Future Generation Computer Systems 85 (2018) 1–8

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

User mobility aware task assignment for Mobile Edge Computing

Zi Wang [a], Zhiwei Zhao [a,c,*], Geyong Min [b], Xinyuan Huang [a], Qiang Ni [d], Rong Wang [e]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
[b] College of Engineering, Mathematics and Physical Sciences, University of Exeter, UK
[c] The Research Institute of Information Technology and National Laboratory for Information Science and Technology (TNList), Tsinghua University, China
[d] School of Computing and Communications, Lancaster University, UK
[e] College of Computer Science, Zhejiang University, China

HIGHLIGHTS

- We formally model the problem of task assignment in MEC as a constraint satisfaction problem, which jointly considers user mobility and resource distributions in the MEC end.
- We propose a lightweight algorithm for assignment, which can increase the resource utilization of the MEC nodes.
- We propose an accurate delay estimation scheme which supports accurate offloading decisions on mobile devices.
- We conduct simulation experiments for performance evaluation. The results show that the proposed work can reduce the task execution delay and increase the resource utilization.

ARTICLE INFO

Article history:
Received 17 August 2017
Received in revised form 26 January 2018
Accepted 8 February 2018
Available online 16 March 2018

Keywords:
Mobile Edge Computing
Task assignment
User mobility
Offloading

ABSTRACT

Mobile Edge Computing (MEC) has emerged as a prospective computing paradigm to provide pervasive computing and storage services for mobile and big data applications. In MEC, many small cell base stations (sBSs) are deployed to establish a mobile edge network (MEN). These sBSs can be usually accessed directly by mobile users. The computational tasks are first offloaded from mobile users to the MEN and then executed in one or several specific sBSs in the MEN. While the offloading decision has been well studied, the task execution delay on the MEN side is overlooked. This paper aims at reducing the task execution delay by task scheduling in MENs. Specifically, we jointly consider the task properties, the user mobility and network constraints. The problem is formalized as a constraint satisfaction problem and a lightweight heuristic solution is proposed for fast scheduling. We conduct simulation experiments to study the performance of the proposed work. The results show that our work is able to significantly reduce the task execution delay in MENs and thus reduces the end-to-end delay for MEC tasks.

1. Introduction

Modern mobile devices are becoming more and more powerful and the mobile applications are becoming increasingly computation-intensive and delay-sensitive, such as real-time online gaming [1], augmented/virtual reality (AR/VR) [2], image/video processing APPs [3] and the vehicle networking systems [4]. These applications can often introduce a large amount of traffic and computational workload, which can potentially cause the battery drain problem on mobile devices [5]. To deal with the resource constraints of mobile devices, Mobile Edge Computing (MEC) has emerged as a promising computing paradigm. In MEC, a number of small cell base stations (sBSs) with computation and storage capability [3] are deployed to construct a network termed as the mobile edge network (MEN). The MENs provide computation and storage services in close proximity to subscribers to meet the high-workload and low-latency requirements [6]. Some projects such as TROPIC [7] and SESAME [8] introduce powerful sBSs, like picocells or femtocells, to share their enhanced high performance computing capabilities for high-performance edge computing. In the MEC paradigm, computational tasks are first offloaded from mobile users to the MENs and then executed in one or several specific sBSs in the network. Compared to traditional cloud computing, MEC is more close to users and incurs much less computational and transmission delay. The deployment of MEC can also provide more flexible resource scheduling for the mobile tasks.

* Corresponding author at: School of Computer Science and Engineering, University of Electronic Science and Technology of China, China.
E-mail addresses: wangzi@mobinets.org (Z. Wang), zzw@uestc.edu.cn (Z. Zhao), g.min@exeter.ac.uk (G. Min), xinyuan@mobinets.org (X. Huang), q.ni@lancaster.ac.uk (Q. Ni), zjajwr@zju.edu.cn (R. Wang).

https://doi.org/10.1016/j.future.2018.02.014
0167-739X/

The current research works on MEC often focus on the offloading decision problem on mobile devices [9,10]. For example, X. Chen et al. [9] employed a game-theoretic computation offloading approach for multi-user systems. S. Guo et al. [10] proposed a dynamic offloading and resource scheduling policy to reduce energy consumption and the application completion time. Some other works such as [11] aimed at utilizing the synergy between the computation resources of MEC and the mobile devices. These works can effectively reduce the computational delay on mobile devices. Although the user-end delay is reduced, we notice that much delay is consumed at the MEC end. Different from the traditional cloud computing where execution delay is stable [12], it can be highly diverse in MEC because the tasks may be executed in different sBSs which have different resource situations. There are several research works on optimizing the task execution delay in MENs [13,14]. Among the problems, the task assignment in the MENs plays an essential role in reducing the execution delay, i.e., assigning the tasks to appropriate sBSs to achieve minimized execution delay in MENs.

The user mobility, which is a fundamental characteristic in MEC, have a large impact on the task execution in two ways. First, mobile users lead to time-varying workload distribution. Different sBSs have different numbers of connected mobile users. Second, mobile users often upload tasks and receive results via different sBSs, involving in-MEN communications and computation workload. However, the existing works often overlook the impact of user mobility and assign tasks to the directly connected sBSs. Some works consider mobility based on intuitive models, which may not reflect the exact impact of user mobility and thus yield inefficient task executions [13].

To address the impact of user mobility on task execution within MENs, we consider task assignment based on user trajectory prediction [15,16]. By jointly considering the user mobility, task properties and the resource distribution in the MEN, we formally model the problem as a constraint satisfaction problem. We then proposed a lightweight heuristic approach to the problem. Based on the task assignment/scheduling scheme, we further propose a delay estimation for MEC tasks to support accurate task offloading on mobile devices. Considering MEC is usually deployed in scenarios such as flight terminals, shopping malls, etc., the user mobility can be highly predictable [16]. We conduct extensive simulation experiments and the results show that the proposed work can significantly reduce the execution time of tasks in MEC networks.

