# An Overhead-Optimizing Task Scheduling Strategy for Ad-hoc Based Mobile Edge Computing

Li Tianze, Wu Muqing, Zhao Min, Liao Wenxing
Beijing Laboratory of Advanced Information Networks
Beijing University of Posts and Telecommunications
Beijing 100876, P. R. China
Email: {litianze,wumuqing,zhaomin,liaowenxing}@bupt.com.cn

***Abstract*—In this paper, an overhead-optimizing multi-device task scheduling strategy for ad-hoc based mobile edge computing system is proposed. This task scheduling strategy takes the opportunity consumption, time delay, energy consumption, and monetary cost into account, aiming at minimizing the overhead of each mobile device. First, a system model for ad-hoc based mobile edge computing is presented and the overhead of mobile device is analyzed. Second, the task scheduling problem is formulated as a distributed multi-device task scheduling game. Then, by constructing a potential function, the task scheduling game is proved to be a potential game, which possesses a property of finite improvement and always owns a Nash equilibrium. Next, an overhead-optimizing multi-device task scheduling algorithm is designed and the computational complexity is analyzed. Finally, simulations are conducted to evaluate the effectiveness of the proposed strategy. The results show that the proposed task scheduling strategy can effectively minimize the overhead of the mobile device and successfully complete the tasks.**

***Index Terms*—Task scheduling, Mobile edge computing, Overhead analysis, Potential game, Optimal strategy.**

## I. Introduction

Mobile devices can provide communication for us almost anywhere and anytime, which are becoming an important part of people's daily lives [1]. With the development of mobile information technology, there are some new applications emerging and attracting wide attentions, such as speech recognizer, natural language translator, image processor, augmented reality, face recognition, and interactive gaming [2]–[4]. These types of applications require a higher memory, battery energy, and computing power than that can be acquired on the resource-constrained mobile devices. As there are many limitations on communication facilities and hardware resources in mobile devices, the gap between the need of performing complex tasks and the limited resource in mobile devices is increasing everyday [5], [6].

To alleviate the resource scarcity of mobile devices, here comes the mobile edge computing. In mobile edge computing paradigm, the data processing for a resource hungry mobile application can be migrated from resource-constrained mobile devices to a powerful mobile cloud-assisted platforms [7]–[9]. Through mobile edge computing, the mobile devices can execute the tasks directly at the edge of the networks and meet these new requirements of mobile applications, such as mobility support, low latency, and location awareness.

There are two categories of mobile edge computing systems, infrastructure-based mobile edge cloud and ad-hoc based mobile edge virtual cloud. The infrastructure-based cloud is empowered by the remote servers, which can provide sufficient resource for mobile devices. Rather than rely on remote servers, the ad-hoc based virtual cloud is formed by a group of mobile devices which work cooperatively to accomplish the task [10]. Recent works show the ad-hoc based mobile edge cloud which utilizing the resources of nearby mobile devices can achieve a better overall system performance. First, the tasks can be done with a much lower cost at these nearby devices. Second, transmitting the task to these nearby mobile devices can significantly reduce communication latency, etc. [11]. In this paper, we consider the ad-hoc based mobile edge cloud.

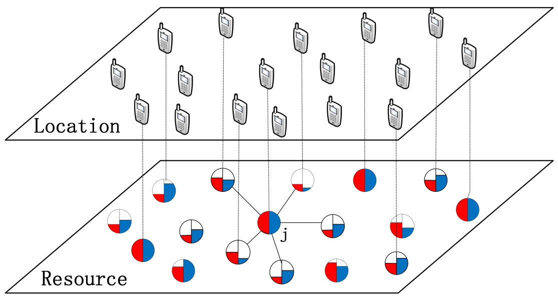

Fig. 1. Illustration of ad-hoc based mobile edge computing system. The red (blue) color portion represents available bandwidth(computation) resource. The mobile device $j$ is a resource-rich device, its neighbor devices can use the resource of it.

The architecture of the ad-hoc based mobile edge computing system is shown in Fig. 1, there are two categories of the main stakeholder: task publisher and participants. Suppose that a mobile device $i$ (task owner) wishes to execute a computation-intensive task while its computation resource is heavily occupied by other applications currently. In this case, the mobile device $i$ would publish the task to these nearby mobile devices and requests for task offloading. If the mobile device $j$ currently possesses a large amount of idle computation resource, it will reply the request. Once the mobile device $i$ receives the reply message, the task is offloaded to mobile device $j$ through the wireless link [12].

Task scheduling is the key point in a mobile edge computing systems, which enables mobile devices to support resource-intensive applications [13]. As we know, few work has been done about the task scheduling problem for the ad-hoc based mobile edge computing system. There are some challenges for the task scheduling problem in the ad-hoc based mobile edge computing systems. First, as each mobile device can move independently, the duration of one contact between two mobile devices determines the communication time between them [14]. The task scheduling strategy must take the contact duration into consideration. Second, task scheduling involves additional data transmission, the wireless access efficiency seriously impacts the performance of the mobile edge computing [15]. If too many mobile devices offload their tasks over a wireless access at the same time, they may cause serious interference with each other, which may reduce the data transfer rate and lead to a low energy efficiency [16]. Third, the computation resource is not free to use, when using the resource of the other device, the task publisher should pay for it as a reward. Consider the above analysis, without a proper task scheduling mechanism, it may not be beneficial to execute the task via mobile edge computing.

The goal of our work is to design an efficient task scheduling mechanism for mobile devices. In this paper, to address the aforementioned concerns, we propose a mathematical model about the task scheduling problem for the ad-hoc based mobile edge computing system. The model takes contact duration, opportunity consumption, energy consumption, time latency, and monetary cost into account, aiming at finding an optimal solution. The main contributions in this paper are as follows.

(1). System model and overhead analysis for ad-hoc based mobile edge computing: We first propose a system model for ad-hoc based mobile edge computing system. Then, the contact duration model, computation model, and communication model are proposed in details as they all play key roles in the process of mobile edge computing. Next, we analyze the main overhead for local computing approach (energy consumption and computational time) and mobile edge computing approach (transporting and computing latency, energy consumption, and monetary cost).

(2). Multi-device task scheduling game formulation: In the ad-hoc based mobile edge computing system, the tasks scheduling problem among mobile devices is developed as a multi-device distributed task scheduling game which takes the contact duration, communication and computing cost, and monetary cost of the mobile devices into account.

(3). Analysis of the task scheduling game: By constructing a potential function, it is proved that the multi-device task scheduling game under the wireless contention access channel is a potential game. Depending on the properties of the potential game, the multi-device task scheduling game possesses a finite improvement property, and there is always a Nash equilibrium for this game.

(4). Designing of the overhead-optimizing distributed task offloading algorithm: We devise an overhead-optimizing distributed task scheduling algorithm for ad-hoc based mobile edge computing, based on this algorithm, the mobile devices can make proper decisions about task scheduling. Then, the computational complexity of this algorithm is discussed and it is proved that this algorithm can achieve a Nash equilibrium within finite iteration times. At last, simulations are conducted to evaluate the performance of the proposed strategy; results show that our strategy has a high performance.

The rest of this paper is organized as follows. First, in Section II the related works are discussed. Then, in Section III a system model of the ad-hoc based mobile edge computing is analyzed. In Section IV, we do some analysis and formulate the task scheduling problem as an overhead-optimizing multi-device task scheduling game. Then, the game is proved to be a potential game. Next, in section V, we propose an overhead-optimizing distributed task scheduling algorithm and do some analysis about it. At last, simulations are conducted to evaluate the performance of the proposed scheme; the results are shown in Section VI. Finally, Section VII concludes this paper.

## II. Related Work

There have been extensive studies on the task scheduling mechanism for mobile edge computing. But most previous designs concentrate on infrastructure-based cloud. In [17], Zhifeng Zhong et al. introduced a Greedy Particle Swarm Optimization based algorithm to solve the task scheduling problem. In this paper, a greedy algorithm was used to quickly solve the initial particle value of a particle swarm optimization algorithm derived from a virtual machine-based cloud platform. In [18], Xiaomin Zhu et al. devised a novel agent based task scheduling strategy for mobile cloud computing to allocate real-time tasks and dynamically provision resources. In [19], Xinchen Lyu et al. proposed a heuristic task scheduling mechanism, which was semi-distributed and jointly optimizes the task scheduling strategy, computation and communication resource to maximize the utility of the system. In [20], Meng-Hsi Chen et al. considered a general multi-user mobile cloud computing system and jointly optimized the offloading decisions of all users as well as the allocation of communication resource, to minimize the cost of computation, energy, and delay for all devices. In [16], Xu Chen et al. formulated a task offloading decision making problem as a decentralized task offloading game and proposed a game theoretic scheme to achieve efficient task offloading for mobile computing system. In [21], Yuyi Mao et al. investigated a green mobile edge computing system with energy harvesting devices and developed an effective computation offloading strategy. In this paper, a low-complexity online algorithm was proposed, which jointly decided the offloading decision, the CPU-cycle frequencies for mobile execution, and the transmit power for computation offloading. In [22], M. Altamimi et al. developed an energy model for the WLAN, third generation, and fourth generation interfaces of mobile devices. The model made devices capable of accurately estimating the energy cost for task offloading. In [23], Qi Qi et al. decoupled resource control of mobile cloud from user plane, formulated the resource orchestration as a multi-objective optimal problem and proposed a particle swarm algorithm to obtain the approximate optimal solutions. Rahimi et al. in [24] proposed a novel framework to model mobile applications as a location-time workflow of tasks, which took the mobility information of mobile device into consideration and translated user mobility patterns to a mobile service usage patterns.

