2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

Task Offloading and Service Migration Strategies for User Equipments with Mobility Consideration in Mobile Edge Computing

Yan Ding[*][†], Chubo Liu[*][†], Kenli Li[*][†], Zhuo Tang[*][†] and Keqin Li[*][†][‡]
[*]College of Information Science and Engineering, Hunan University, Changsha, Hunan 410082, China.
[†]National Supercomputing Center in Changsha, Hunan 410082, China.
[‡]Department of Computer Science, State University of New York, New Paltz, New York 12561, USA.
Email: {ding, liuchubo, lkl, ztang}@hnu.edu.cn and lik@newpaltz.edu

***Abstract*—Recently, a great number of works have focused on task offloading optimization in mobile edge computing (MEC). However, rare works involve user equipment (UE) mobility. When involving mobility in MEC, the problem becomes even harder. Even a slight movement of UE can significantly affect the strategy and overhead of the UE. Usually, the types of UE mobility can be categorized as random mobility, short-term predictable mobility, and fully known mobility, depending on whether the future location of the UE is known. In this paper, we aim to optimize task offloading and service migration for UEs with different mobility considerations. Specifically, we try to find appropriate task offloading and service migration strategies to optimize energy consumption or latency of UEs according to the characteristics of different mobility types. We conduct extensive experiments using the real world data which records the movement trajectory of UEs. Experimental results show that our methods perform better compared to six other common strategies and can further reduce the overhead of UEs by using their mobility characteristics.**

***Index Terms*—Mobile edge computing, service migration, task offloading strategy, user equipment with mobility.**

I. Introduction

A. Motivation

MEC is proposed by European Telecommunications Standards Institute (ETSI) in 2014. It is an architecture that provides computing resource to UEs at the edge of the network, aiming to improve the quality of service (QoS) and the quality of experience (QoE) [1]. MEC utilizes high-speed wireless network technologies such as 5G to implement communication between UEs and MEC servers, reducing the data transmission delay [2]. UEs offload their heavy tasks to MEC servers for executing to reduce the overhead (e.g., energy consumption and latency) of the UEs [3].

Although a great number of works have studied on task offloading optimization in MEC [4], [5], [6], [7], [8], [9], [10], it is still an open issue to investigate the optimization problem [2]. This is because the above works do not consider the UE mobility, and ignore the impact of the movement of UEs on task offloading strategy [11]. If UEs move far away from the MEC server that is responding to their request, this could result in significant QoS and QoE degradation, and service interruption due to long transmission latency of the offloaded task [2], [3].

However, when involving mobility in MEC, the problem becomes even harder. The reason lies in that the service (the basic environment for executing the task of UEs, e.g., virtual machine and docker) migration is introduced into the process of task offloading optimization [11], [12], [13]. To our knowledge, rare works involve UE mobility. Based on the mobility type, the research can be classified into three categories: i) For UEs with random mobility, Taleb et al. [13] proposed a Markov decision process based algorithm and Sun et al. [14] studied the problem under long-term cost budget constraint; ii) For UEs with short-term predictable mobility, Wu et al. [15] and Plachy et al. [16] respectively proposed a location prediction method; iii) For UEs with fully known mobility, Wang et al. [17] minimized the overhead by considering the task properties and mobility of UEs jointly. Although the above works propose some methods to make the service migration strategy and minimize the overhead caused by UE mobility, they assume that the offloading decision has been made ahead. Yu et al. [18] introduced the service migration decision in the process of making offloading strategy, but they did not make the resource allocation strategy of UE. Meanwhile, all the above works study only one mobility type of service migration and do not investigate the impact of different mobility types on the service migration strategy and overhead of UE.

B. Our Contributions

In order to address the above limitations, in this paper, we study the task offloading and service migration strategies for UEs with different mobility types, and propose several methods accordingly. The methods we propose not only determine the offloading decision, but also allocate the resources of the UEs and decide whether to perform the service migration.

Based on the above discussions, in this work, we involve the following three questions: i) How to make the task offloading strategy for UEs with mobility? ii) How to reduce the overhead of UEs by jointly considering the task offloading and service migration strategies? iii) How to determine the strategies to further minimize the overhead for UEs based on the characteristics of mobility types?

In order to address the above issues, the problem is to schedule the offloaded tasks of UEs with mobility on a set of MEC servers as well as to allocate the CPU frequency and transmission power of the UEs, while making service migration decision, such that the overhead of the UEs is minimized. The main contributions of our work are as follows.

- We formulate two overhead optimization problems, i.e., the energy minimization problem with resource and latency constraints, and the latency minimization problem with resource and energy constraints. Meanwhile, we prove that there are optimal solutions to the problems when the service deployment strategy is given.
- We propose three appropriate task offloading and service migration strategies to optimize energy consumption or latency of UEs according to the characteristics of different mobility types.
- We conduct the extensive experiments using the real world data which records the movement trajectory of UEs. The convergence of algorithms, the effectiveness of algorithms to reduce the overhead of UEs, and the impact of mobility types and various key parameters on the strategy and overhead are demonstrated by the experiments.

To the best of our knowledge, this is the first paper that jointly investigates task offloading and service migration strategies optimization for UEs with different mobility types. The scenario is closer to the real-world.

The rest paper is outlined as follows. Section 2 presents the system model and formulations of the problems studied in this paper. Section 3 develops the algorithms in detail. Section 4 describes the simulation experiments and evaluates the performance of the algorithms. Section 5 gives our conclusions.

II. System Model and Problem Formulation

A. System Model

We consider a scenario as shown in Fig. 1. We use UE$_i$ to represent a UE and VM$_i$ to represent the service instance of UE$_i$, where $1 \leq i \leq N$. Each UE has an application that contains a set of tasks. The task set of UE$_i$ is represented by A_i. $a_k \in A_i$ represents a task of UE$_i$, where $|A_i| \geq k \geq 1$. a_0 indicates that the task that can only be executed locally. w_{i,a_k} represents the workload of a_k, which indicates the number of CPU clock cycles required to complete the task. δ_{i,a_k} represents the processing density of a_k (bits per cycle). The location of UE$_i$ is (x_{i,a_k}, y_{i,a_k}), where x_{i,a_k}, y_{i,a_k} are the longitude and latitude of UE$_i$ when $a_k \in A_i$ is executed. We consider a discrete moving process that a task is executed each time UE$_i$ arrives at a new location [15], [16], [17]. As shown in the figure, the dot on the movement trajectory is the task execution location. When a task is completed, the UE moves to the next location. Meanwhile, UEs have their own movement feature. For example, UE$_1$ moves randomly, which means that we do not know anything about the future location of the UE.