The major contributions of this paper are summarized as follows:

1. We formally model the problem of task assignment as a constraint satisfaction problem, which jointly considers the user mobility, task properties and the resource distributions in the MEC network.
2. We propose a lightweight algorithm for the problem, which can effectively increase the resource utilization at the MEC end.
3. Based on the proposed assignment, we propose an accurate delay estimation scheme which can support accurate offloading decisions on mobile devices.
4. We conduct simulation experiments and show that the proposed work outperforms the existing works in terms of task execution delay.

This rest of the paper is organized as follow. Section 2 summarizes the related works. Section 3 presents the proposed model and algorithm. Section 4 presents the evaluation of the proposed work. Section 5 concludes this work and points future directions.

2. Related work

A large number of existing works have studied the optimization problems on mobile edge computing. Most of these works focus on the decision making problem for task offloading at the user end. These works will first estimate the expected delay or energy consumption for target tasks and then offload them to the mobile edge if the delay or energy consumption is reduced. According to the optimization goal, we divide these works into two categories: works on optimizing energy consumption and works on reducing the task delay. Our work falls into the second category and differs from the existing works in that our aim is to reduce the delay at the network end instead of the user end.

Works on optimizing energy consumption. K. Zhang et al. [17] presented a multi-device computation offloading framework for MEC and formulated an optimization problem that minimized the device energy consumption. A three-stage offloading scheme was proposed to obtain the sub-optimal solution, which (1) classified the mobile device, (2) determined the priority and (3) allocated the radio resource. X. Chen et al. [9] further considered the interference and collisions when there were too many users trying to offload tasks to the same sBS, which can significantly increase the energy consumption of mobile devices. The offloading was formulated as a multi-user game, which was proved always admitting a Nash equilibrium. W. Labidi et al. [18] considered the time varying channel state for wireless offloading and proposed a scheduling scheme for task offloading, which tried to make the best use of wireless channels and user buffers to reduce the energy consumption.

Works on reducing the task delay. J. Liu et al. [11] tried to minimize the execution delay for single users with one-dimensional search algorithm. The algorithm outputted a policy for offloading decision according to the application buffer queuing state. Besides, the characteristics of wireless channels were also considered. Y. Mao et al. [19] jointly optimized the task offloading scheduling and transmitted power allocation problem to reduce the offloading delay. Plachy et al. [20] took the spatial diversity of the sBSs into account in the offloading process. The sBS that was responsible for task execution was chosen by the users. Then the results would be returned to the users via the sBS with the highest RSSI of the wireless links. However, the work was designed for single task offloading.

There are also some works that jointly optimize both energy and task delay. Reducing task delay often adds additional energy consumption in MEC, especially for the tasks that execute faster on mobile devices than on mobile edges. Some works set an energy threshold and then minimize the execution delay without exceeding the energy threshold. For example, Y. Mao et al. [14] proposed a dynamic offloading scheme for single user to minimize the execution delay for energy harvesting devices, where the energy harvesting technique added complexity to the offloading algorithms. A lightweight approximation approach was proposed to achieve a good tradeoff between complexity and the delay minimization. J. Yang et al. [21] used a multi-stage sequential game model to meet the energy and delay requirements at the same time.

Different from the aforementioned works, our work emphasizes on minimizing the delay within the MENs. The delay in MENs not only adds to the overall task execution delay but also impacts the decision making process at the user end. We propose a novel in-network task scheduling approach to reduce the task execution delay, where task information, resource information and user mobility are jointly considered. Different from the [13] which used the contact rate to capture user mobility, our work is built on top of the user trajectory prediction [15,16], which can accurately capture user mobility in most MEC scenarios such as airports, shopping malls, etc. Besides, to deal with the impact on offloading decision, we propose a novel delay estimation scheme for mobile devices.

3. Mobility aware task assignment for MEC

In this section, we present the application scenario for MEC, the motivation of mobility aware design and the main algorithm design for task assignment in MEC.

3.1. Impact on task execution

A mobile edge network (MEN) contains multiple small cell base stations (sBSs) and covers a certain area such as airport, shopping mall, library, etc. The mobile users carry mobile devices that have computation-intensive tasks. The sBSs are connected to construct a network providing storage and computational services. A mobile user may move from one sBS to another sBS during the task execution time or the media caching progress. For example, an AR navigator needs to load the indoor map and monitors the user trajectory. The AR information such as shop vouchers needs to be loaded and displayed in the users' devices. Considering a user is roaming among the sBSs, the computational results of the AR display should be delivered to the user via different sBSs along the user's moving path. In such cases, the tasks need to be carefully scheduled to be executed in the most appropriate sBSs which need to be close to the user's position (time varying) and should have enough resource allocated to finish the tasks before deadline. Intuitively, when the tasks are lightweight and can be executed within the period of the user's stay at the corresponding sBS, the task should be executed immediately at the sBS and then returned to the mobile user. When the tasks are heavyweight and cannot be finished in the stay period of the user, the tasks should be split into many sub-tasks and transfer some tasks to the successive sBSs along the user's trajectory. As a result, many tasks can be processed in parallel and the user can keep receiving the task results along the path without waiting for the task executions. In a nutshell, all the sBSs along the user's path can be utilized for executing the offloaded tasks. Therefore, the delay can be reduced compared to the case where only one specific sBS is selected for executing all tasks.

3.2. Impact on the offloading decisions

Apparently, if the tasks are dynamically scheduled at the MEN end, the current delay estimation on the execution delay is no longer applicable. The challenge for the mobile user to estimate the task execution delay lies in that the resource at the MEN end is unknown to the mobile users. Therefore, different from the existing schemes, we need to either estimate the resource usage at the MEN end or establish a feedback mechanism to notify the mobile user about the resources or execution delay information. Besides, there may be other users that offload multiple tasks to the MEN, which also has a large impact on the delay estimation at the user end.