As far as we know, there are only a few articles that address the task scheduling issue for ad-hoc based mobile edge cloud. In [25] Li Tianze et al. designed a cooperation enhancement model based on game theory for ad-hoc based mobile computing. In this paper, the authors analyzed the profits of all these mobile devices and formulated a model based on Stackelberg game, which taken the efforts and states of all mobile devices into consideration. In [26], Lingjun Pu et al. proposed a novel task scheduling framework based on network-assisted D2D collaboration, where mobile devices can beneficially and dynamically share the computation and communication resources among each other via the assistance of the network operator.

The above task scheduling mechanisms require all the mobile devices submit their information to a central scheduler, based on which the central scheduler makes the scheduling strategy. Take a different approach, in this paper, we design a distributed task scheduling mechanism based on game theory, which takes the contact duration, communication and computing cost, and monetary cost of the mobile devices into account. In this mechanism, each mobile device performs task scheduling decisions locally, which could reduce the control and signaling overhead and improve the performance of the ad-hoc based mobile edge computing system.

## III. System Model

### A. Assumptions

- Task in the device: In the mobile edge computing system, each mobile device owns multiple tasks to be executed. Based on the task scheduling decision, a task can be executed either locally on the mobile device or on the nearby devices; each task can be executed independently.
- Consumptions: The mobile device should pay for it when using the resource of other devices. To a mobile device, the consumptions for offloading a task involves: (i) Energy consumption for transferring the data of the task. (ii) Time consumption incurred when transferring a task and executing it on the mobile edge cloud. (iii) Monetary consumption for transferring data and using the computing resources.
- Availability of the resources: The available resources in the mobile device is limited. Therefore, if there are too many mobile devices request for task offloading at the same time, some tasks will be rejected.

### B. System Model

Now, we introduce the system model of an ad-hoc based mobile edge computing system. Suppose that there are some mobile devices $s = \{1, 2, \cdots, N\}$. Each device can establish a wireless link with its neighbor devices through Wi-Fi direct. The remaining energy and total initial energy of mobile device $i$ are $e_i$ and $e_i^0$. The used computing resource and total computing resource in mobile device $i$ are $c_i$ and $c_i^0$. The state of device $i$ can be described as $s_i = (e_i, c_i, e_i^0, c_i^0)$. As each mobile device has limited energy and computing resource, it can do nothing anymore when there is no energy or computing resource, we define the function $h(e_i, c_i, e_i^0, c_i^0)$ to describe the opportunity consumption of mobile device $i$.

$$h(e_i, c_i, e_i^0, c_i^0) = \frac{e_i^0}{e_i} + \frac{c_i^0}{c_i^0 - c_i} \tag{1}$$

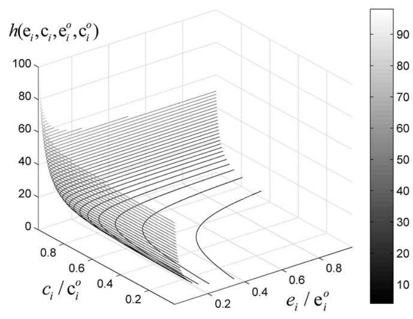

Fig. 2. Contour map of the opportunity consumption of mobile device $i$.

The contour map of mobile device $i$'s opportunity consumption is shown in figure 2. Where $e_i/e_i^0$ denotes the proportion of remaining energy and total initial energy, $c_i/c_i^0$ denotes the proportion of used computing resource and the total computing resource. The less remaining energy and computational resources, the greater the opportunity consumption of a mobile device.

Similar to many existing works [27]–[29], we consider the stream and batch based tasks of mobile devices as Google DataFlow and MapReduce programming model. A parameter tuple $(B_i, O_i, D_i)$ is adopted to characterize the delay sensitive and computation intensive task of a mobile device $i (1 \leq i \leq N)$. Where $B_i \backslash O_i$ represents the input and output data size of the task. $D_i$ represents the workload (the amount of computing resource required) of the task.

The task can be executed either locally on its original mobile device or on the other mobile device nearby through task offloading. We next discuss the task execution model.

*1) Local Execution:* Each mobile device $i$ can execute its own task locally. Suppose that the computation capability of mobile device $i$ is $F_i^l$. The execution time of the task can be given by $T_i^l = (D_i + D_i^w)/F_i^l$. Where $D_i$ is the workload of the task and $D_i^w$ is the workload of tasks which wait to be executed in mobile device $i$. The energy consumption can be given by $E_i^l = v_i D_i$. Where $v_i$ is a coefficient which represents the energy consumption for per execution unit. Then we can derive the overhead of mobile device with the local computing method in terms of time and energy consumptions.

$$Z_i^l = \alpha_l T_i^l + \beta_l E_i^l \tag{2}$$

Where $\alpha_l$,$\beta_l$ are two weighting factors which indicate the weights of time consumption and energy consumption in the decision-making process respectively. We set $0 \leq \alpha_l, \beta_l \leq 1$ and $\alpha_l + \beta_l = 1$. To meet the specific demands of mobile devices, different mobile devices are allowed to choose different weight factors. For example, in the decision-making process, if a mobile device is in a low battery state, to save more energy it would choose a larger $\beta_l$, and put more weight on the energy consumption. When a mobile device is running some time delay sensitive applications, to reduce the time delay, the device would choose a larger $\alpha_l$, and put more weight on the execution time.

*2) Offloaded Execution:* First we discuss task scheduling decision of each mobile device. Denote $\Psi_i$ as a binary indicator which is 1 if there is a task in the mobile device $i$ required for execution, and 0 else. In addition, denote $\phi_{ij}$ as a binary task offloading indicator which is 1 if the task of a mobile device $i$ is offloaded to its neighbor device $j$, and 0 else. $\phi_{ii}$ indicates whether the mobile device $i$ locally executes its task. We denote the neighbors of mobile device $i$ as a set $N_i = \{j, k, \cdots\}$, each mobile device in $N_i$ can communicate directly with mobile device $i$. About the task scheduling decision we have the following constraints:

$$\begin{gathered}\phi_{ij} = 0, j \notin N_i \\ \phi_{ij} = \{0, 1\} \\ \textstyle\sum_{j \in N_i} \phi_{ij} = \psi_i, i \in S\end{gathered} \tag{3}$$

The first constraint in formula 3 ensures that the task can be offloaded to the mobile devices which have a wireless link with $i$. The second constraint in formula 3 represents that the task can be executed locally or offloaded to the nearby devices. The third constraint represents that these tasks of all the mobile devices should be scheduled only one time.

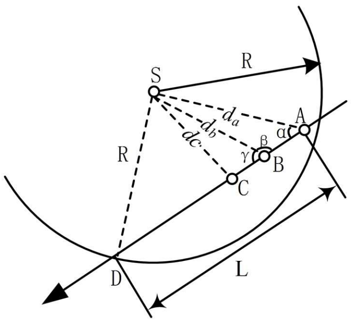

Fig. 3. Illustration of the relative movement between mobile devices $i$ and $j$.

Then, we discuss the contact duration of the mobile devices. Suppose that mobile devices $i$ and $j$ are two neighbor devices, and they both maintain a uniform linear motion in the recent time period. The relative movement speed between them is $v$, the relative distance between the mobile devices can be obtained by measuring the signal strength.