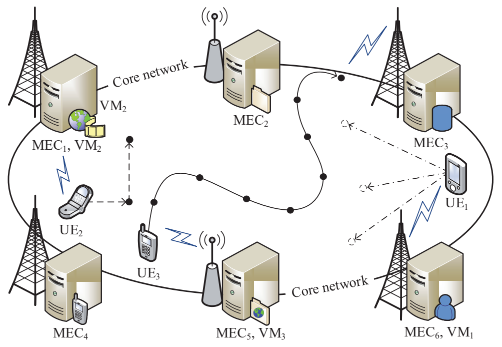


Fig. 1. An illustration of the scenario studied in this paper.

UE$_2$ moves in a certain regularity, and we can predict the next location of the UE. UE$_3$ moves in a given route, which means that we can know all future locations of the UE.

We use MEC$_j$ to represent an MEC server, where $1 \leq j \leq M$. MEC servers are connected to the backbone network, so high-speed data transmission between the MEC servers is possible. MEC servers are stationary and the location of the servers can be represented by (x_j, y_j), where x_j, y_j are the longitude and latitude of the server. We use $I_{i,a_k} = j$ to represent the deployment location of VM$_i$ when $a_k \in A_i$ is executed. For example, VM_3 is deployed at MEC_5, thus $I_{3,a_k} = 5$. We use a binary variable $\lambda_{i,j,a_k} \in \{0,1\}$ to indicate whether a_k is executed at MEC$_j$. If a_k is offloaded to MEC$_j$, then $\lambda_{i,j,a_k} = 1$, otherwise $\lambda_{i,j,a_k} = 0$. $\lambda_{i,0,a_k} = 1$ indicates a_k is executed locally. Because a task can only be executed at one location at a time, there is constraint, i.e., $\sum_{j=0}^{M} \lambda_{i,j,a_k} = 1$. Service migration is an operation that the service instance of UEs running at the current MEC server is migrated to the another MEC server, and provides seamless service for UEs. $v_{i,j',j,a_k} \in \{0,1\}$ represents the service migration decision, where $j' = I_{i,a_{k-1}}$. If $v_{i,j',j,a_k} = 1$, then VM$_i$ will be migrated from MEC$_{j'}$ to MEC$_j$. It should be noted that $v_{i,j,j,a_k} = 0$. VM$_i$ can only be deployed at an MEC server at a time, thus $\sum_{j=1}^{M} v_{i,j',j,a_k} \leq 1$.

In the scenario, we assume that all service instances can execute the task immediately after the migration. Meanwhile, we ignore the situation of service migration failure, i.e., service migration is always successful.

B. Communication Model

The maximum transmission power of UE$_i$ is $p_{i,max}$ Watt (W). In this paper, we use the Rayleigh fading channel model [19] and ignore the overhead of result data receiving. The rate of UE$_i$ to transmit a_k to MEC$_j$ is $R_{i,j,a_k} = W_i \log_2(1 + p_{i,j,a_k} h_i^2 / d_{i,j,a_k}^{\omega_i} N_i)$, where $W_i, p_{i,j,a_k}, h_i, d_{i,j,a_k}, \omega_i, N_i$ are the transmission channel bandwidth, the transmission power of UE$_i$ to upload a_k to MEC$_j$, the transmission channel fading coefficient, the distance between UE$_i$ and MEC$_j$ when a_k is executed, the channel path loss exponent, and the channel white Guassian noise, respectively.

C. Computation Model

1) Local computation model: We assume that UE$_i$ has a maximum CPU frequency represented by $f_{i,max}$ (cycles per second) and the CPU is equipped with the technique of dynamic voltage and frequency scaling (DVFS) such that can adjust the frequency of CPU. We use $0 \le f_{i,a_k} \le f_{i,max}$ to represent the actual CPU frequency of UE$_i$ when the UE executes a_k. The local computational time of a_k is $T^l_{i,a_k} = w_{i,a_k}/f_{i,a_k}$. According to [20], the local energy consumption of a_k is $E^l_{i,a_k} = w_{i,a_k}\kappa_i f^2_{i,a_k}$, where κ_i is the coefficient factor of UE$_i$'s chip architecture.

2) MEC server computation model: The computing power of an MEC server is denoted as f_j (cycles per second). The computational time of $a_k \in A_i$ executed at MEC$_j$ is $T^j_{i,a_k} = t_{i,j,t,a_k} + t_{i,j,e,a_k} + t_{i,j,w,a_k} + v_{i,j',j,a_k} t_{i,j,m,a_k}$, where $t_{i,j,e,a_k} = w_{i,a_k}/f_j$, $t_{i,j,t,a_k} = w_{i,a_k}\delta_{i,a_k}/R_{i,j,a_k}$, $t_{i,j,m,a_k} = \alpha_i \sum_{k=1}^{|A_i|} w_{i,a_k}\delta_{i,a_k}$, t_{i,j,w,a_k} are the computational time of a_k executed by MEC$_j$, the transmission delay of a_k, the migration delay of VM$_i$, the average waiting delay if a_k is executed by MEC$_j$, respectively. It should be noted that α_i is the migration delay impact factor of the workload and processing density for UE$_i$'s application, and can be gotten from [21]. As shown in the equation, the service migration overhead is introduced when $v_{i,j',j,a_k} = 1$. Accordingly, the energy consumption of a_k executed by MEC$_j$ is $E^j_{i,a_k} = p_{i,0}(t_{i,j,e,a_k} + t_{i,j,w,a_k} + v_{i,j',j,a_k} t_{i,j,m,a_k}) + p_{i,j,a_k} t_{i,j,t,a_k}$, where $p_{i,0}$ (W) is the idle power of UE$_i$.

D. Problem Formulation

1) Energy Minimization Problem: Based on the above definitions, the energy consumption of UE$_i$ is formulated as

$$E_i = \sum_{k=1}^{|A_i|} \left(\sum_{j=1}^{M} \lambda_{i,j,a_k} E^j_{i,a_k} + \Big(1 - \sum_{j=1}^{M} \lambda_{i,j,a_k}\Big) E^l_{i,a_k} \right). \quad (1)$$

The energy minimization problem of UE$_i$ is formulated as

$$\begin{aligned} \text{P1}: \quad & \min_{V_i,\Lambda_i,P_i,F_i} E_i, && (2)\\ s.t. \quad & C_1 : 0 \le f_{i,a_k} \le f_{i,max}, \\ & C_2 : 0 \le p_{i,j,a_k} \le p_{i,max}, \\ & C_3 : \sum_{j=0}^{M} \lambda_{i,j,a_k} = 1, \lambda_{i,j,a_k} \in \{0,1\}, \\ & C_4 : \sum_{j \ne j'}^{M} v_{i,j',j,a_k} \le 1, v_{i,j',j,a_k} \in \{0,1\}, \\ & C_5 : T_i \le T_{i,max}, \end{aligned}$$

where V_i, Λ_i, P_i, F_i are the service migration decision set of A_i, offloading decision set of A_i, the transmission power strategy set of A_i, and the CPU frequency strategy set of A_i. In P1, C_1, C_2 are the maximum CPU frequency constraint and maximum transmission power constraint, respectively. C_3 indicates that a task can only be executed at one location at a time. C_4 indicates that VM$_i$ can only be migrated to one MEC server at a time. C_5 is the computational time constraint of the application.