3.3. Design

The MEC system is illustrated in Fig. 1 for the indoor scenario. The mobile users offload their computation-intensive and delay-sensitive tasks to the mobile edge networks which consist of a number of sBSs. In the system, we consider a set $N = \{1, 2, 3, \ldots, N\}$ of sBSs which is followed a mesh topology that is similar to [22] cover data distribution in the network. All the sBSs are capable for receiving, executing and transferring the offloaded tasks. We also assume software-defined network architecture is used in the MEN. All sBSs are controlled and monitored by a central controller. The users move round the area and their trajectories can be predicted using existing approaches [16,23]. Our goal is to minimize the execution time in the MEN end by scheduling the multiple tasks from multiple users. The task offloading information is first uploaded to the sBSs and then collected to the central controller. After that, the central controller computes the best strategy for scheduling the offloaded tasks. Before presenting the model, we first summarize the notations as in Table 1.

We consider a set of users U and each user i has a bunch of computation tasks $T_i = t_{i,j}|j = 1, 2, \ldots |T_i|$, which can be offloaded to the MEN. For each user i, the trajectory P_i consists of a number of sBSs. To reduce the delay at the user end, the tasks at the users are offloaded to the MEN. Next we need to assign the offloaded tasks from multiple users to the sBSs along the corresponding paths. $d^e_{(i,j),k} = 1$ if the task (i, j) is assigned to the sBS sBS_k.

Our aim now is to find the most appropriate $D^e = \{d^e_{(i,j),k}|i \in U, j \in T_i, k \in P_i\}$ to maximize the total gains of MEC, i.e., the average delay reduction for all users. Then the task scheduling can be modeled as an optimization problem as follows:

$$\begin{aligned} &\max_{D^e} \frac{1}{|U|} \sum_{i \in U} \sum_{j \in T_i} d_{i,j}(t^l_{i,j} - t^{edge}_{i,j}) \\ &s.t. \ \ \forall t_{i,j}, \sum_{k \in P_i} d^e_{(i,j),k} \leq 1 \end{aligned} \tag{1}$$

As depicted in the above equations, each task (i, j) should be executed no more than once along the user i's trajectory P_i. Our job now is to determine $d^e_{(i,j),k}$ for all $i \in U, j \in T_i$ and $k \in P_i$.

The time spent on the edges $t^{edge}_{i,j}$ is given by:

$$t^{edge}_{i,j} = \sum_{k \in P_i} t^q_{(i,j),k} + t^e_{(i,j),k} + t^t_{i,j} \tag{2}$$

where the task transferring time $t^t_{i,j}$ is given by:

$$\begin{aligned} t^t_{i,j} = &\frac{s_{i,j}}{ru_{i,j}} + \frac{\eta_{i,j} s_{i,j}}{rd_{i,j}} \\ &+ \Big(\sum_{k \in P_i} \big(w_{(i,j),k} \frac{s_{i,j}\eta_{i,j}}{rt_{i,j}} + (1 - w_{(i,j),k}) \frac{s_{i,j}}{rt_{i,j}}\big) - \frac{s_{i,j}}{rt_{i,j}}\Big) \end{aligned} \tag{3}$$

where the first item denotes the time for task uploading, the second item denotes the time for results downloading, the last item denotes the time for in-MEN task transfer. It is worth noting that $ru_{i,j}$ and $rd_{i,j}$ are determined by the specific sBSs that are selected for uploading and downloading the tasks. $w_{(i,j),k}$ is an indicator to check whether task (i, j) has already been executed before arriving sBS_k. $w_{(i,j),k} = 1$ if the task has been executed before sBS_k and $w_{(i,j),k} = 0$ otherwise. The output data size is calculated as $\eta_{i,j} s_{i,j}$. As the first item already accounts the time of task transfer from the user to the first sBS, we need to eliminate an extra in-MEN task transfer time (as calculated in the last item).

The queuing time for task (i, j) on sBS_k equals to the time of the execution time for all tasks that are before $t_{(i,j),k}$. and is given by:

$$t^q_{(i,j),k} = \sum_{n=1}^{idx(t_{(i,j),k})} t^e_n \tag{4}$$

where $idx(t_{(i,j),k})$ denotes the index for task (i, j) in sBS_k. t^e_n denotes the execution time for task n and is given by $\frac{l_n}{f_k}$. l_n denotes the execution load of task n and f_k denotes the CPU frequency of sBS_k.

3.4. The heuristic algorithm

The above optimization problem is a constraint satisfaction problem, which is NP-complete [24]. To solve the problem, we propose a heuristic as follows.

Each user first upload a summary of their tasks including the data size, execution load (# of CPU instructions, which is known