The relative movement between mobile devices $i$ and $j$ is shown in 3. Where $R$ is the maximum distance for the wireless link between mobile devices $i$ and $j$. Suppose that mobile device $i$ in point A and mobile device $j$ in point S at the initial time, the distance between i and j is $d_a$. $\Delta t$ time later, mobile device $i$ moved to point B relatively, the distance between i and j is $d_b$. $2\Delta t$ time later, mobile device $i$ moved to point C relatively, the distance between i and j is $d_c$ ($d_a, d_b, d_c$ can be obtained by measuring the signal strength). Then, the contact duration between mobile devices $i$ and $j$ is:

$$T_{ij} = \frac{d_{sa}\cos\alpha + \sqrt{R^2 - {d_{sa}}^2(1 - \cos\alpha^2)}}{v} - 2\Delta t \tag{4}$$

Where $v = \frac{\sqrt{\frac{1}{2}({d_{sa}}^2 + {d_{sc}}^2) - {d_{sb}}^2}}{\Delta t}$, and $\cos\alpha = \frac{{d_{sa}}^2 + (v \cdot \Delta t)^2 - {d_{sb}}^2}{2d_{sa} \cdot (v \cdot \Delta t)}$.

Proof: According to the cosine theorem, it can be conclude that ${d_{sb}}^2 + |AB|^2 - 2d_{sb} \cdot |AB|\cos\beta = {d_{sa}}^2$, and ${d_{sb}}^2 + |BC|^2 - 2d_{sb} \cdot |BC|\cos\gamma = {d_{sc}}^2$. As devices $i$ and $j$ maintain a uniform linear motion in the recent time period, $|AB| = |BC| = v \cdot \Delta t$. So, we can get that $v = \frac{\sqrt{\frac{1}{2}({d_{sa}}^2 + {d_{sc}}^2) - {d_{sb}}^2}}{\Delta t}$.

In the triangle SAB, $\cos\alpha = \frac{{d_{sa}}^2 + (v \cdot \Delta t)^2 - {d_{sb}}^2}{2d_{sa} \cdot (v \cdot \Delta t)}$. In the triangle SAD, ${d_{sa}}^2 + |AD|^2 - 2d_{sa} \cdot |AD|\cos\alpha = R^2$. As $|AD| = l, l \geq 0, R \geq d_{sa}$, then we can get ${d_{sa}}^2 + l^2(\cos\alpha^2 + sin\alpha^2) - 2d_{sa} \cdot l\cos\alpha = R^2$. As $l \geq 0, R \geq d_{sa}$, we can get $l = d_{sa}\cos\alpha + \sqrt{R^2 - {d_{sa}}^2(1 - \cos\alpha^2)}$. So, the contact duration between devices i and j is: $T_{ij} = d_{sa}\cos\alpha + \sqrt{R^2 - {d_{sa}}^2(1 - \cos\alpha^2)} \big/ v - 2\Delta t$.

Next, we discuss the communication and computation models. In the Wi-Fi network, the media access control protocol is carrier sense multiple access, which implements that the communication among mobile devices in the shared wireless spectrum is carried out by a packet level. Assume that the channel information will be constant over the offloading interval, then, given the task scheduling decisions of mobile device $j's$ neighbor devices $N_j$, $\phi_{N_j} = (\phi_{lj}, \phi_{kj}, \cdots\cdots \phi_{nj})$, the expected throughput between mobile device $i$ and $j$ over the wireless channel can be formulated as formula 5.

$$r_{ij} = R_j \frac{w_{ij}}{w_{ij} + \sum_{k \in N_j, k \neq i} w_{kj}} \tag{5}$$

In the above formula, $R_j$ denotes the maximum transmitting rate of mobile device $j$. $w_{ij}$ is an integer which denotes the weight of mobile device $i$ in the wireless contention channel to access to mobile device $j$. A larger weight means that the mobile device is dominant in occupying the wireless channel. If $w_{ij} = 1$ for any mobile device, it implies that wireless access channel is in an equal-sharing case.

As discussed in the subsection of assumptions, the mobile device $i$ may incur some kinds of consumptions when offloading a task, namely: time consumption, energy consumption, and monetary consumption. In the following, we will introduce these consumptions respectively.

We denote the time consumption as $T_n^c$, which is composed of the following parts: time to send the input data of a task $t_{ij}^s = B_i \big/ \sum_{j \in N_i} r_{ij}\phi_{ij}$, time to execute the task on the nearby device $t_{ij}^{ex} = D_i \big/ \sum_{j \in N_i} F_{ij}\phi_{ij}$, and time to receive the output data $t_{ij}^r$. Where $F_{ij}$ denotes the computation resource assigned to the mobile device $i$ by mobile device $j$. We denote the energy consumption as $E_n^c$, which is mainly composed of data sending energy $E_{ij}^s = P_i^s B_i \big/ \sum_{j \in N_i} r_{ij}\phi_{ij}$ and data receiving energy $E_{ij}^r$, where $P_i^s$ denotes the transmission power of device $i$.

To many applications, the input data of a task includes the input parameters, system settings, and program codes. In general, the size of the output data is much smaller than the size of input data. Similar with many researches such as [5], [30], we neglect the energy and time consumption of the mobile device to receive the output data, that is to say $t^r_{ij} = 0$, $E^r_{ij} = 0$. Therefore, the total time consumption $T^c_i$ of mobile device can be expressed as formula 6:

$$T^c_i = B_i \Big/ \sum\nolimits_{j \in N_i} r_{ij}\phi_{ij} + D_i \Big/ \sum\nolimits_{j \in N_i} F_{ij}\phi_{ij} \tag{6}$$

Then, the total energy consumption $E^c_n$ of mobile device $i$ can be expressed as as formula 7:

$$E^c_i = P^s_i B_i \Big/ \sum\nolimits_{j \in N_i} r_{ij}\phi_{ij} \tag{7}$$

We denote the monetary cost for offloading the task as $P^c_i$. Suppose that the mobile device $j$ charges $p^{in}_j$ for per byte input data, and $p^{ex}_{v_{ci}}$ for per execution unit (we neglect the monetary cost for output data). Then the monetary consumption of mobile device $i$ can be formulated as formula 8.

$$P^c_i = \sum\nolimits_{j \in N_i} p^{in}_j B_i \phi_{ij} + \sum\nolimits_{j \in N_i} p^{ex}_j D_i \phi_{ij} \tag{8}$$

According to the above analysis, we can derive the total overhead $Z^c_i$ of mobile device $i$ in terms of the time consumption, energy consumption, and monetary consumption as formula 9:

$$Z^c_i = \alpha_c T^c_i + \beta_c E^c_i + \gamma_c P^c_i \tag{9}$$

To deal with time delay, energy consumption and monetary cost in one single equation, the weighting factors $\alpha_c, \beta_c, \gamma_c$ are introduced in the formula 9. Similar to the factors in formula 2, the sum of $\alpha_c, \beta_c, \gamma_c$ is 1, and each can be set with different values depending on the demands of the mobile device.

## IV. Task Offloading Analysis

Based on the computation and communication models in section III, we can conclude that these task scheduling decisions $\phi_{ij}$ among the mobile devices $N_j$ are coupled. If there are too many mobile devices allocate their tasks to mobile device $j$ via wireless access at the same time, they may cause serious interference to each other, which would lead to a low data transfer rate. When the data transfer rate is low, it would incur a long transmission time and a high energy consumption to offload the task. In this case, perform the tasks locally without task offloading would be more beneficial.

When mobile device $i$ considers to offload its task to mobile device $j$, all task scheduling decisions of these devices in $N_j$ have influence on device $i$. We denote the task scheduling decisions of all these mobile devices in $N_j$ except $i$ as $\phi_{\mathrm{N}_j \backslash i} = (\phi_{kj}, \phi_{lj}, \cdots, \phi_{nj})$. Given the decisions $\phi_{\mathrm{N}_j \backslash i}$, mobile device $i$ would make a proper task scheduling decision based on the total overhead.

The above problem can be formulated as an overhead-optimizing multi-device task scheduling game $\Gamma = (S, \{\phi_{ij}\}_{i,j \in S}, \{Z_i\}_{i \in S})$. In the above formula $Z_i = Z^l_i$ if $\phi_{ii} = 1$, and $Z_i = Z^c_i$ if $\phi_{ii} = 0$. In this task scheduling game game, the mobile devices $i$ and $j$ in set $S$ are the players, the task scheduling decision $\phi_{ij}$ is the game strategy for player $i$, and the overhead $Z_i$ is the cost function of each mobile device in the task scheduling game, which is to be minimized by the players.

We define the decision $\phi^*_{ij}$ which could minimize the overhead of mobile device $i$ $\min\limits_{\phi_{ij \in \{0,1\}}} Z_i(\phi_{ij}, \phi_{N_j \backslash i})$ as the proper decision. The proper decision $\phi^*_{ij}$ should satisfy these following four constraints.

1) opportunity consumption constraint. We define a threshold for the opportunity consumption of a mobile device, only these mobile devices whose opportunity consumptions are less than the threshold could provide the resource, which aims at avoiding resource consumption when there are no enough resources in the mobile device. The threshold is set as $h_0 = 20$.