2) Latency Minimization Problem: The latency of an application generated by UE$_i$ is formulated as

$$T_i = \sum_{k=1}^{|V_i|} \left(\sum_{j=1}^{M} \lambda_{i,j,a_k} T^j_{i,a_k} + \Big(1 - \sum_{j=1}^{M} \lambda_{i,j,a_k}\Big) T^l_{i,a_k} \right). \quad (3)$$

The latency minimization problem of UE$_i$ is formulated as

$$\begin{aligned} \text{P2}: \quad & \min_{V_i,\Lambda_i,P_i,F_i} T_i, && (4)\\ s.t. \quad & C_1, C_2, C_3, C_4, \\ & C_6 : E_i \le E_{i,max}. \end{aligned}$$

In P2, C_6 is the energy constraint of the application. As shown in P1 and P2, we can know that the two problems are mixed integer programming problems and NP-hard problems [2].

III. Task Offloading, Resource Allocation, and Service Migration Algorithm

A. Energy Minimization

When the deployment policy of VM$_i$ is determined in advance, we can relax λ_{i,j,a_k} to be a continuous variable, i.e., $0 \le \lambda_{i,j,a_k} \le 1$, so that the original problems will be transformed to standard linear programming problems. Therefore, we can solve the two problems by transforming them to 1-dimensional optimization problems. The basis theorem of the method is as follows [22].

Theorem 1 $\inf_{\beta,\sigma} f(\beta,\sigma) = \inf_{\sigma} \tilde{f}(\sigma)$, *where* $\tilde{f}(\sigma) = \inf_{\beta} f(\beta,\sigma)$.

For a task, if $I_{i,a_k} = j$, the subproblem of P1 can be formulated as

$$\begin{aligned} \text{P3}: \quad & \min_{\boldsymbol{s_{i,a_k}}} E_{i,a_k}, && (5)\\ s.t. \quad & C_1, C_2, \\ & C_7 : \sum_{j=0}^{M} \lambda_{i,j,a_k} = 1, \lambda_{i,j,a_k} \in [0,1], \\ & C_8 : T_{i,a_k} \le t_{i,r,a_k}, && (6) \end{aligned}$$

where $E_{i,a_k} = E^l_{i,a_k} + \lambda_{i,j,a_k}(E^j_{i,a_k} - E^l_{i,a_k})$, $T_{i,a_k} = \lambda_{i,j,a_k} T^j_{i,a_k} + (1 - \lambda_{i,j,a_k}) T^l_{i,a_k}$, and $t_{i,r,a_k} = w_{i,a_k} T_{i,max} / \sum_{k=1}^{|A_i|} w_{i,a_k}$ is the maximum latency time of a_k. In P3, $\boldsymbol{s_{i,a_k}}$ represents a task offloading strategy, including offloading decision λ_{i,j,a_k}, CPU frequency f_{i,a_k}, and transmission power p_{i,j,a_k}. C_8 is the computational time constraint of the task. The subproblem P3 can be further decomposed into two subproblems based on λ_{i,j,a_k}. With the help of Theorem 1, we can first make a decision to decide the execution location of a task.

1) Offloading decision: It is easy to know that E_{i,a_k} is a linear function w.r.t. λ_{i,j,a_k}. The offloading decision can be obtained from $\lambda_{i,j,a_k} = \Phi(E^l_{i,a_k} > E^j_{i,a_k})$, where $\Phi(o) \in \{0,1\}$ is a boolean function. If o is true, then $\Phi(o) = 1$. Otherwise, $\Phi(o) = 0$.

2) CPU frequency strategy: If $\lambda_{i,0,a_k} = 1$, the task is executed locally. According to C_8, we have $f_{i,a_k} \geq \tilde{f}_{i,a_k}$, where $\tilde{f}_{i,a_k} = \sum_{k=1}^{|A_i|} w_{i,a_k}/T_{i,max}$. And according to C_1, the optimal CPU frequency can be obtained from $f^*_{i,v_a} = \min\{f_{i,max}, \tilde{f}_{i,a_k}\}$.

3) Transmission power strategy: If $\lambda_{i,j,a_k} = 1$, the task is executed remotely. According to C_8, we have $p_{i,j,a_k} \geq (2^{\tau_{i,j,a_k}} - 1)/z_{i,j,a_k}$, where $\tau_{i,j,a_k} = w_{i,a_k}\delta_{i,a_k}/W_i(t_{i,r,a_k} - t_{i,j,e,a_k} - t_{i,j,w,a_k} - v_{i,j',j,a_k}t_{i,j,m,a_k})$, and $z_{i,j,a_k} = h_i^2/(d^{\omega_i}_{i,j,a_k}N_i)$. Thus, $p_{i,j,min,a_k} = (2^{\tau_{i,j,a_k}} - 1)/z_{i,j,a_k}$ is the minimal transmission power for transmitting a_k within the limited delay. Based on C_2, the optimal transmission power of a_k is $p^*_{i,j,a_k} = \min\{p_{i,max}, p_{i,j,min,a_k}\}$. Accordingly, we have the following corollary.

Corollary 1 *For $a_k \in A_i$ and MEC$_j$, if $p_{i,j,min,a_k} \leq p_{i,max}$, then the server is an available server for UE$_i$.*

Proof. If $p_{i,j,min,a_k} > p_{i,max}$, then the server can not complete the task within the limited delay. □

Corollary 1 checks the feasibility of an MEC server to execute a task. Thus, for $a_k \in A_i$, UE$_i$ can first determine a set of available MEC servers ($\mathbf{M}_{\boldsymbol{i,a_k}}$) to reduce the complexity of making strategy.

4) Service migration decision: Service migration introduces additional overhead for UEs. But if migration gain is bigger than migration cost, the migration can be initiated. Thus, the service migration decision can be obtained from $v_{i,j',j,a_k} = \Phi(E^{j'}_{i,a_k} > E^{j}_{i,a_k})$.