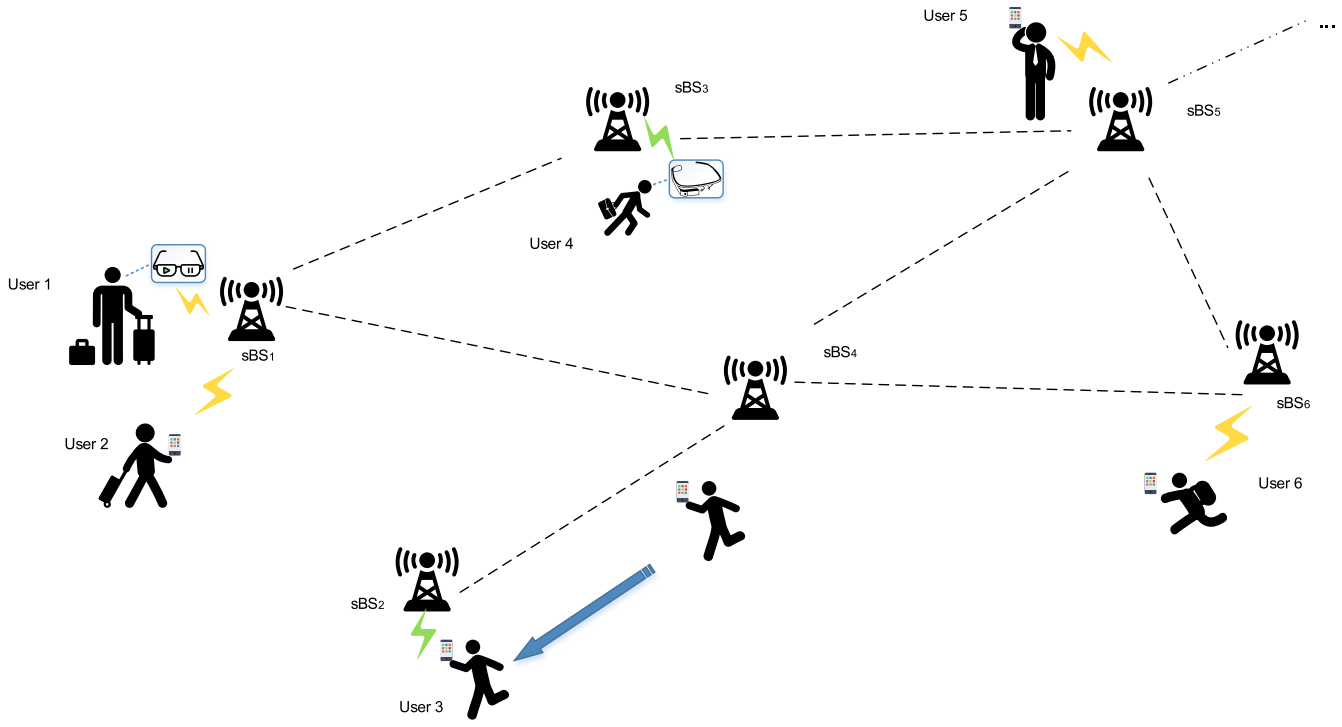

Fig. 1. System model.

Table 1
Input & Output.

Parameter	Description
s_i	The size of data for offloading task i
c_i	CPU cycles required for task i
t_i^l	The local execution time for task i (on the user's device)
P_i	The set of sBSs on user i's path
T_i	The set of tasks of user i
U	The set of all users with offloading tasks
$d_{i,j}$	The offloading decision for task j of user i, $d_{i,j} \in \{0, 1\}$
$d^e_{(i,j),k}$	The task assignment for task j of user i on sBS k, $d^e_{(i,j),k} \in \{0, 1\}$
D^e	The task assignment set for all sBS, $D^e = \{d^e_{(i,j),k} \mid i \in U, j \in T_i, k \in P_i\}$
f_i	The CPU frequency for executing task i
sBS_i	The ith small base station
$t^{edge}_{i,j}$	The total time spent on the edge sBSs for task $t_{i,j}$
$t^e_{(i,j),k}$	The execution time of task i on sBS j on sBS k
$t^q_{(i,j),k}$	The queuing time for task (i, j) on sBS k
$t^t_{i,j}$	The task transferring time for $t_{i,j}$
ru_i	The uploading transmission rate for task i
rd_i	The downloading transmission rate for task i
rt_i	The transmission rate for task transfer among sBSs
η_i	The ratio of input/output data size for task i

for a specific application), the local execution time and the expected output data size. Once received the information, the central controller of MEN processes each task by assigning it to the sBS that achieves the lowest delay. During the process of updating the queue time on a sBS, the criteria of choosing which task that will be served preferentially when the conflict happens is to select the task with minimal execution time. If the task execution delay with the best sBS is still larger than the local execution time, the task will be rejected and executed locally. When all tasks have been assigned an sBS or rejected, the assignment progress is finished. The above algorithm is shown in Algorithm 1.

Next we use an illustrating example in Fig. 2 to elaborate our scheme. There are three sBSs and five tasks ($t_0 \sim t_4$) which need to be assigned. The task information is shown in the table, and the queue buffer is left empty before assignment.

At first, we calculate the t^{edge} for each task on all sBSs along its path and then select the sBS with the minimal t^{edge} to allocate the task. Next, we check the conflict when more than one tasks are allocated to the same sBS. For example, in Fig. 2 step 2, we can see the T_4 is conflicted with T_0 and T_1 is conflicted with T_3. So, we need to update the queue time and re-calculate the t^{edge} for those conflict tasks. The sBS will choose the task with minimal execution time as a prior one. Therefore, the queue time can be updated through Eq. (4). After updating the queue time, we need to re-calculate the t^{edge} for those conflicting tasks, which can be seen in step 3. The tasks can be re-allocated to the more optimal sBSs after updating their t^{edge} in step 4, and then the process repeats

Task	Path	Task Input Data (s_i)	Upload & Download Time (s_i/ru_i & $\eta*s_i/rd_i$,)	Task & Result Transmit Time (s_i/rt_i & $\eta*s_i/rt_i$)	sBS Execution Time (t^e)	Local Execute Time (t^l)
T_0	sBS_0 -> sBS_1-> sBS_2	25MB	2.5s & 0.5s	0.25s & 0.05s	10s	100s
T_1	sBS_1 -> sBS_2	30MB	3.0s & 0.6s	0.30s & 0.06s	8s	80s
T_2	sBS_2 -> sBS_1-> sBS_0	30MB	3.0s & 0.6s	0.30s & 0.06s	5s	50s
T_3	sBS_1 -> sBS_2	20MB	2.0s & 0.4s	0.20s & 0.04s	9s	90s
T_4	sBS_0 -> sBS_1	15MB	1.5s & 0.3s	0.15s & 0.03s	4s	40s