2) Overloading constraint. As the computation resource in the mobile device is limited, the overloading constraint prevents executing too many tasks on the same mobile device. All these tasks assigned to the mobile device $j$ should be less than the remaining computation resource $c_j$ of mobile device $j$, which can be formulated as: $\sum_{i \in N_j} D_i \phi^*_{ij} \le c_j, j \in S$.

3) Delay tolerance constraint. Suppose that mobile device $i$ considers to offload its task to mobile device $j$, firstly, the contact duration should be longer than the total time consumption: $T_{ij} < t^c_i$. Then, we define the time delay constraint $T_{\max}$ for one given task. If the task could be successfully completed, the execution time should be less than the time delay constraint.

4) Benefit constraint. Among the task scheduling decisions which satisfy these above three constraints, if the proper decision is offloading execution, the overload $Z^c_i$ should be less than the overload $Z^l_i$, $Z^c_n \le Z^l_n$. Combine formulas 2, 5, 9 we can get:

$$\frac{\sum_{k \in N_j \backslash i} w_{kj}}{w_{ij}} \le H_{\mathrm{ij}} = \frac{R_j[\alpha_l(D_i + D^w_i)/F^l_i + \beta_l v_i D_i - \gamma_c(p^{in}_j B_i + p^{res}_j D_i) - \alpha_c D_i/F_j]}{\alpha_c B_i + \beta_c P^s_i B_i} - 1 \tag{10}$$

So, under wireless contention access channel, if a mobile device $i$ chooses offloading execution to complete a task, its weight proportion among the whole mobile devices in $N_j$ satisfies formula 10.

We now introduce the Nash equilibrium of the task scheduling game $\Gamma$. A strategy profile is a Nash Equilibrium if in the equilibrium there is no mobile device can further reduce its overhead by changing its own strategy. To analyze the existence of the Nash Equilibrium in the overhead-optimizing multi-device task scheduling game. We introduce a powerful tool which named potential game.

A game is named the potential game if there is a potential function $\Phi(\phi)$ which admits that for every mobile device $i \in S$, if: $Z_i(\phi'_{ij}, \phi_{N_j \backslash i}) < Z_i(\phi_{ij}, \phi_{N_j \backslash i})$ We have: $\Phi_i(\phi'_{ij}, \phi_{N_j \backslash i}) < \Phi_i(\phi_{ij}, \phi_{N_j \backslash i})$. There are some attractive properties for a potential game, which are that it always possesses a Nash equilibrium and any asynchronous better update process must be finite and leads to a Nash equilibrium [31]. Next, we will prove that the overhead-optimizing multi-device task scheduling game is a potential game with the constructed potential function as formula 11.

$$\Phi(\phi_{ij}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k \ne i} w_{ij} w_{kj} U_{\{\phi_{ij} = \phi_{kj}\}} U_{\{\phi_{ij} > 0\}} + \sum_{i=1}^{N} w^2_{ij} H_{ij} U_{\{\phi_{ij} = 0\}} \tag{11}$$

**Algorithm 1** Initialization algorithm

```
1: Initialization:
2: for each mobile device $i$ do
3:     get the state of itself $s_i = (e_i, c_i, e_i^0, c_i^0)$
4:     compute the opportunity consumption $h_i$
5:     if $h_i > h_0$ then
6:         get the parameters about its resource
7:         get its neighbor devices set $N_i$
8:         compute the contact duration with its neighbors $T_{ij}, j \in N_i$
9:         broadcast these parameters about its resource and these contact durations with its neighbors
10:    end if
11:    for a given task $a_i$
12:    compute the execution time $T_i^l$
13:    make the initial task scheduling decisions $\phi_{ij} = 0 (j \in N_i)$
14: end for
15: End initialization
```

where $U_{\{A\}}$ is a indicator function, $U_{\{A\}} = 1$ if the event A is true, and $U_{\{A\}} = 0$ if the event A is false.

Assume that a mobile device $i$ changes its current decision $\phi_{ij}$ to a new decision $\phi'_{ij}$ and the new decision leads to a decrease to its overhead $Z_i(\phi'_{ij}, \phi_{N_j\setminus i}) < Z_i(\phi_{ij}, \phi_{N_j\setminus i})$. Then we will show that the new decision can also leads to a decrease for its potential function $\Phi_i(\phi'_{ij}, \phi_{N_j\setminus i}) < \Phi_i(\phi_{ij}, \phi_{N_j\setminus i})$. In general, the mobile device assigns a fixed computation resource for the task and charge the same $p_c^{in}$ for per byte input data. So the execution time $t_{ij}^{ex}$ and the total monetary cost $P_n^c$ for offloading the task is fixed. Then we consider two cases as following: 1. $\phi_{ij} = 0, \phi'_{ij} \neq 0$; 2. $\phi_{ij} \neq 0, \phi'_{ij} = 0$.

For case 1: Since $\phi_{ij} = 0$, $\phi'_{ij} \neq 0$ and $Z_i(\phi'_{ij}, \phi_{N_j\setminus i}) < Z_i(\phi_{ij}, \phi_{N_j\setminus i})$, according formula 10 we know that $\sum_{k\in S\setminus\{i\}} w_{kj} \Big/ w_{ij} < H_{ij}$, this implies that:

$$\begin{aligned}
&\Phi(\phi'_{ij}, \phi_{N_j\setminus i}) - \Phi(\phi_{ij}, \phi_{N_j\setminus i}) \\
&= \tfrac{1}{2}\sum_{k\neq i} w_{ij}w_{kj}U_{\{\phi'_{ij}=\phi_{ij}\}} + \tfrac{1}{2}\sum_{i\neq k} w_{kj}w_{ij}U_{\{\phi_{ij}=\phi'_{kj}\}} - w_{ij}^2 H_{ij} \\
&= w_{ij}\sum_{k\neq i} w_{kj}U_{\{\phi_{ij}=\phi'_{kj}\}} - w_{ij}^2 H_{ij} < 0
\end{aligned} \tag{12}$$

For case 2: Since $\phi_{ij} \neq 0, \phi'_{ij} = 0$, and $Z_i(\phi'_{ij}, \phi_{N_j\setminus i}) < Z_i(\phi_{ij}, \phi_{N_j\setminus i})$, we know that $H_{ij} < \sum_{k\in S\setminus i} w_{kj} \Big/ w_{ij}$, this implies that:

$$\begin{aligned}
&\Phi(\phi'_{ij}, \phi_{N_j\setminus i}) - \Phi(\phi_{ij}, \phi_{N_j\setminus i}) \\
&= w_{ij}^2 H_{ij} - \tfrac{1}{2}\sum_{k\neq i} w_{ij}w_{kj}U_{\{\phi_{kj}=\phi_{ij}\}} - \tfrac{1}{2}\sum_{i\neq k} w_{kj}w_{ij}U_{\{\phi_{ij}=\phi_{kj}\}} \\
&= w_{ij}^2 H_{ij} - w_{ij}\sum_{k\neq i} w_{kj}U_{\{\phi_{kj}=\phi_{ij}\}} < 0
\end{aligned} \tag{13}$$

So, with the potential function given in formula 11, the overhead-optimizing multi-device task scheduling game is a potential game, and there is a Nash equilibrium in this game.

## V. Overhead-optimizing Task Scheduling Mechanism

### A. Algorithm Design

In this section, we propose an overhead-optimizing task scheduling strategy under wireless contention access channel. The purpose of the task scheduling strategy is to coordinate the decisions of mobile devices to optimize the overall performance. As described above, the mobile devices monitor the states of their resources and broadcasts these parameters to the nearby mobile devices periodically. If a mobile device considers offloading its task, it receives these parameters and does some analysis to make a task scheduling decision.

As the task scheduling problem among the mobile devices is a potential game. We employ the finite improvement property of the potential game and propose an overhead-optimizing task scheduling mechanism as described in algorithms 1 and 2. In this task scheduling mechanism, each mobile device should do some analysis based on these parameters received from their neighbor devices. If a mobile device is a task offloading beneficial user, then it will compete for the decision update opportunity. If the mobile device gets the opportunity, it can update its task scheduling decision. Otherwise, the mobile device should do the analysis again and compete for next update opportunity. Details of the process are as follows:

(1)First, do the initialization work as described in algorithm 1. Each mobile device checks its state and get the parameters about itself and get the neighbor devices. When there are some tasks to be executed, the mobile device computes the execution time $T_i^l$, then sets the initial offloading decision as $\phi_{ij} = 0, (i, j \in S)$.

(2)Then, run task scheduling mechanism as described in algorithm 2. As the mobile devices who have redundant resource periodically broadcast the parameters of their resource, we define the time between two broadcasting as a **decision slot**. In one decision slot $t$, the task offloading decision $\phi_{ij}^*(t)$ which can minimize the overhead of mobile device $i$ is the **best scheduling decision**. In this decision slot $t$, the task scheduling decision $\phi_{ij}(t)$ made by the mobile device $i$ and mobile device $j$ after a negotiation process is the **final scheduling decision**.