Algorithm 1 EO-RM

Input: $A_i, T_{i,max}, f_{i,max}, p_{i,max}, W_i, h_i, N_i, \omega_i, w_i, \delta_i, I_{i,a_0}$.
Output: Λ_i^*, F_i^*, P_i^*, V_i^*, and I_i^*.

```
1:  for $a_k \in A_i$ do
2:      Obtain $\mathbf{M}_{\boldsymbol{i,a_k}}$ from Corollary 1;
3:      Calculate $f^*_{i,a_k}$, $p_{i,j',a_k}$, $E^l_{i,a_k}$ and $E^j_{i,a_k}$;
4:      $E^m_{i,a_k} \leftarrow E^j_{i,a_k}$;
5:      for MEC$_j \in \mathbf{M}_{\boldsymbol{i,a_k}}$ do
6:          Calculate $p_{i,j,a_k}$ and $E^j_{i,a_k}$;
7:          if $E^j_{i,a_k} < E^m_{i,a_k}$ then
8:              $v^*_{i,I_{i,a_{k-1}},j,a_k} \leftarrow 1$;
9:              $I^*_{i,a_k} \leftarrow j$;
10:             $E^m_{i,a_k} \leftarrow E^j_{i,a_k}$;
11:         end if
12:     end for
13:     Update $\lambda^*_{i,I_{i,a_k},a_k}$ and $p^*_{i,I_{i,a_k},a_k}$;
14: end for
15: return $\Lambda_i^*$, $F_i^*$, $P_i^*$, $V_i^*$, and $I_i^*$.
```

Based on the above, we can get the task offloading strategy and the corresponding overhead for executing the task at each MEC server (MEC$_j \in \mathbf{M}_{\boldsymbol{i,a_k}}$). Then, the server MEC$_{j^*}$ with minimal overhead is the optimal server. If $I_{i,a_k} \neq j^*$, the service migration is triggered. Since the mobility type of UE affects the task offloading and service migration strategies, we can further to minimize the overhead according to the mobility characteristics. Next, we detail the task offloading and service migration algorithms for UEs with different mobility types.

5) The algorithm for UEs with random mobility: For UE$_i$ with random mobility, the strategy can only be made based on the current state of the UE. Algorithm 1 shows the process of making energy-optimal offloading and service migration strategies for UE$_i$ with random mobility. For given a_k and MEC$_{j'}$, f_{i,a_k}, p_{i,j',a_k}, E^l_{i,a_k}, and E^j_{i,a_k} can be obtained accordingly. Then, UE$_i$ iterates $\mathbf{M}_{\boldsymbol{i,a_k}}$ to try to find the optimal MEC server MEC$_{j^*}$ while making the service migration decision. The complexity of the algorithm is $O\big(\sum_{k=1}^{|A_i|} |\mathbf{M}_{\boldsymbol{i,a_k}}|\big)$.

Algorithm 2 EO-PM

Input: $A_i, T_{i,max}, f_{i,max}, p_{i,max}, W_i, h_i, N_i, \omega_i, w_i, \delta_i, I_{i,a_0}$.
Output: Λ_i^*, F_i^*, P_i^*, V_i^*, and I_i^*.

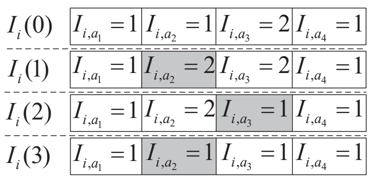
```
1:  for $a_k, a_{k+1} \in A_i$ do
2:      Obtain $\mathbf{M}_{\boldsymbol{i,a_k}}$ and $\mathbf{M}_{\boldsymbol{i,a_{k+1}}}$ from Corollary 1;
3:      Obtain $\boldsymbol{s_{i,a_k}}$, $\boldsymbol{s_{i,a_{k+1}}}$, $I_{i,a_k} = j_k$, and $I_{i,a_{k+1}} = j_{k+1}$
        from Algorithm 1;
4:      if $\lambda_{i,j_k,a_k} = \lambda_{i,j_{k+1},a_{k+1}} = 1$ and $j_k \neq j_{k+1}$ then
5:          for MEC$_j \in \mathbf{M}_{\boldsymbol{i,a_k}} \cap \mathbf{M}_{\boldsymbol{i,a_{k+1}}}$ do
6:              if $E^j_{i,a_k} + E^j_{i,a_{k+1}} < E^{j_k}_{i,a_k} + E^{j_{k+1}}_{i,a_k}$, $T^j_{i,a_k} \leq t_{i,r,a_k}$
                and $T^j_{i,a_{k+1}} \leq t_{i,r,a_{k+1}}$ then
7:                  $I_{i,a_k} \leftarrow j$;
8:                  $I_{i,a_{k+1}} \leftarrow j$;
9:                  Update $\boldsymbol{s_{i,a_k}}$, $\boldsymbol{s_{i,a_{k+1}}}$, $v_{i,j',j,a_k}$, and
                    $v_{i,j_k,j_{k+1},a_{k+1}}$;
10:             end if
11:         end for
12:     end if
13:     Obtain $\boldsymbol{s^*_{i,a_k}}$, $\boldsymbol{s^*_{i,a_{k+1}}}$, $v^*_{i,j',j,a_k}$, and $v^*_{i,j_k,j_{k+1},a_{k+1}}$;
14:     $k \leftarrow k + 2$;
15: end for
16: return $\Lambda_i^*$, $F_i^*$, $P_i^*$, $V_i^*$ and $I_i^*$.
```

6) The algorithm for UEs with short-term predictable mobility: For UE$_i$ with short-term predictable mobility, the strategy should be made not only based on the current location of UE$_i$, but also the future location of the UE. In this paper, we assume that we can predict the location of a_{k+1} when a_k is executed [18]. Thus, the problem can be formulated as

$$\begin{aligned} \text{P4}: \quad & \min_{\boldsymbol{s_{i,a_k}}, \boldsymbol{s_{i,a_{k+1}}}} \quad E_{i,a_k} + E_{i,a_{k+1}}, \\ & s.t. \quad C_1, C_2, C_7, C_8. \end{aligned} \tag{7}$$

Let $I_{i,a_k} = j_k$ and $I_{i,a_{k+1}} = j_{k+1}$ be the VM deployment polices for a_k and a_{k+1} obtained from Algorithm 1 respectively. And then, we try to find a common MEC server to execute a_k and a_{k+1}, and adjust the joint strategy for the two tasks, with the rational revealed by the following theorem.

Theorem 2 *If $j_k \neq j_{k+1}$ and $\lambda_{i,j_k,a_k} = \lambda_{i,j_{k+1},a_{k+1}} = 1$, there may be a new optimal strategy for two tasks, i.e., $I'_{i,a_k} =$*



$I_i(0)$	$I_{i,a_1}=1$	$I_{i,a_2}=1$	$I_{i,a_3}=2$	$I_{i,a_4}=1$
$I_i(1)$	$I_{i,a_1}=1$	$I_{i,a_2}=2$	$I_{i,a_3}=2$	$I_{i,a_4}=1$
$I_i(2)$	$I_{i,a_1}=1$	$I_{i,a_2}=2$	$I_{i,a_3}=1$	$I_{i,a_4}=1$
$I_i(3)$	$I_{i,a_1}=1$	$I_{i,a_2}=1$	$I_{i,a_3}=1$	$I_{i,a_4}=1$

Fig. 2. An illustration process of Algorithm 3.

$I^{'}_{i,a_{k+1}} = j^*$ and $\lambda_{i,j^*,a_k} = \lambda_{i,j^*,a_{k+1}} = 1$, where MEC$_{j^*} \in \mathbf{M}_{i,a_k} \cap \mathbf{M}_{i,a_{k+1}}$. Otherwise, the original strategies are the optimal strategies for a_k and a_{k+1}.