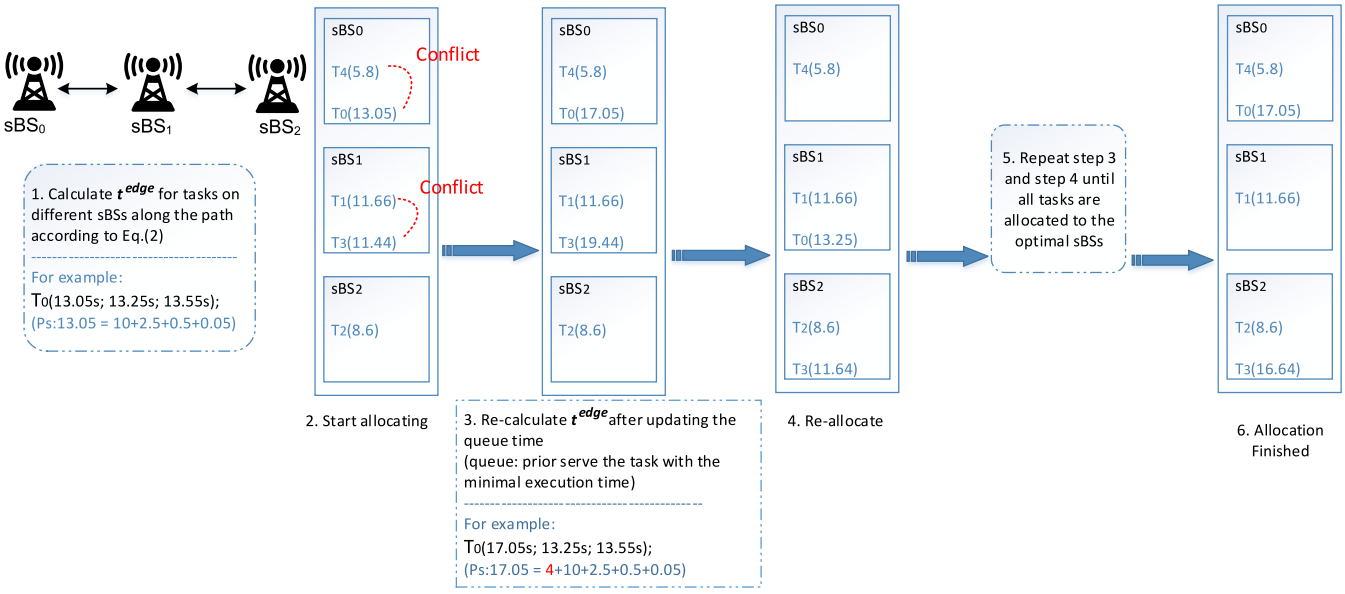

Fig. 2. Illustration for the proposed algorithm.

Task	Path	Task Input Data (s_i)	Upload & Download Time (s_i/ru_i & $\eta*s_i/rd_i$,)	Task & Result Transmit Time (s_i/rt_i & $\eta*s_i/rt_i$)	sBS Execution Time (t^e)	Local Execute Time (t^l)
T_a	sBS_1 -> sBS_2	25MB	2.5s & 0.5s	0.25s & 0.05s	10s	100s
T_b	sBS_1 -> sBS_2	30MB	3.0s & 0.6s	0.30s & 0.06s	12s	120s

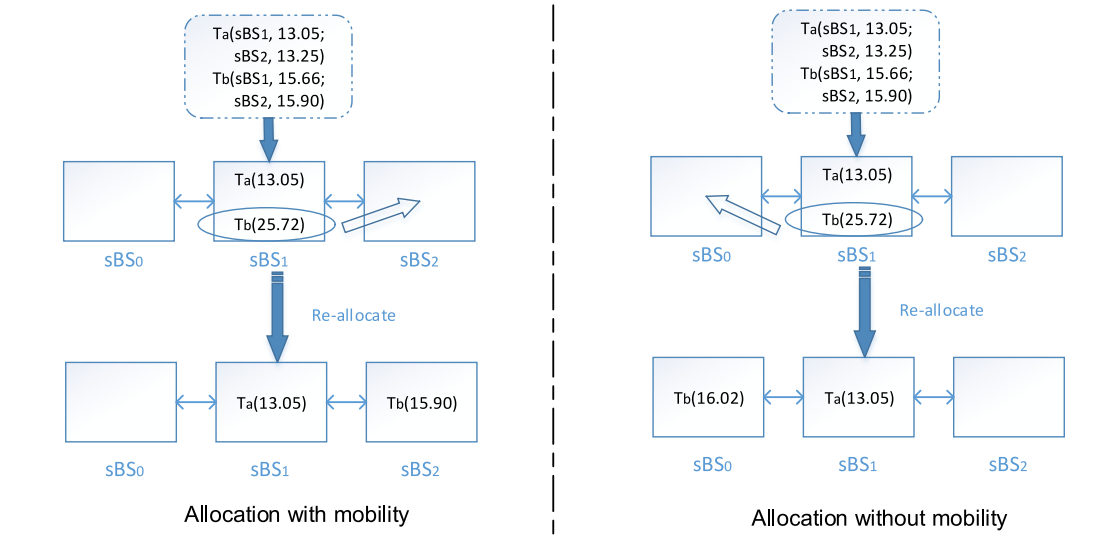

Fig. 3. Illustration for the mobility comparison.

Algorithm 1: The Assignment Scheme

Input:

1. The task information for each task (i, j):

   the local execution time $\{t^l_{i,j}\}$;the input data size and the output ratio, $s_{i,j}$ and $\eta_{i,j}$;

2. The sBS information:

   the involved sBSs along user i's trajectory P_i; the CPU frequency for each sBS i, f_i; the transmission rates ru, rd, rt;

Output: The decision matrix for all tasks and all sBSs, D^e

```
for each i ∈ U do
    for each j ∈ T_i do
        for each k ∈ P_i do
            d^e_{(i,j),k} = 0 //initiation before assignment
        end
        k_t = arg_k min t^{edge}_{i,j}
        if t^{edge}_{i,j} < t^l_{i,j} then
            d^e_{(i,j),k_t} = 1
        end
        else
            d^e_{(i,j),k_t} = 0 //the task will not be offloaded to the MEN
        end
    end
end
```

above steps until all tasks are allocated to the optimal sBSs. Finally, T_4 and T_0 are executed at sBS_0, T_1 is executed at sBS_1, T_2 and T_3 are executed at sBS_2.