For each mobile device $i$ in one decision slot, if $T_i^l > T_{\max}$, the mobile device selects those mobile devices $V_i$ which satisfy these four constraints described in section IV. Else, if $T_i^l < T_{\max}$, the mobile device $i$ computes its local computing overhead $Z_i^l$ and cloud computing overhead $Z_i^c$ on every mobile edge cloud nearby, then selects those appropriate mobile devices $V_i$ which satisfy $Z_i^c \leq Z_i^l$ and these four constraints described in section IV.

If $V_i = \emptyset$, the best offloading decision for mobile device $i$ in this decision slot is $\phi_{ij}^*(t) = 0$. If $V_i \neq \emptyset$, choose the mobile device $j$ which could minimize the overhead of mobile device $i$, then the best offloading decision for mobile device $i$ is $\phi_{ij}^*(t) = 1$.

(3)In decision slot $t$, the initial scheduling decision of mobile device $i$ is $\phi_{ij}(t-1)$ which is the final scheduling decision made in decision slot $t-1$. If the mobile device $i$ can reduce its overhead by changing the scheduling policy in this decision slot, we define the changing as a **better update**

**Algorithm 2** Overhead-optimizing Task Scheduling Algorithm

```
1:  loop for each decision slot $t$
2:  for each mobile device $i$ in decision slot $t$ do
3:     receive the parameters of its neighbor mobile devices
4:     get these contact durations $T_{ij}, j \in N_i$ with its neighbors
5:     if $T_i^l > T_{\max}$ then
6:        select those mobile devices $V_i$ which satisfy these
          four constraints as described in section IV.
7:     else
8:        compute $Z_i^l$ according to formula 6 and $Z_i^c$ on each
          mobile device nearby.
9:        select those mobile devices $V_i$ which satisfy $Z_i^c \leq Z_i^l$
          and these four constraints as described in section IV.
10:    end if
11:    find the best task scheduling decision $\phi_{ij}^*(t)$.
12:    if $(\phi_{ij} = 0, \phi_{ij}^* = 1)$ or $(\phi_{ij} = 1, \phi_{ij}^* = 0)$ then
13:       send REQ to the mobile device $j$ with a random
          back-off mechanism.
14:       if receive the REP message for itself then
15:          update the final decision $\phi_{ij}(t) = \phi_{ij}^*(t)$.
16:       else
17:          keep the initial decision $\phi_{ij}(t) = \phi_{ij}(t-1)$.
18:       end if
19:    else
20:       keep the initial decision $\phi_{ij}(t) = \phi_{ij}(t-1)$.
21:    end if
22: end for
    end loop until no REQ in the system for K decision slots
```

**response**, and express it as $BU_i(t)$. The better update response $BU_i(t)$ can be formulated as:

$$BU_i(t) = \begin{cases} 1, if \phi_{ij}(t-1) = 0, V_i(t) \neq \emptyset \\ 0, if \phi_{ij}(t-1) = 1, V_i(t) = \emptyset \\ \emptyset, otherwise \end{cases} \tag{14}$$

$BU_i(t) = 0, 1$ means that mobile device $i$ can reduce its overhead. Then we apply the finite improvement property of potential game and design the following rules: in each decision slot these mobile devices which can reduce their overhead compete for the decision update opportunity, and only one mobile device can update the decision at each decision slot. For mobile device $i$, if $BU_i(t) \neq \emptyset$ it will send REQ message to the broker node to contend for an opportunity to update its decision. Otherwise, if $BU_i(t) = \emptyset$, mobile device $i$ will stay in the current decision and not contend for the update opportunity.

(4) To the decision update message REQ, we apply the random back-off mechanism to avoid the collision problem of these update messages from different mobile devices. The random back-off mechanism is defined as setting a time length $t_{\max}$ for decision update message first, then each mobile device randomly generates a back-off time $t_0$ which satisfies a uniform distribution over $[0, t_{\max}]$. When the back-off time $t_0$ expires, if a mobile device has not received any REP message, it will send its decision update message REQ to the proper device $j$. When mobile device $j$ receives the REQ message from mobile device $i$, it will send REP message for $i$. Then mobile device $i$ updates its final scheduling decision $\phi_{ij}(t) = \phi_{ij}^*(t)$. For other mobile devices $k, k \neq i$, on hearing the REP message for $i$, they will not update their final scheduling decisions $\phi_{kj}(t) = \phi_{kj}(t-1), k \neq i$.

When there is no REQ for M decision slots in the system, the algorithm comes to an end. Next, we will analyze this algorithm and show that the algorithm will converge to equilibrium within finite decision slots.

### *B. Algorithm Analysis*

Now, we analyze the computational complexity of the overhead-optimizing task scheduling algorithm. In each iteration, N mobile devices will execute the operations in rows 2-20. As most operations only involve some basic arithmetical calculations, in each iteration the computational complexity of this algorithm is O($N$). Assuming that it takes C times of iteration for this algorithm to terminate. Then we can conclude that the computational complexity of the overhead-optimizing task scheduling algorithm is O(C$N$).

Suppose that $H_{max} = \max_{i \in S}\{H_{ij}\}$, $W_{\max} = \max_{i \in S}\{w_{ij}\}$, $W_{\min} = \min_{i \in S}\{w_{ij}\}$, and $w_{ij}$ is a non-negative integer for any mobile device $i \in S$, then the overhead-optimizing task scheduling algorithm will be terminated within at most $\frac{1}{2}W_{\max}^2 N^2 + W_{\max}^2 H_{\max} N \big/ W_{\min}$ iteration times.

Proof: First of all, according to formula 11, we know that:

$$\begin{aligned} 0 \leq \Phi(\phi) &\leq \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} W_{\max}^2 + \sum_{i=1}^{N} W_{\max}^2 H_{\max} \\ &= \tfrac{1}{2} W_{\max}^2 N^2 + W_{\max}^2 H_{\max} N \end{aligned} \tag{15}$$

During one iteration, suppose that a mobile device $i \in S$ changes its current decision $\phi_{ij}$ to a new decision $\phi'_{ij}$ and this new decision leads to a decrease for its total overhead function, $Z_i(\phi'_{ij}, \phi_{N_j \backslash i}) \leq Z_i(\phi_{ij}, \phi_{N_j \backslash i})$, then we will show that this new decision can lead to a decrease for the potential function by at least $W_{\min}$.

$$Z_i(\phi'_{ij}, \phi_{N_j \backslash i}) \leq Z_i(\phi_{ij}, \phi_{N_j \backslash i}) - W_{\min} \tag{16}$$

Then, we will analyze these following two cases:(1) $\phi_{ij} = 0, \phi'_{ij} \neq 0$; (2) $\phi_{ij} \neq 0, \phi'_{ij} = 0$.

For case 1: Based on formula 12, we can conclude that:

$$\begin{aligned} &\Phi(\phi'_{ij}, \phi_{N_j \backslash i}) - \Phi(\phi_{ij}, \phi_{N_j \backslash i}) \\ &= w_{ij} \left( \sum_{k \neq i} w_{kj} U_{\{\phi_{kj} = \phi'_{ij}\}} - w_{ij} H_{ij} \right) < 0 \end{aligned} \tag{17}$$

Since $w_{ij}$ are integers for any $n \in S$, we know that:

$$\sum_{k \neq i} w_{kj} U_{\{\phi_{kj} = \phi'_{ij}\}} + 1 \leq w_{ij} H_{ij} \tag{18}$$

Thus, according to formula 18, we have:

$$\begin{aligned} &\Phi(\phi'_{ij}, \phi_{N_j \backslash i}) - \Phi(\phi_{ij}, \phi_{N_j \backslash i}) \leq \\ &w_{ij} \left( \sum_{k \neq i} w_{kj} U_{\{\phi_{kj} = \phi'_{ij}\}} - (\sum_{k \neq i} w_{kj} U_{\{\phi_{kj} = \phi'_{ij}\}} + 1) \right) = -w_{ij} \\ &\Rightarrow \Phi(\phi'_{ij}, \phi_{N_j \backslash i}) \leq \Phi(\phi_{ij}, \phi_{N_j \backslash i}) - w_{ij} \\ &\Rightarrow \Phi(\phi'_{ij}, \phi_{N_j \backslash i}) \leq \Phi(\phi_{ij}, \phi_{N_j \backslash i}) - W_{\min} \end{aligned} \tag{19}$$

For case 2: As case 1, based on formula 13, we conclude that:

$$\begin{aligned}&\Phi(\phi'_{ij},\phi_{N_j\backslash i})-\Phi(\phi_{ij},\phi_{N_j\backslash i})\\&=w_{ij}\left(w_{ij}H_{ij}-\sum_{k\neq i}w_{kj}U_{\{\phi_{kj}=\phi_{ij}\}}\right)<0\end{aligned}\tag{20}$$

With the similar augment in case 1, we conclude that:

$$w_{ij}H_{ij}+1\leq\sum_{k\neq i}w_{kj}U_{\{\phi_{kj}=\phi_{ij}\}}\tag{21}$$

Thus, according to formula 21, we have:

$$\begin{aligned}&\Phi(\phi'_{ij},\phi_{N_j\backslash i})-\Phi(\phi_{ij},\phi_{N_j\backslash i})\leq\\&w_{ij}\left(w_{ij}H_{ij}-(w_{ij}H_{ij}+1)\right)=-w_{kv_c}\\&\Rightarrow\Phi(\phi'_{ij},\phi_{N_j\backslash i})\leq\Phi(\phi_{ij},\phi_{N_j\backslash i})-W_{\min}\end{aligned}\tag{22}$$

Thus, according to formulas 15 and 16, we conclude that the algorithm can be terminated within at most $\frac{1}{2}W_{\max}^2N^2+W_{\max}^2H_{\max}N/W_{\min}$ iteration times.