Proof. If $I_{i,a_k} \neq I_{i,a_{k+1}}$, $\lambda_{i,j_k,a_k} = \lambda_{i,j_{k+1},a_{k+1}} = 1$, and there is a new optimal strategy for the two tasks, i.e., $\lambda_{i,I^{'}_{i,a_k},a_k} = \lambda_{i,I^{'}_{i,a_{k+1}},a_{k+1}} = 1$, where $I^{'}_{i,a_k} \neq I^{'}_{i,a_{k+1}}$, we have an inequality, i.e., $E^{j_k}_{i,a_k} + E^{j_{k+1}}_{i,a_{k+1}} > E^{j^{'}_k}_{i,a_k} + E^{j^{'}_{k+1}}_{i,a_{k+1}}$, where MEC$_{j^{'}_k} \in \mathbf{M}_{i,a_k}$ and MEC$_{j^{'}_{k+1}} \in \mathbf{M}_{i,a_{k+1}}$. However, we know that $E^{j_k}_{i,a_k} \le E^{j^{'}_k}_{i,a_k}$ and $E^{j_{k+1}}_{i,a_{k+1}} \le E^{j^{'}_{k+1}}_{i,a_{k+1}}$, which contradicts with the premise. Therefore, if there is a new strategy for the 2 tasks, $I^{'}_{i,a_k} = I^{'}_{i,a_{k+1}} = j^*$ should be true (i.e., the two tasks are executed at a common MEC server MEC$_{j^*}$). Moreover, if MEC$_{j^*} \notin \mathbf{M}_{i,a_k} \cap \mathbf{M}_{i,a_{k+1}}$, which means that one of the tasks can not be completed within the limited delay. Therefore, if $I_{i,a_k} = I_{i,a_{k+1}}$ and $\lambda_{i,j_k,a_k} = \lambda_{i,j_{k+1},a_{k+1}} = 1$, then MEC$_{I_{i,a_k}}$ is the optimal execution location for two tasks.

If $\lambda_{i,j_k,a_k} \neq \lambda_{i,j_{k+1},a_{k+1}}$, we assume that $\lambda_{i,0,a_k} = 1$ and $\lambda_{i,j_{k+1},a_{k+1}} = 1$, we have inequalities $E^{l}_{i,a_k} \le E^{j}_{i,a_k}$ ($\forall$MEC$_j \in \mathbf{M}_{i,a_k}$) and $E^{j_{k+1}}_{i,a_k} \le E^{j}_{i,a_k}$ ($\forall$MEC$_j \in \mathbf{M}_{i,a_{k+1}}$). Thus, there must be no MEC server that can further reduce the overhead of the 2 tasks. Otherwise, we have $E^{l}_{i,a_k} + E^{j_{k+1}}_{i,a_k} > E^{I^{'}_{i,a_k}}_{i,a_k} + E^{I^{'}_{i,a_{k+1}}}_{i,a_k}$, which contradicts with the premise. Therefore, we have the conclusion. $\square$

Algorithm 2 shows the process of making energy-optimal offloading and service migration strategies for UE$_i$ with short-term predictable mobility. The optimal strategies of a_k and a_{k+1} can be obtained from Algorithm 1, respectively. Then, the service is rescheduled in MEC$_j \in \mathbf{M}_{i,a_k} \cap \mathbf{M}_{i,a_{k+1}}$ according to Theorem 2. The complexity of the algorithm is $O\big((\sum_{k=1}^{|A_i|} |\mathbf{M}_{i,a_k}| + |\mathbf{M}_{i,a_{k+1}}|)/2 + |\mathbf{M}_{i,a_k} \cap \mathbf{M}_{i,a_{k+1}}|\big)$.

7) The algorithm for UEs with fully known mobility: For UE$_i$ with fully known mobility, we can make the strategy based on the whole process of the application execution. Algorithm 3 shows the process of making energy-optimal offloading and service migration strategies for UE$_i$ with fully known mobility. Figure 2 illustrates an example process of Algorithm 3. We first obtain initial Λ_i, F_i, P_i, V_i, and I_i through Algorithm 2. As shown in Figure 2, $I_i(0)$ represents the initial service deployment strategy of UE$_i$.

Algorithm 3 has two iteration processes for updating service migration strategy. In the first iteration process (lines 4-25), we readjust the strategy with an early service migration strategy, that is we attempt toward reduce the energy consumption

Algorithm 3 EO-FM

Input: $A_i, T_{i,max}, f_{i,max}, p_{i,max}, W_i, h_i, N_i, \omega_i, w_i, \delta_i, I_{i,a_0}, \epsilon_i, \Gamma_i$.
Output: Λ^*_i, F^*_i, P^*_i, V^*_i, and I^*_i.

```
1: Obtain $\Lambda_i$, $F_i$, $P_i$, $V_i$, and $I_i$ through Algorithm 2;
2: $\Sigma_t \leftarrow 0$;
3: $\gamma \leftarrow 0$;
4: while $\sum_{k=1}^{|A_i|} E_{i,k} - \Sigma_t > \epsilon_i$ and $\gamma < \Gamma_i$ do
5:     $\phi \leftarrow I_{i,a_1}$;
6:     $\rho \leftarrow 1$;
7:     $\gamma \leftarrow \gamma + 1$;
8:     for $a_k \in A_i$ do
9:         if $I_{i,a_k} \neq \phi$ and $k - \rho > 1$ then
10:            $k^{'} \leftarrow k - 1$;
11:            while $\rho < k^{'}$ do
12:                if MEC$_{I_{i,a_k}} \in \mathbf{M}_{i,a_{k^{'}}}$ then
13:                    if $E^{I_{i,a_k}}_{i,a_{k^{'}}} + \sum_{\xi=k^{'}+1}^{k} E_{i,\xi} < \sum_{\xi=k^{'}}^{k} E_{i,\xi}$ and
                       $T^{I_{i,a_k}}_{i,a_{k^{'}}} \le t_{i,r,a_{k^{'}}}$ then
14:                        $I_{i,a_{k^{'}}} \leftarrow I_{i,a_k}$;
15:                        Update $\boldsymbol{s_{i,a_k}}$ and $v_{i,j^{'},j,a_k}$
16:                    end if
17:                end if
18:                $k^{'} \leftarrow k^{'} - 1$;
19:            end while
20:            $\phi \leftarrow I_{i,a_{k^{'}+1}}$;
21:            $\rho \leftarrow k^{'} + 1$;
22:        end if
23:    end for
24:    $\Sigma_t \leftarrow \sum_{k=1}^{|A_i|} E_{i,k}$;
25: end while
26: for $a_k \in A_i$ do
27:     if $I_{i,a_k} \neq I_{i,a_{k+1}}$ and $I_{i,a_k} = I_{i,a_{k+2}}$ and MEC$_{I_{i,a_k}} \in$
        $\mathbf{M}_{i,a_{k+1}}$ then
28:         if $E^{I_{i,a_k}}_{i,a_k} + E^{I_{i,a_k}}_{i,a_{k+1}} + E^{I_{i,a_k}}_{i,a_{k+2}} < \sum_{k^{'}=k}^{k+2} E_{i,k^{'}}$ and
            $T^{I_{i,a_k}}_{i,a_{k+1}} < t_{i,r,a_{k+1}}$ then
29:             $I_{i,a_{k+1}} \leftarrow I_{i,a_k}$;
30:             Update $\boldsymbol{s_{i,a_{k+1}}}$ and service migration strategy;
31:         end if
32:     end if
33: end for
34: return $\Lambda^*_i$, $F^*_i$, $P^*_i$, $V^*_i$, and $I^*_i$.
```