Studies in [15] and [16] presented the user mobility prediction which can achieve an accuracy about 90% to map a moving path for users with their proposed algorithm by using Lagrange's interpolation and non-parametric approach based on kernel density estimation, respectively. These methods may be more accurate under an indoor scene like our scenario. In Fig. 3, we illustrate the comparison between the allocation considering mobility and that without considering mobility. In the left part, at first, two tasks will be allocated to the sBS according to our algorithm based on the mobility prediction, and the tasks are finally allocated to the sBSs along the user moving path with minimal delays. In the right part, the same tasks will be allocated to the sBS according to the random sBS selection algorithm, which does not consider the user mobility. T_a chooses sBS_1 as the serving sBS. The serving sBS_1 computes the execution delays, and it chooses it own as computing sBS due to the shortest delay. However, T_b chooses sBS_1 as the serving sBS and randomly selects sBS_0 as the computing sBS to avoid the queue time which is caused by sBS_1 executing T_a. As a result, the delays of T_a and T_b by using the method which does not consider the user mobility is larger than our scheme.

4. Evaluation

In this section, we use simulation experiments to numerically evaluate our proposed work. Table 2 shows the parameters used for the simulation. These settings are in line with the recommendations for small cell networks defined by 3GPP and the open source data from telecom operators [25].

Given the small coverage of small cell base station and the Multi-Hop Cognitive Radio Networks in [26], we consider the Orthogonal Frequency-Division Multiple Access (OFDMA) used for communication in downlink and the Single-Carrier Frequency-Division Multiple Access (SC-FDMA) used for uplink. As a result, the interference among users could be ignored. We assume that the input data size of each task ranges from 1 MB to 30 MB, and the CPU cycles requirement ranges from $2 * 10^9$ cycles per task (cpt) to $2 * 10^{10}$ cpt for different type of applications. Besides the different CPU frequency between sBS and user device, we additionally consider the impact of acceleration rate on CISC and RISC for different type of devices. Based on the parameter presented in Table 2, we assume that user mobility is a constant speed following the predicted path P that we describe in 3.3. The users upload their tasks to the closest sBS with the best wireless link quality and obtain the results from the output sBS according to the assignment result.

Table 2
Simulation settings.

Parameter	Value/Range
Small cell base station (sBS) uplink/downlink	80 Mbps/80 Mbps
Intra MEN transmission rate	1000 Mbps
sBS CPU frequency & instruction set	2 GHz & CISC
sBS coverage radius	10 meters
User device CPU frequency & instruction set	2 GHz & RISC
Moving speed of the mobile users	1 m/s
Input data size	1∼30 MB
CPU cycles required by one task	$2 * 10^9$∼$2 * 10^{10}$
Acceleration ratio between CISC and RISC	10∼50
The ratio of input/output data size	5:1
Number of users & sBSs for experiment 1	200 & 0∼30
Number of users & sBSs for experiment 2	0∼600 & 10

Similar to the SDN architecture, all sBSs are directly connected to a central controller. The task information is gathered at the controller and all tasks are assigned at the controller. It is worth mentioning that the delay for information collection and the assignment itself is not considered in the simulation because the amount of the summary information is very small and the assignment is done in the powerful central controller. Although we do not assume the wireless information from different users, the problem can be solved using the existing solutions [27–29]. Since we consider the end-to-end uploading/downloading delay in the model, the existing works on collision resolution can be directly incorporated in our model. The only difference will be the estimation of the uploading and downloading rates ru and rd.

We use four different policies for comparison.

1. All tasks are executed at the user end.
2. Optimized assignment without considering the user mobility (trajectories). The tasks executed at the task uploading sBS and the results are sent to the user via sBSs along the user path (can be directly connected to users).
3. Optimized assignment considering mobility with maximal optimized execution time (maximizing the optimization time) for each task.
4. Our scheme — Optimized assignment considering mobility with prior serving the task with minimal execution time.

It is worth noting that the sequence for processing the tasks has important impact on the end-to-end performance. The third policy uses a criteria for first executing the task with maximal difference between the local and the edge. We repeat our experiments for 100 times to get the average execution time under different conditions.

Fig. 4 shows the average task delays for different network scales in terms of sBSs. We can see that (1) the local execution achieves the largest delay and our optimal optimization with mobility achieves the lowest delay. The reason is straightforward as follows. For local execution, the MEC resources are not used at all; For optimization that utilizes the MEC resource: if mobility is not considered, only the sBS directly connected to the user is utilized and

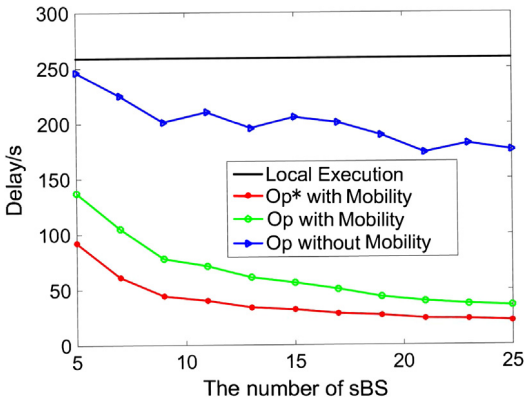

Fig. 4. The average delay for different network scales.