## VI. Experimental Evaluation

### A. Simulation setup

To investigate the performance of the proposed overhead-optimizing task scheduling mechanism, we choose to conduct simulations. We design the ad-hoc based mobile edge computing scenario as that 50 mobile devices are randomly distributed within an area of 1000m * 1000m, the mobility model of each mobile device is the random way-point model with the speed of 10m/s. Each mobile device can connect to the nearby devices within 200 meters via a Wi-Fi network.

In this simulation, the parameters about mobile devices are set as follows: fifteen pieces of computation tasks come randomly to the mobile devices in the network per second, the tasks $a_i=(B_i,O_i,D_i)$ are set as $B_i=5MB, O_i=0.1MB, D_i\sim U(30,60)MI$, the time delay constraint $T_{\max}\sim U(5,10)s$.

The parameters of the mobile devices are set as: $F_n^l=5MI/s$, $(\alpha_l,\beta_l)=(0.5,0.5)$, $v_n=0.2$, $(\alpha_c,\beta_c,\gamma_c)=(0.4,0.4,0.2)$, $P_n^s=1$. Their total initial energy and total computation resource were set as $(e_i^0,c_i^0)=(10^5J,\ 300MI)$. At some point $t_0$, their remaining energy and the free computation resource were uniformly distributed $e_i(t_0)\sim U(0,10^5)$, $c_i(t_0)\sim U(0,300)$. The states of mobile devices evolve as $e_i(t)=e_i(t-1)-v_i\sum_j D_jU_{\{\phi_{ji}=1,t-1\leq t_{a_j}\leq t\}}$ and $\mathrm{c}_i(t)=\mathrm{c}_i(0)-\sum_j D_jU_{\{\phi_{ji}=1,t_{a_j}\leq t\leq t_{a_j}+D_j/F_{ji}\}}$, where $t_{a_j}$ represents the time when task $a_j$ comes, $U_{\{\phi_{ji}=1,t-1\leq t_{a_j}\leq t\}}$ means the task comes in the time slot $[t-1,t]$ and executed by mobile device $i$, $U_{\{\phi_{ji}=1,t_{a_j}\leq t\leq t_{a_j}+D_j/F_{ji}\}}$ means the task executed by mobile device $i$ and has not been completed at time $t$.

The parameters of the mobile edge cloud are set as: $F_{ij}=10MI/s$, $F_i^l=5MI/s$, $p_i^{in}=1MB^{-1}$, $p_i^{res}\sim U(0.2,0.3)$. The parameters of the Wi-Fi wireless channel are set as: $R_i=10MB/s$, $w_{ij}=1$.

We evaluate the capability of our proposed task scheduling algorithm by comparing its performance with the following three schemes:

(1).Local Computing without task scheduling

The first scenario is that each mobile device chooses to execute its task by itself. In this case, the task scheduling decision of each mobile device is $\phi_{ii}=1$, the total overhead in this scenario is given by formula 2. This scenario provides a baseline for network performance across all mobile devices.

(2).Task scheduling with randomly selected device

In the second scenario, each mobile device computes the execution time of the task first. If the execution time is greater than the time delay constraint $T_{\max}^i$, the mobile device chooses to offload its tasks to a randomly selected neighbor device without considering the impact to other mobile devices. In this case, the total overhead of a mobile device can be expressed by formula 13. This scenario provides a baseline for the performance of task scheduling schemes.

(3).Cross entropy based optimization scheme

In the third scenario, we use the centralized cross-entropy method to solve the task scheduling problem, which is a stochastic search technique and has been proved to be effective in finding the approximate optimal solution of the optimization problem [32]. In this case, the task scheduling strategy of each mobile device can be obtained as a result of the optimization process which aims at maximizing the profit of the resource provider.

In this simulation, we first evaluate the system performance of each scheme under the given scenario, then we analyze the two key factors: communication data size, and computation size, which influence the performance of the task scheduling scheme in the ad-hoc based mobile edge computing system.

### B. Simulation result

*1) System performance:* First of all, we study the unsuccessfully completed task. The so called unsuccessfully completed task is the task whose execution time is longer than the time constraint of itself. We define the number of unsuccessfully completed tasks as $UN$, which can be calculated by formula 23. As mentioned in section IV, $U_{\{A\}}$ is a indicator function, $U_{\{A\}}=1$ if the event A is true, and $U_{\{A\}}=0$ if the event A is false.

$$UN=\sum_{\mathrm{i}\in S}U_{\{T_i^l>T_{\max}^i,\phi_{ii}=1\}}+\sum_{i\in S}U_{\{T_i^c>T_{\max}^i,\phi_{ii}=0\}}\tag{23}$$

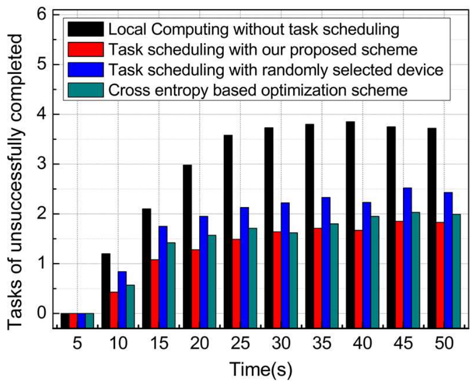

Fig. 4. Number of tasks unsuccessfully completed.

The number of unsuccessfully completed tasks $UN$ in all four schemes are depicted in Fig. 4. As the time delay constraint $T_{\max} > 5$, there is no unsuccessfully completed task in the first five seconds. In this figure, it is obviously that the task scheduling schemes all have a better performance in $UN$ than the local computing scheme. In the local computing scheme, if there are too many tasks to be executed, at some point some tasks should wait to be executed and could not be completed in time due to the lack of computing resources. Those mobile devices in task scheduling schemes can use the resources of the neighbor resource-rich device, if there are too many tasks, they can offload the tasks to their neighbors, and complete the tasks quickly.

As the proposed task scheduling scheme aims at reducing the overhead of mobile devices and takes the contact duration, wireless accessing coordinating, and computational resource allocating into consideration, the mobile devices in our proposed scheme can have a stable high-speed wireless channel to transmit the data of the task, the mobile devices in our proposed scheme can complete the task in time and has a better performance in $UN$.

Then, we research the execution time of one task in each scheme. We define the execution time of a task as $ET_i$, then the execution time of a task can be expressed as formula 24.

$$ET_i = T_i^c \cdot U_{\{\phi_{ii}=0\}} + T_i^l \cdot U_{\{\phi_{ii}=1\}} \tag{24}$$

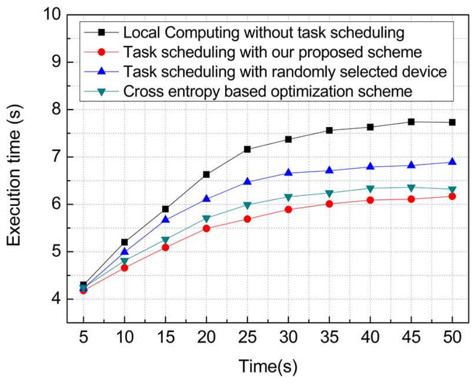

Fig. 5. The average execution time per task.

The average execution time of a task in the system over time is depicted in Fig. 5. Through this figure, we can see that the local computing scheme has a much longer execution time than that of task offloading schemes. In the task offloading schemes, the scheme with a randomly selected mobile device has the longest execution time, and our proposed scheme has the shortest execution time. In a local computing scheme, the execution time of a task is mainly the executing time. However, in the mobile edge cloud computing schemes, the time consumption of a task mainly includes the data transmitting time, and the task executing time in the neighbor resource-rich device. In general, the task executing time in the resource-rich device is smaller than the local executing time. As our task scheduling scheme takes the contact duration into consideration and can optimize the wireless transmission resources and computing resources at the same time, it has a relatively shorter execution time for one task than other task offloading schemes.

