.1109/NOMS.2018.8406147.

[24] He Q, Cui G, Zhang X, Chen F, Yang Y. A game-theoretical approach for user allocation in edge computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 2020, 31(3): 515-529. DOI: 10.1109/TPDS.2019.2938944.

[25] Wu B, Zeng J, Ge L, Tang Y, Su X. A game-theoretical approach for energy-efficient resource allocation in MEC network. In *Proc. the 2019 IEEE International Conference on Communications*, May 2019. DOI: 10.1109/ICC.2019.8761727.

[26] Zhu T, Li J, Cai Z, Li Y, Gao H. Computation scheduling for wireless powered mobile edge computing networks. In *Proc. the 2020 IEEE Conference on Computer Communications*, Jul. 2020, pp.596-605. DOI: 10.1109/INFOCOM41043.2020.9155418.

[27] Monderer D, Shapley L. Potential games. *Games and Economic Behavior*, 1996, 14(1): 124-143. DOI: 10.1006/game.1996.0044.

[28] Roughgarden T. Algorithmic game theory. *Communications of the ACM*, 2010, 53(7): 78-86. DOI: 10.1145/1785414.1785439.

[29] Xia X, Chen F, He Q, Grundy J, Abdelrazek M, Jin H. Online collaborative data caching in edge computing. *IEEE Transactions on Parallel and Distributed Systems*, 2020, 32(2): 281-294. DOI: 10.1109/TPDS.2020.3016344.

[30] Li B, He Q, Cui G, Xia X, Yang Y. READ: Robustness-oriented edge application deployment in edge computing environment. *IEEE Transactions on Services Computing*, 2022, 15(3): 1746-1759. DOI: 10.1109/TSC.2020.3015316.

[31] Wang S, Zhao Y, Xu J, Yuan J, Hsu C H. Edge server placement in mobile edge computing. *Journal of Parallel and Distributed Computing*, 2019, 127: 160-168. DOI: 10.1016/j.jpdc.2018.06.008.

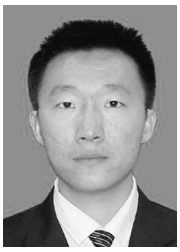
[32] Guo Y, Wang S, Zhou A, Xu J, Yuan J, Hsu C. H. User allocation-aware edge cloud placement in mobile edge computing. *Software: Practice and Experience*, 2020, 50(5): 489-502. DOI: 10.1002/spe.2685.

[33] Gedeon J, Krisztinkovics J, Meurisch C, Stein M, Wang L, Mühlhaüser M. A multi-cloudlet infrastructure for future smart cities: An empirical study. In *Proc. the 1st International Workshop on Edge Systems, Analytics and Networking*, Jun. 2018, pp.19-24. DOI: 10.1145/3213344.3213348.

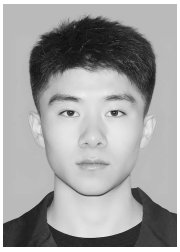
[34] Pu L, Chen X, Xu J, Fu X. D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration. *IEEE Journal on Selected Areas in Communications*, 2016, 34(12): 3887-3901. DOI: 10.1109/JSAC.2016.2624118.

[35] Wang E, Luan D, Yang Y, Wang Z, Dong P, Li D, Liu W, Wu J. Distributed game-theoretical route navigation for vehicular crowdsensing. In *Proc. the 50th International Conference on Parallel Processing*, Aug. 2021, pp.1-11. DOI: 10.1145/3472456.3472498.

[36] Jain R K, Chiu D M W, Hawe W R. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. arXiv:cs/9809099, 1998. https://arxiv.org/abs/cs/9809099, Sept. 2021.



En Wang received his B.E. degree in software engineering in 2011, his M.E. degree in computer science and technology in 2013, and his Ph.D. degree in computer science and technology in 2016, all from Jilin University, Changchun. He is currently a professor in the Department of Computer Science and Technology at Jilin University, Changchun. He was also a visiting scholar in the Department of Computer and Information Sciences at Temple University in Philadelphia. His current research focuses on the efficient utilization of network resources, scheduling and drop strategy in terms of buffer management, energy efficient communication between human-carried devices, and mobile crowdsensing.



Han Wang received his B.E. degree in software engineering from Jilin University, Changchun, in 2021. He is currently a Master student with the Department of Computer Science and Technology at Jilin University, Changchun. His current research focuses on edge computing, game theory, and multi-armed bandits.

Peng-Min Dong received his B.E. degree in software engineering from Jilin University, Changchun, in 2019. He is currently a Master student in the Department of Software at Jilin University, Changchun. His current research focuses on edge computing, cloud computing and game theory.

Yuan-Bo Xu received his B.E. degree in computer science and technology in 2012, his M.E. degree in computer science and technology in 2015, and his Ph.D. degree in computer science and technology in 2019, all from Jilin University, Changchun. He is currently a professor in the Department of Computer Science and Technology at Jilin University, Changchun. He was a visiting scholar at Rutgers, the State University of New Jersey from 2017 to 2019. His research interests include applications of data mining, recommender system, and mobile computing. He has published some research results on journals such as TKDE, TMM, and TNNLS and conferences as INFOCOM, CIKM, and ICDM.

Yong-Jian Yang received his B.E. degree in automatization from Jilin University of Technology, Changchun, in 1983, his M.E. degree in computer communication from Beijing University of Post and Telecommunications, Beijing, in 1991, and his Ph.D. degree in software and theory of computer from Jilin University, Changchun, in 2005. He is currently a professor and a Ph.D. supervisor at Jilin University. His research interests include network intelligence management, wireless mobile communication and services, and wireless mobile communication.