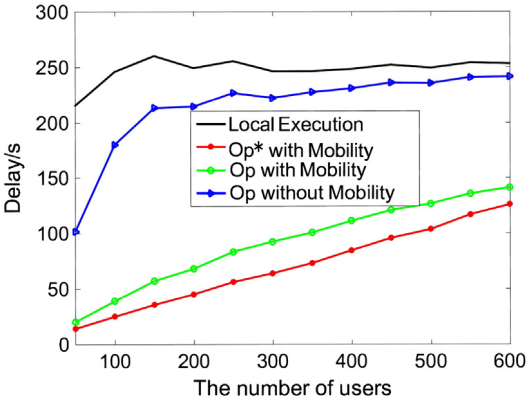

Fig. 5. The average delay for different user scales.

much queuing delay will be incurred. When mobility is considered and multiple sBSs are utilized along the user trajectory, the delay can be further reduced. (2) The delay reduction increases as the network scale increases. The reason is that when there are more sBSs, more resources can be utilized to enhance the performance for mobile devices. (3) The speed for delay reduction decreases as the network scale increases. This is because that as more and more tasks from mobile devices are offloaded to the MEN, there are fewer resources left to be used for optimization. As a result, the speed of reduction is reduced. (4) The third policy achieves sub-optimal result. The reason is that different sequence for task assignment may cause that some computation-intensive tasks are assigned to resource constrained sBSs and lightweight tasks are assigned to more powerful sBSs or be rejected, which is a waste of MEC resource.

Fig. 5 shows the averaged delay for different user scales with the same MEN. The MEN consists of ten sBSs. The user size grows from 50 to 600 and the trajectories are randomly generated. It can be inferred that as there are more and more users, the delay reduction becomes less. This is because there will be less resource available for each task as the number of users keeps growing. This simulation result can also be used to guide the sBS deployment for a given number of mobile users. We can also compared to the case with fixed number of users (Fig. 4), the delay reduction of optimization considering mobility compared to optimization without mobility is enlarged. The reason is that when there are multiple tasks and multiple users, the parallelism of the sBSs can have more optimization space compared to that with small number of tasks and users.

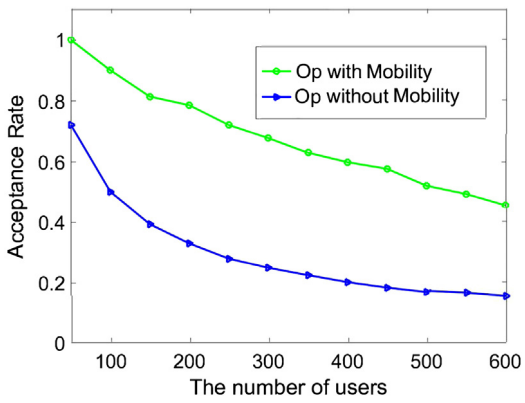

Fig. 6. The acceptance rate for different user scales.

We dig into the decision process of the proposed algorithm. Fig. 6 shows the acceptance rates for tasks in different situations that if we consider the mobility or not. The MEN consists of ten sBSs. The user size grows from 50 to 600 and the trajectories are randomly generated. We can see that all tasks are accepted by the MEN at first, and the acceptance rate maintains 45.6% when the number of users expand to 600, which means the delay can be effectively reduced for tasks using our optimization scheme considering mobility. Differently, only 15% users are accommodated in the MEN when there are 600 users for optimization without considering mobility. Therefore, our scheme is more appropriate for the dense MEN with more users.

5. Conclusion

In this paper, we investigate the problem of task assignment in Mobile Edge Computing for multi-task multi-user scenarios. We model the assignment as a constraint satisfaction problem and propose a lightweight algorithm, where the task information, small base station information and the user mobility are jointly considered. We conduct simulation experiments to study the performance of the proposed work. The results show that compared to the works without considering user mobility, the proposed work can greatly reduce the task execution delay. We will focus on finding the optimal sequence for task assignment and further optimize the task delay by considering the wireless collisions among different users.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61602095), the National Key Research and Development Program of China (2017YFB1400102), the Fundamental Research Funds for the Central Universities, China (No. ZYGX2016KYQD098 and No. ZYGX2016J079), the Quancomm Research Fund (Tsinghua), China, National Postdoctoral Program for Innovative Talents of China, and the EU FP7 CLIMBER, UK project under Grant Agreement No. PIRSES-GA-2012-318939.

References

[1] D. Meilander, F. Glinka, S. Gorlatch, L. Lin, W. Zhang, X. Liao, Bringing mobile online games to clouds, in: Computer Communications Workshops, INFOCOM WKSHPS, 2014 IEEE Conference on, IEEE, 2014, pp. 340–345.
[2] W. Barfield, Fundamentals of Wearable Computers and Augmented Reality, CRC Press, 2015.
[3] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Commun. Surv. Tutor. (2017).