Third, we study the energy consumption of the task owner and define it as $EC_i$. If a task is executed by the mobile device locally, the energy consumption is $E_i^l$, if the task is executed by the neighbor resource-rich mobile device, the energy consumption is $E_i^c$, so the energy consumption of the task owner can be expressed as formula 25.

$$EC_i = E_i^c \cdot U_{\{\phi_{ii}=0\}} + E_i^l \cdot U_{\{\phi_{ii}=1\}} \tag{25}$$

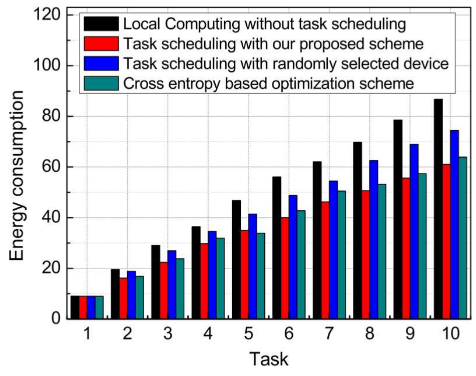

Fig. 6. The energy consumption of task owner.

The energy consumption of task owner in each scheme is depicted in Fig. 6. In a local computing scheme without task offloading, the energy consumption to complete the tasks is mainly the task executing consumption. However, in the mobile edge cloud computing schemes, the energy consumption to complete the tasks is mainly the data transmitting consumption. In general, the data transmitting consumption is lower than the task executing consumption, so the mobile devices in the local computing scheme have the largest energy consumption to complete tasks.

Then through Fig. 6, we can see that the proposed scheme and the cross-entropy based task scheduling scheme have nearly the same energy consumption with each other, and all lower then the energy consumption of the scheme which with a randomly selected mobile cloud. The reason is that in the scheme of randomly selected cloud, the mobile devices transmit their tasks to the neighbor mobile device immediately if they can get a benefit through it, and ignore the interference to other mobile devices. This may lead to a lower data transmission rate, a longer data transmission time and a higher energy consumption.

Then, we study the overhead of a mobile device over time. We define the overhead of a mobile device $i$ at time $t_1$ as $O_i(t_1)$, which represents the whole overhead of the tasks which comes from time $(t_1 - \Delta t)$ to $(t_1 + \Delta t)$. Assume that a task $a_i(t_0)$ comes to the mobile device $i$ at time $t_0$, as discussed in section II the overhead of task $a_i(t_0)$ for mobile device is $Z_i^l \cdot U_{\{\phi_{ii}=1\}} + Z_i^c \cdot U_{\{\phi_{ii}=0\}}$. So to mobile device $i$, the overhead at time $t_1$ can be expressed as formula 26.

$$O_i(t_1) = \sum_{a_i(t_0)} \left( Z_i^l \cdot U_{\{\phi_{ii}=1\}} + Z_i^c \cdot U_{\{\phi_{ii}=0\}} \right) U_{\{(t_1-\Delta t)<t_0<(t_1+\Delta t)\}} \tag{26}$$

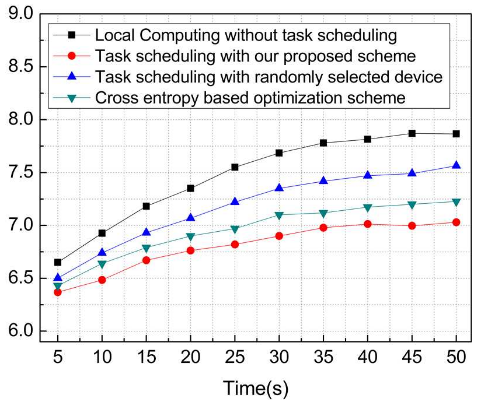

Fig. 7. The average overhead for mobile device.

Fig. 7 depicts the average overhead of the mobile devices in each scheme. In this figure, we can see that the local computing mechanism has the heaviest overhead, the overhead in our task scheduling scheme is a slightly lighter than the overhead in the cross entropy based scheme, and both the above two schemes have a lighter overhead than that in the scheme of the randomly selected device. The reason is that mobile devices in local computing scheme can not offload the tasks and have to execute the tasks themselves. However, the mobile devices in task offloading schemes can choose to execute the tasks locally or execute the tasks through the resource-rich device. In general, executing the task through the resource-rich device has a low overhead. In the scheme of the randomly selected device, the mobile device regardless of the impact to others, which to a certain extent, increase the overhead of themselves. The cross entropy based scheme take the profit of the resource provider into first consideration, and the overhead of the mobile device is a little higher than our proposed task scheduling mechanism.

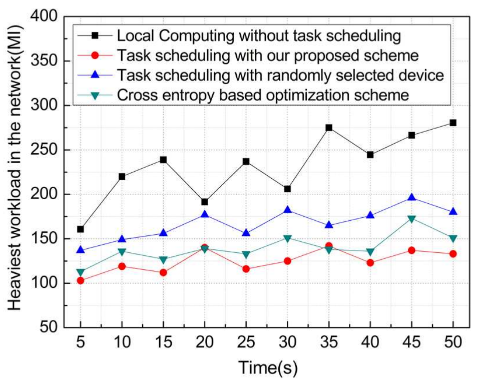

Fig. 8. The heaviest workload in the system.

Next, we study the heaviest workload of each mobile device in the system. The workload of the mobile device $i$ is the total workload of these tasks which had come to $i$ but have not been executed. At time $t$, we define the workload of the mobile device $i$ as $WL_i(t)$. Then the heaviest workload in the system can be expressed as $\max\{WL_i(t)\}, i = 1, 2, 3...N$.

The heaviest workload in the system over time is shown in Fig. 8. In the beginning, the number of tasks is not very large and there are relatively adequate computing resources in the system compared with the amount of tasks, so the heaviest workload in the system is low. With time going on, there are lots of tasks come to the system, and the heaviest workload in the system increases relatively. In the overall perspective, the scheme without task offloading has the heaviest workload, the heaviest workload in the task offloading schemes are a little lower, as the mobile devices in the task offloading schemes can allocate their tasks to the other mobile devices. Our proposed task scheduling scheme takes the overhead of each mobile device into consideration and performs well in the characteristic of heaviest workload.

*2) Influence factors analysis:* As communication data size and computation size are the main influence factors on the performance of a computation task. In this section, we evaluate the performance of each scheme with different communication data size and computation size.

(1).Communication data size

When a task is executed locally on a mobile device, there is no data transfer. However, during mobile edge computing, task offloading incurs additional communication cost as the mobile device has to transfer the task and its related data. To evaluate the impact of communication data size on the task scheduling schemes, we implement the simulations with different data size of task $B_i$, the computation size is set as a fixed value 50MI, the others parameters are set the same as the parameters in section 6.1.

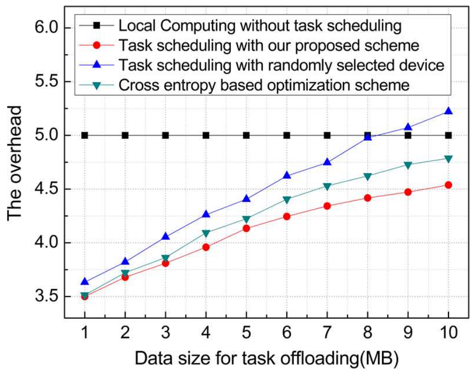

Fig. 9. Average overhead for mobile device.

The average overhead for the mobile device is shown in Fig. 9. The number of unsuccessfully completed tasks in the network is shown in Fig. 10. Through these two figures we observe that in the task offloading schemes, the average overhead of mobile devices and the number of unsuccessfully completed tasks increase as the data size $B_i$ increases. However, in the local computing scheme, the average overhead of mobile devices

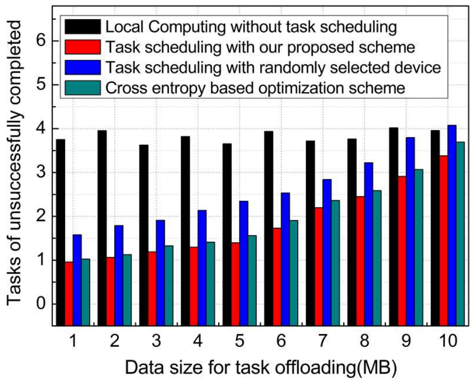

Fig. 10. Tasks of unsuccessfully completed.

remain constant, and the number of unsuccessfully completed tasks almost has no change as the data size $B_i$ increases. The reason is that a larger data size leads to additional transmit consumption for task offloading via wireless communication, which does not happen when the tasks are executed locally on the mobile devices. That is to say, the data size of the task has no influence on the mobile devices in local computing scheme but has a significant impact on the mobile devices which in task offloading schemes.