by migrating service ahead. In the algorithm, $k - \rho - 1$ represents the number of tasks between a_ρ and a_k, where a_ρ is the first task executed by VM$_i$ when the service is deployed at MEC$_{I_{i,a_\rho}}$, and a_k is the first task executed by VM$_i$ after the first VM$_i$ migration in the execution process of a_ρ's successor tasks. If $k - \rho - 1 > 0$, the update process will be initiated. As shown in Figure 2, if VM$_i$ is migrated to MEC$_2$ in advance, we can update the strategy when the new energy consumption summation $\sum_{\xi=1}^{3} E_{i,\xi}$ is less than the original energy consumption summation. Then, we get

$I_i(1)$. The iteration will be repeated until the reduced energy consumption is less than ϵ_i or the number of iterations exceeds Γ_i. Then, we get $I_i(2)$.

In the second iteration process, we reduce the overhead by avoiding service migration (lines 26-33). If $I_{i,a_k} \neq I_{i,a_{k+1}}$, $I_{i,a_k} = I_{i,a_{k+2}}$ and $\text{MEC}_{I_{i,a_k}} \in \mathbf{M}_{\boldsymbol{i,a_{k+1}}}$, we consider reducing $\sum_{k'=k}^{k+2} E_{i,k'}$ by avoiding service migration when a_{k+1} is executed. If the new energy consumption summation is less than the original, we can update the task offloading and service migration strategies. It should be noted that the constraint of delay must be satisfied during the update process. Finally, the optimal strategy $I_i(3)$ can be obtained. The complexity of the algorithm is $O\big((\sum_{k=1}^{|A_i|} |\mathbf{M}_{\boldsymbol{i,a_k}}| + |\mathbf{M}_{\boldsymbol{i,a_{k+1}}}|)/2 + |\mathbf{M}_{\boldsymbol{i,a_k}} \cap \mathbf{M}_{\boldsymbol{i,a_{k+1}}}| + \Gamma_i|A_i|^2\big)$.

B. Latency Minimization

For a task, if $I_{i,a_k} = j$, the subproblem of P2 can be formulated as

$$\begin{aligned} \text{P5}: \quad & \min_{\boldsymbol{s_{i,a_k}}} T_{i,a_k}, \\ s.t. \quad & C_1, C_2, C_7, \\ & C_9: E_{i,a_k} \leq e_{i,r,a_k}, \end{aligned} \tag{8}$$

where $e_{i,r,a_k} = w_{i,a_k} E_{i,max} / \sum_{k=1}^{|A_i|} w_{i,a_k}$ is the maximum energy consumption of a_k. Similar to P3, P5 can also be further decomposed into two subproblems based on λ_{i,j,a_k}.

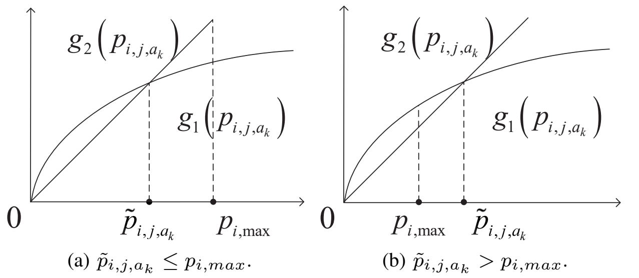
1) The optimal solution of the latency minimization problem: The offloading decision of P5 can be made by $\lambda_{i,j,a_k} = \Phi(T_{i,a_k}^{l} > T_{i,a_k}^{j})$. If $\lambda_{i,0,a_k} = 1$, since $f_{i,a_k}^2 \kappa_i w_{i,a_k} \leq e_{i,r,a_k}$ must be satisfied for local execution, the optimal CPU frequency is $f_{i,v_a}^* = \min\Big(f_{i,max}, \sqrt{E_i / \sum_{k=1}^{|A_i|} w_{i,a_k}\kappa_i}\Big)$.

Accordingly, if $\lambda_{i,j,a_k} = 1$, the transmission power optimization must be satisfied to $\pi_{i,j,a_k} \log_2(1 + p_{i,j,a_k} z_{i,j,a_k}) \geq p_{i,j,a_k}$, where $\pi_{i,j,a_k} = W_i\big(e_{i,r,a_k} - p_{i,0}(t_{i,j,e,a_k} + t_{i,j,w,a_k} + v_{i,j',j,a_k} t_{i,j,m,a_k})\big)/w_{i,a_k}\delta_{i,a_k}$. Moreover, we introduce $g(p_{i,j,a_k}) = g_1(p_{i,j,a_k}) - g_2(p_{i,j,a_k})$, where $g_1(p_{i,j,a_k}) = \pi_{i,j,a_k} \log_2(1 + p_{i,j,a_k} z_{i,j,a_k})$ and $g_2(p_{i,j,a_k}) = p_{i,j,a_k}$. The optimal transmission power is illustrated in Figure 3. If $g_1(p_{i,max}) > g_2(p_{i,max})$, $p_{i,j,a_k}^* = p_{i,max}$. Otherwise, $p_{i,j,a_k}^* = \tilde{p}_{i,j,a_k}$, where $\tilde{p}_{i,j,a_k}$ is the solution of $g(p_{i,j,a_k}) = 0$. $\tilde{p}_{i,j,a_k}$ can be obtained with the help of the following theorem.

Theorem 3 *$g(p_{i,j,a_k})$ is a monotonic non-increasing function w.r.t. $p_{i,j,a_k} \geq (\pi_{i,j,a_k} z_{i,j,a_k}/\ln 2 - 1)/z_{i,j,a_k}$.*

Proof. Because $g'(p_{i,j,a_k}) = \pi_{i,j,a_k} z_{i,j,a_k} / \big(\ln 2(1 + p_{i,j,a_k} z_{i,j,a_k})\big) - 1$, thus if $p_{i,j,a_k} \geq (\pi_{i,j,a_k} z_{i,j,a_k}/\ln 2 - 1)/z_{i,j,a_k}$, then $g'(p_{i,j,a_k}) \leq 0$ and $g(p_{i,j,a_k})$ is a monotonic non-increasing function w.r.t. p_{i,j,a_k}. □

Theorem 3 reveals that we can use binary search method to find $\tilde{p}_{i,j,a_k}$ [22]. And the optimal transmission power can be obtained from $p_{i,j,a_k}^* = \min\{\tilde{p}_{i,j,a_k}, p_{i,max}\}$. Due to the space limitation, the binary search algorithm is omitted. It is easy to know that the complexity of the binary search algorithm is $O(U_{i,a_k})$, where U_{i,a_k} is the maximal number of iterations.