[4] Y. Wu, G. Min, L.T. Yang, Performance analysis of hybrid wireless networks under bursty and correlated traffic, IEEE Trans. Veh. Technol. 62 (1) (2013) 449–454.
[5] X. Ma, P. Huang, X. Jin, P. Wang, S. Park, D. Shen, Y. Zhou, L.K. Saul, G.M. Voelker, eDoctor: Automatically diagnosing abnormal battery drain issues on smartphones, in: NSDI, Vol. 13, 2013, pp. 57–70.
[6] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, et al., Mobile-edge computing introductory technical white paper, in: White Paper, Mobile-Edge Computing, MEC Industry Initiative, 2014.
[7] F.E. Project, Distributed computing, storage and radio resource allocation over cooperative femtocells (TROPIC), 2012. Available: http://www.ict-tropic.eu/, [Online].
[8] H.E. Project, Small cells coordination for multi-tenancy and edge services, sesam, 2015. Available: http://www.sesame-h2020-5g-ppp.eu/, [Online].
[9] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (5) (2016) 2795–2808.
[10] S. Guo, B. Xiao, Y. Yang, Y. Yang, Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing, in: Computer Communications, IEEE INFOCOM 2016-the 35th Annual IEEE International Conference on, IEEE, 2016, pp. 1–9.
[11] J. Liu, Y. Mao, J. Zhang, K.B. Letaief, Delay-optimal computation task scheduling for mobile-edge computing systems, in: Information Theory, ISIT, 2016 IEEE International Symposium on, IEEE, 2016, pp. 1451–1455.
[12] D. Huang, et al., Mobile cloud computing, in: IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter, Vol. 6, No. 10, 2011, pp. 27–31.
[13] C. Wang, Y. Li, D. Jin, Mobility-assisted opportunistic computation offloading, IEEE Commun. Lett. 18 (10) (2014) 1779–1782.
[14] Y. Mao, J. Zhang, K.B. Letaief, Dynamic computation offloading for mobile-edge computing with energy harvesting devices, IEEE J. Sel. Areas Commun. 34 (12) (2016) 3590–3605.
[15] T.M.T. Do, O. Dousse, M. Miettinen, D. Gatica-Perez, A probabilistic kernel method for human mobility prediction with smartphones, Pervas. Mob. Comput. 20 (2015) 13–28.
[16] B. Li, H. Zhang, H. Lu, User mobility prediction based on Lagrange's interpolation in ultra-dense networks, in: Personal, Indoor, and Mobile Radio Communications, PIMRC, 2016 IEEE 27th Annual International Symposium on, IEEE, 2016, pp. 1–6.
[17] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, Y. Zhang, Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks, IEEE Access 4 (2016) 5896–5907.
[18] W. Labidi, M. Sarkiss, M. Kamoun, Energy-optimal resource scheduling and computation offloading in small cell networks, in: Telecommunications, ICT, 2015 22nd International Conference on, IEEE, 2015, pp. 313–318.
[19] Y. Mao, J. Zhang, K.B. Letaief, Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems, in: Wireless Communications and Networking Conference, WCNC, 2017 IEEE, IEEE, 2017, pp. 1–6.
[20] J. Plachy, Z. Becvar, P. Mach, Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network, Comput. Netw. 108 (2016) 357–370.
[21] J. Yang, B. Jiang, Z. Lv, K.-K.R. Choo, A task scheduling algorithm considering game theory designed for energy management in cloud computing, Future Gener. Comput. Syst. (2017).
[22] G. Min, Y. Wu, A.Y. Al-Dubai, Performance modelling and analysis of cognitive mesh networks, IEEE Trans. Commun. 60 (6) (2012) 1474–1478.
[23] X. Chen, X. Wu, X.-Y. Li, X. Ji, Y. He, Y. Liu, Privacy-aware high-quality map generation with participatory sensing, IEEE Trans. Mob. Comput. 15 (3) (2016) 719–732.
[24] C. Carbonnel, M.C. Cooper, Tractability in constraint satisfaction problems: a survey, Constraints 21 (2) (2016) 115–144.
[25] E.U.T.R. Access, Further Advancements for E-UTRA Physical Layer Aspects, 3GPP TR 36.814, Tech. Rep., 2010.
[26] Y. Wu, G. Min, A.Y. Al-Dubai, A new analytical model for multi-hop cognitive radio networks, IEEE Trans. Wirel. Commun. 11 (5) (2012) 1643–1648.
[27] X. Ji, Y. He, J. Wang, K. Wu, D. Liu, K. Yi, Y. Liu, On improving wireless channel utilization: A Collision tolerance-based approach, IEEE Trans. Mob. Comput. 16 (3) (2017) 787–800.
[28] M. Deng, H. Tian, X. Lyu, Adaptive sequential offloading game for multi-cell mobile edge computing, in: Telecommunications, ICT, 2016 23rd International Conference on, IEEE, 2016, pp. 1–5.
[29] X. Ji, Y. He, J. Wang, K. Wu, K. Yi, Y. Liu, Voice over the dins: improving wireless channel utilization with collision tolerance, in: Network Protocols, ICNP, 2013 21st IEEE International Conference on, IEEE, 2013, pp. 1–10.

Zi Wang received his B.S. degree at the College of Computer Science and Engineering in University of Electronic Science and Technology of China (UESTC). He is currently a M.S. student in UESTC. His research interests focus on Mobile Edge Computing.

Zhiwei Zhao received his Ph.D. degree at the College of Computer Science, Zhejiang University in 2015. He is currently an assistant professor at the College of Computer Science and Engineering in University of Electronic Science and Technology of China. His research interests focus on wireless computing, heterogeneous wireless networks, protocol design and network coding. He is a member of IEEE.

Geyong Min is the Chair Professor and Director of High Performance Computing and Networking (HPCN) Research Group at the University of Exeter, UK. He received the Ph.D. degree in Computing Science from the University of Glasgow, UK, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. He joined the University of Bradford as a Lecturer in 2002, became a Senior Lecturer in 2005 and a Reader in 2007, and was promoted to a Professor in Computer Science in 2012. His main research interests include Next-Generation Internet, Wireless Networks, Mobile Computing, Cloud Computing, Big Data, Multimedia Systems, Information Security, System Modeling and Performance Optimization.

Xinyuan Huang received his B.S. degree at the College of Computer Science and Engineering in University of Electronic Science and Technology of China (UESTC). He is currently a M.S. student in UESTC.

Qiang Ni is a professor in the School of Computing and Communications, Lancaster University. His current research interests are Wireless Networking and Communications, including Energy-Efficient Green Communications, Cognitive Radio Networks, Broadband Wireless, Intelligent Communication Techniques, Context-User-Aware SDN Networks, Fog-Cloud computing, Smart Grids Communication, Vehicular Networks, Quantum Communication and Mobile Positioning.

Rong Wang received his B.S. degree at the College of Computer Science and Engineering in University of Electronic Science and Technology of China (UESTC). He is currently a M.S. student in UESTC.