Moreover, through these two figures, we observe that the average overhead of mobile devices increases slowly when the data size is small. However, the number of unsuccessfully completed tasks increases quickly when the data size is large. The reason is that when the data size is large, more mobile devices choose to complete the tasks locally, so as to avoid the heavy cost of task offloading via wireless access. As more and more tasks are executed locally on the mobile devices, and the limitation of computation resource in the mobile devices, the number of unsuccessfully completed tasks increases quickly.

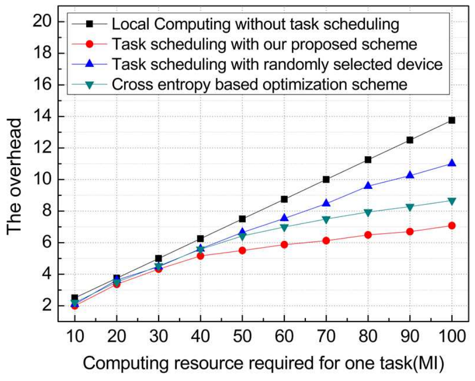

Fig. 11. Average overhead for mobile device.

(2).Computation size

The computation size of the task is the main factor which influences the computing resource consumption, energy consumption and executing time of the mobile device. To investigate the impact of computation size on the mobile edge computing system, we then implement simulations with a different number of required computing resource $D_i$ for completing a task. The communication data size for task offloading $B_i$ is set as a fixed value 5MB, the others parameters are set the same as the parameters in section 6.1.

The average overhead for the mobile device is shown in Fig. 11. Through this figure, we observe that in all these four schemes the average overhead of the mobile devices increases as the computation size of the task $D_i$ increases. But, the average overhead of mobile devices in the task offloading schemes increases slower than that in the local computing scheme. The reason is that a larger computation size leads to a large resource consumption and time consumption for the mobile device to execute the task, these mobile devices in local computing scheme have to complete the task by themselves, so the overhead increases immediately as the computation size of the task $D_i$ increases. In general, the mobile devices in task offloading schemes could allocate the tasks to mobile edge cloud, but, if there are too many devices offload their tasks at the same time, this may cause serious interference to each other. At that time, part of the tasks have to be executed locally, and the average overhead of mobile devices also increases as the computation size of task increases.

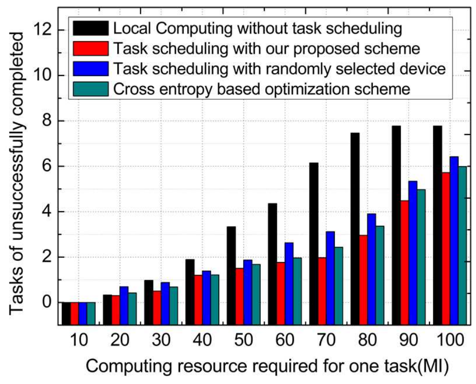

Fig. 12. Tasks of unsuccessfully completed.

The number of unsuccessfully completed tasks in the network is shown in Fig. 12. In this figure, we can see that in all these four schemes the number of unsuccessfully completed tasks increases as the computation size $D_i$ increases. However, the unsuccessfully completed tasks by task offloading schemes increase much slower than that of local computing scheme. The reason is that the mobile devices in local computing scheme have a limited resource, as the computation size increases, some of the tasks can not be dealt in time. While in the task offloading schemes, as the computation size of task $D_i$ increases, more mobile devices choose to utilize the resource of the resource-rich device which can mitigate the heavy overhead of local computing. But to avoid the serious interference with each other, part of the tasks have to be executed in the mobile devices locally. If computation size of the task is too large, the un-offloaded task could not be completed in time.

## VII. CONCLUSION

In this paper, we analyzed the task scheduling problem for ad-hoc based mobile edge computing and proposed an overhead-optimizing task scheduling mechanism. The task scheduling mechanism took the opportunity consumption, energy consumption, time delay, and monetary cost into account, aiming at minimizing the overhead for the mobile devices. We first presented the communication model and computing model, then analyzed the overhead in both task offloading and local computing patterns. Next, we formulated the task scheduling problem as a multi-device task scheduling game and proved that the game was a potential game. Then an overhead-optimizing task scheduling algorithm was devised, and the computational complexity was analyzed. At last, simulations were conducted to evaluate the effectiveness of the proposed scheme, and the performance was compared with other three schemes. The results showed that the proposed task scheduling scheme could effectively minimize the overhead of the mobile devices and successfully complete the tasks.

For the future work, we are considering extending the task scheduling model with task priority, in which case, the tasks with high priority should be performed first.

## Acknowledgments

This work is supported by Beijing Laboratory of Advanced Information Networks.

## References

[1] M. Satyanarayanan, "Mobile computing: The next decade," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 15, no. 2, pp. 2–10, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2016598.2016600

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

[3] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Computers and Communications (ISCC), 2012 IEEE Symposium on*, July 2012, pp. 000 059–000 066.

[4] D. AbdElminaam, H. Abdul Kader, M. Hadhoud, and S. El-Sayed, "Elastic framework for augmenting the performance of mobile applications using cloud computing," in *Computer Engineering Conference (ICENCO), 2013 9th International*, Dec 2013, pp. 134–141.

[5] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11036-012-0368-0

[6] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *Wireless Communications, IEEE*, vol. 20, no. 3, pp. 14–22, June 2013.

[7] A. Khan, M. Othman, S. Madani, and S. Khan, "A survey of mobile cloud computing application models," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 393–413, First 2014.

[8] ——, "A survey of mobile cloud computing application models," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 393–413, First 2014.

[9] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen, "Enabling adaptive high-frame-rate video streaming in mobile cloud gaming applications," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 25, no. 12, pp. 1988–2001, Dec 2015.

[10] N. Fernando, S. Loke, and W. Rahayu, "Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing," in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, Dec 2011, pp. 281–286.

[11] S. Al Noor, R. Hasan, and M. Haque, "Cellcloud: A novel cost effective formation of mobile cloud based on bidding incentives," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, June 2014, pp. 200–207.

[12] ——, "Cellcloud: A novel cost effective formation of mobile cloud based on bidding incentives," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, June 2014, pp. 200–207.

[13] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 945–953.

[14] Z. Li, Y. Liu, H. Zhu, and L. Sun, "Coff: Contact-duration-aware cellular traffic offloading over delay tolerant networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 5257–5268, Nov 2015.

[15] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 2716–2720.

[16] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, April 2015.

[17] Z. Zhong, K. Chen, X. Zhai, and S. Zhou, "Virtual machine-based task scheduling algorithm in a cloud computing environment," *Tsinghua Science and Technology*, vol. 21, no. 6, pp. 660–667, Dec 2016.

[18] X. Zhu, C. Chen, L. T. Yang, and Y. Xiang, "Angel: Agent-based scheduling for real-time tasks in virtualized clouds," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3389–3403, Dec 2015.

[19] X. Lyu, H. Tian, P. Zhang, and C. Sengul, "Multi-user joint task offloading and resources optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2016.

[20] M. H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[21] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, Dec 2016.

[22] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy cost models of smartphones for task offloading to the cloud," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 3, pp. 384–398, Sept 2015.

[23] Q. Qi, J. Liao, J. Wang, Q. Li, and Y. Cao, "Dynamic resource orchestration for multi-task application in heterogeneous mobile cloud computing," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2016, pp. 221–226.

[24] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "Music: Mobility-aware optimal service allocation in mobile cloud computing," in *2013 IEEE Sixth International Conference on Cloud Computing*, June 2013, pp. 75–82.

[25] L. Tianze, W. Muqing, and Z. Min, "A stackelberg game based task offloading mechanism for ad-hoc based mobile cloud computing," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*, Sept 2016, pp. 618–622.

[26] L. Pu, X. Chen, J. Xu, and X. Fu, "D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, Dec 2016.

[27] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 1–1, 2015.

[28] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Processor-network speed scaling for energy: delay tradeoff in smartphone applications," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1–14, 2015.

[29] Q. Ning, C. A. Chen, R. Stoleru, and C. Chen, "Mobile storm: Distributed real-time stream processing for mobile clouds," in *IEEE International Conference on Cloud NETWORKING*, 2015.

[30] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing &#38; Services: Social Networks and Beyond*, ser. MCS '10. New York, NY, USA: ACM, 2010, pp. 6:1–6:5. [Online]. Available: http://doi.acm.org/10.1145/1810931.1810937

[31] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–1, 2015.

[32] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.