(a) $\tilde{p}_{i,j,a_k} \leq p_{i,max}$. (b) $\tilde{p}_{i,j,a_k} > p_{i,max}$.

Fig. 3. The illustrations of p_{i,j,a_k}^*.

Similarity, the service migration decision can be obtained from $v_{i,j',j,a_k} = \Phi(T_{i,a_k}^{j'} > T_{i,a_k}^{j})$.

2) The algorithms for UEs with different mobility types: For UEs with different mobility types, we can use Algorithms 1, 2, and 3 to make the strategy. However, since the UE regards the latency as the optimization objective, the overhead used in original algorithms (i.e., E_{i,a_k}) should be replaced with T_{i,a_k}. The algorithms are called LO-RM, LO-PM, and LO-FM, respectively.

IV. Simulation Experiments and Results Analysis

A. Experiment Setting

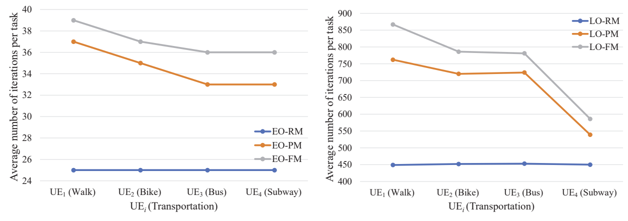
In the experiment, there are $N = 4$ UEs and $M = 24$ MEC servers. Referring to [7], the parameters of UE$_i$ are given as follows: $p_{i,max} = 5$ W, $p_{i,0} = 0.01$ W, $f_{i,max} = 8 \times 10^7$, $T_{i,max} = 0.5$ s, $E_{i,max} = 10^{-5}$ J, $\delta_{i,a_k} = 1$ KB/cycle, $\kappa_i = 10^{-14}$, $W_i = 10^{11}$ HZ, $h_i = 10^{-3}$, $\omega_i = 1.5$, $N_i = 10^{-9}$, and $\alpha_i = 10^{-8}$. The computing power of MEC$_j$ is given as $f_j = 5 + 0.1j \times 10^{10+0.1j}$. Without loss of generality, we set $t_{i,j,w,a_k} = 0$. We also assume that UE$_i$ has a task set, such as A_i={10, 12, 14, 16, 18, 20, 22, 24, 26, 30, 32, 34, 36, 18, 20, 22, 10, 12, 14, 16}, where the element of A_i indicates the workload of a task. In addition, we use the GPS trajectory dataset Geolife [23] to represent the movement trajectory of UE. In the dataset, the movement trajectory of UE is represented by a sequence of time-stamped points (2 seconds apart), each of which contains the latitude and longitude of the UE. We select four movement trajectories with different means of transportation (i.e., walk, bike, bus, and subway) for the four UEs on the same road in Beijing. We assume that the MEC servers have been deployed on the road. Each movement trajectory of UEs contains 20 location points.

B. The Convergence of the Algorithms

As shown in Fig. 4(a), for energy-optimal algorithm, the average number of iterations of task is less than 40. since the transmission power is obtained from the binary search algorithm, the average number of iterations of task is more than what the energy-optimal algorithm needs. As shown in Fig. 4(b), for latency-optimal algorithm, the average number of iterations of task is less than 870. As shown in the figures, for different mobility types and transportation means, the algorithms can converge.

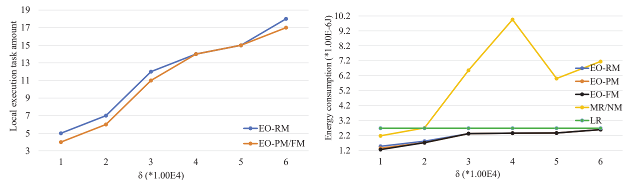
TABLE I
THE OVERHEAD COMPARISON OF UE$_i$ USING DIFFERENT SCHEMES/ALGORITHMS

Schemes /Algorithms	**Energy consumption (J)**				**Latency time (s)**			
	UE$_1$ **(Walk)**	UE$_2$ **(Bike)**	UE$_3$ **(Bus)**	UE$_4$ **(Subway)**	UE$_1$ **(Walk)**	UE$_2$ **(Bike)**	UE$_3$ **(Bus)**	UE$_4$ **(Subway)**
MM	2.56E-06	2.42E-06	2.59E-06	2.29E-06	**5.09E-07**	1.32E-06	1.37E-06	1.25E-06
MR	1.94E-07	1.38E-07	1.43E-07	1.97E-07	**5.09E-07**	1.32E-06	1.31E-06	1.25E-06
MP	2.53E-06	2.32E-06	6.79E-06	4.07E-06	4.00E-06	3.94E-06	1.08E-05	5.30E-06
NM	1.94E-07	1.39E-07	1.43E-07	1.97E-07	**5.09E-07**	2.80E-06	2.71E-06	2.77E-06
LM	25,984	25,984	25,984	25,984	\	\	\	\
LR	2.68E-06	2.68E-06	2.68E-06	2.68E-06	0.26	0.26	0.26	0.26
EO/LO-RM	1.94E-07	1.38E-07	1.43E-07	1.97E-07	**5.09E-07**	1.32E-06	1.31E-06	1.25E-06
EO/LO-PM	1.33E-07	**1.07E-07**	1.25E-07	1.51E-07	**5.09E-07**	1.25E-06	1.26E-06	1.15E-06
EO/LO-FM	**1.26E-07**	**1.07E-07**	**1.17E-07**	**1.48E-07**	**5.09E-07**	**9.79E-07**	**1.16E-06**	**1.00E-06**



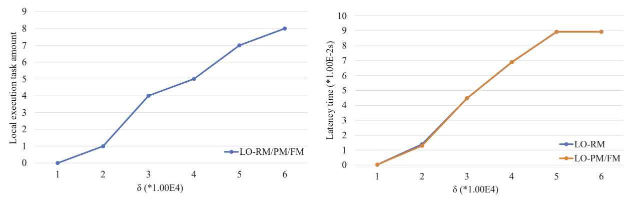
(a) Energy-optimal algorithms. (b) Latency-optimal algorithms.

Fig. 4. The average number of iterations per task for UEs with differen transportation means.



(a) (b)

Fig. 5. (a) The impact of δ on the local execution. (b) The impact of δ on the energy consumption.



(a) (b)

Fig. 6. (a) The impact of δ on the local execution. (b) The impact of δ on the latency time.

C. The Effectiveness of the Algorithms

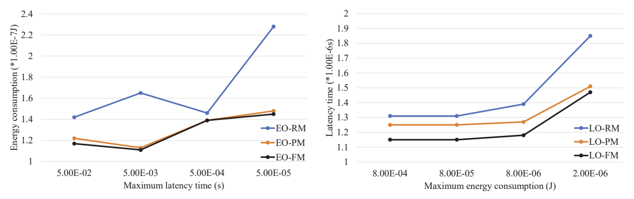
We use the following six task offloading and service migration schemes as baselines to evaluate the effectiveness of the algorithms proposed in this paper: i) All tasks will be executed locally while using $f_{i,max}$. The scheme is marked as LM; ii) All tasks will be executed locally while using DVFS technology to adjust the CPU frequency. We mark this scheme as LR; iii) All tasks will be offloaded to the MEC server for executing. However, UE$_i$ uploads its task using $p_{i,max}$. We mark this scheme as MM; iv) All tasks will be offloaded to the MEC server for executing. However, UE$_i$ can adjust its transmission power. We mark this scheme as MR; v) All tasks can be executed locally or remotely. However, UE$_i$ sends requests only to the most powerful servers among all available MEC servers. The scheme is marked as MP; vi) All tasks can be executed locally or remotely. However, UE$_i$ sends requests only to the server closest to itself among all available MEC servers. The scheme is marked as NM.

Since the energy consumption of the task executed by $f_{i,max}$ exceeds the maximum energy constraint of the latency minimization problem, the result of LM is omitted in Table I. Choosing different transportation means will result in different movement distances and available server sets, thus affecting task offloading and service migration strategies, and the overhead of UE. Therefore, the effectiveness of the algorithms are different. As shown in the tables, if the future location of the UE can be known in advance, the service migration decision can be pre-determined to further minimize the overhead of UE. Thus, the overhead of the EO/LO-FM algorithms is minimal. Experimental results show that compared with the six other strategy schemes, the algorithms proposed in this paper performs better and can further reduce the overhead of UE by using the characteristics of mobility.

D. The Impact of Other Parameters

In the next, we analyze the impact of δ, $T_{i,max}$, and $E_{i,max}$ on the strategies and overhead of UE$_3$.

1) The impact of δ: As can be seen from Figures 5(a) and 6(a), as δ increases, the overhead of UE increases and the UE is more and more inclined to execute the task locally. It should be noted that since some schemes cannot complete the application within the limited energy or latency, the schemes are omitted in the figures. As shown in Fig. 5(b), as δ increases, the three curves (EO-RM/PM/FM) gradually coincide. This is because the increasing of δ reduces the number of service migration operations, and the algorithms EO/LO-PM and EO/LO-FM reduce the overhead of UE by rescheduling the service instance deployment policy, thus affecting the effectiveness of the algorithms. In Fig. 6(b), the three curves (LO-RM/PM/FM) almost coincide. The reason lies in that we assume the high-speed data transmission between the MEC



(a) (b)

Fig. 7. (a) The impact of $T_{i,max}$ on the energy consumption. (b) The impact of $E_{i,max}$ on the latency time.

servers. Therefore, the effectiveness of LO-PM/FM to reduce the overhead is affected.

2) The impact of $T_{i,max}$ and $E_{i,max}$: In Fig. 7(a), we see that the overhead of UE$_3$ when $T_{i,max} = 0.0005$ s is less than the overhead of the UE when $T_{i,max} = 0.005$ s. According to Corollary 1, we know that the $T_{i,max}$ affects $\mathbf{M}_{\boldsymbol{i,a_k}}$, thus also affecting the strategy and overhead of UE. Therefore, the situation is reasonable. From the perspective of the curves in Figures 7(a) and 7(b), as the constraints tighten, the overhead gradually increases.

V. Conclusion

In order to study the execution overhead minimization problems of UE with mobility in MEC, we jointly optimize the task offloading strategy and service migration strategy. We first formulate energy minimization and latency minimization problems respectively and propose three task offloading and service migration strategy algorithms for UEs with different mobility types. Then, we conduct the simulation experiments using the real world data which records the movement trajectory of UE. The convergence of algorithms, the effectiveness of algorithms to reduce the overhead of UEs, and the impact of various key parameters are demonstrated by the experiments.

In the paper, we consider a discrete moving scenario, and study the impact of three simple mobility type on task offloading and VM migration strategies. However, there are many applications need to be executed in continuous time, and more complicated mobility types. Therefore, for the application and mobility type, we should further study the task offloading and service migration strategies optimization to bring the scenario closer to the real world.

Acknowledgments

We would like to express our gratitude to the three anonymous reviewers for their comments on improving the quality of the manuscript. This work was partially supported by the National Outstanding Youth Science Program of National Natural Science Foundation of China (Grant No. 61625202), and the Program of National Natural Science Foundation of China (Grant Nos. 61876061, 61572176).

References

[1] ETSI, "New white paper: Etsi's mobile edge computing initiative explained," ETSI, Report, 2015.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, thirdquarter 2017.

[3] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278 – 289, 2018.

[4] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, and Y. Paek, "Techniques to minimize state transfer costs for dynamic execution offloading in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 13, no. 11, pp. 2648–2660, 2014.

[5] C. You and K. Huang, "Exploiting non-causal cpu-state information for energy-efficient mobile cooperative computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4104–4117, June 2018.

[6] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8769–8780, Sep. 2018.

[7] K. Li, "A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing," *IEEE Transactions on Sustainable Computing*, pp. 1–10, 2018.

[8] C. Liu, K. Li, J. Liang, and K. Li, "Cooper-match: Job offloading with a cooperative game for guaranteeing strict deadlines in mec," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[9] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 771–786, April 2019.

[10] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Learning driven computation offloading for asymmetrically informed edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 8, pp. 1802–1815, Aug 2019.

[11] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, 2017.

[12] S. Secci, P. Raad, and P. Gallard, "Linking virtual machine mobility to user mobility," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 927–940, Dec 2016.

[13] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, April 2019.

[14] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, Oct 2018.

[15] Q. Wu, X. Chen, Z. Zhou, and L. Chen, "Mobile social data learning for user-centric location prediction with application in mobile edge service migration," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[16] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *IEEE International Symposium on Personal*, 2016.

[17] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Generation Computer Systems*, vol. 85, pp. 1 – 8, 2018.

[18] F. Yu, H. Chen, and J. Xu, "Dmpo: Dynamic mobility-aware partial offloading in mobile edge computing," *Future Generation Computer Systems*, vol. 89, pp. 722 – 735, 2018.

[19] B. Sklar, "Rayleigh fading channels in mobile digital communication systems. i. characterization," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 136–146, Sep. 1997.

[20] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[21] A. Anand, J. Lakshmi, and S. K. Nandy, "Virtual machine placement optimization supporting performance slas," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 1, Dec 2013, pp. 298–305.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[23] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory," *IEEE Data(base) Engineering Bulletin*, June 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/geolife-a-collaborative-social-networking-service-among-user-location-and-trajectory/