Journal of Parallel and Distributed Computing 150 (2021) 69–84

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Short- and long-term cost and performance optimization for mobile user equipments

Yan Ding [a,b], Kenli Li [a,b,*], Chubo Liu [a,b], Zhuo Tang [a,b], Keqin Li [a,b,c,**]

[a] *College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China*
[b] *National Supercomputing Center in Changsha, Changsha 410082, Hunan, China*
[c] *Department of Computer Science, State University of New York, New Paltz, NY 12561, USA*

ARTICLE INFO

Article history:
Received 24 April 2020
Received in revised form 27 September 2020
Accepted 12 December 2020
Available online 1 January 2021

Keywords:
Greedy strategy
Lyapunov optimization
Mobile edge computing
Mobility characteristic
Task offloading strategy

ABSTRACT

Task offloading strategy optimization in mobile edge computing (MEC) has always been a hot issue. However, the mobility of a user equipment (UE) seriously affects the UE's cost and performance. This paper proposes three mobility types depending on whether the mobility characteristic of a UE is known, and formulates an energy minimization problem and a latency minimization problem to optimize the cost and performance, respectively. We first develop greedy strategy based task offloading algorithms for UEs according to their mobility characteristics. However, accurately obtaining the mobility characteristics of the UEs over a long time in practice is a huge challenge, especially in a highly random environment like the MEC. To address the issue, we use a Lyapunov optimization method to develop the algorithms that do not require any prior knowledge of the mobility characteristics to minimize the long-term energy and latency of UEs. Experimental results show that the greedy strategy based algorithms can optimize the cost and performance of UEs by using their mobility characteristics, and perform better than the Lyapunov optimization based algorithms in a short-term. However, the Lyapunov optimization based algorithms perform better than the greedy strategy based algorithms over a long-term.

1. Introduction

1.1. Motivation

The rapid development of the mobile internet and related hardware has helped the advent of Internet of Things (IoT) era. With these new technologies, some complex applications, such as image recognition, virtual reality, augmented reality, and path navigation [1], can be executed by *user equipments* (UEs), such as mobile phone and other IoT devices. However, due to the limitations of the computing power, storage capacity, and battery life of UEs, these applications are sometimes not efficiently executed by the UEs, thus degrading the quality of experience (QoE) [19]. *Mobile edge computing* (MEC) is expected to emerge as a promising technology to mitigate these conflicts.

MEC is an architecture that provides limited resources, such as computing power, to UEs at the edge of network, thus improving the quality of service (QoS) and QoE. High-speed wireless network technologies implement the instant communication between UEs and MEC servers, reducing the communication delay and reliving the network jitter. The heavy tasks of UEs are uploaded to MEC servers for processing to optimize their cost and performance, i.e., minimize energy and latency (computational time) [2]. A lot of researchers worked on the task offloading strategy optimization problem in MEC [3,5–7,11–13,29,33,34]. However, the above work ignored the impact of UE mobility on the strategy [30]. Moreover, ignoring the UE mobility is not suitable for the real-world scenario [27]. UEs are not always fixed at a certain location, and may be moving [1]. Meanwhile, the mobility of UEs seriously affects the strategy's cost and performance [10]. The long distance between the UEs and MEC servers will significantly reduce QoS and QoE.

However, the optimization problem becomes even harder when it involves the mobility. To provide seamless service for UEs with mobility, the services of the UEs will be migrated among MEC servers to follow their movement. In this paper, the service of a UE refers to the fundamental environment for processing offloaded tasks of the UE, such as Docker [16] and virtual machine [15]. Thus, the process of making an offloading strategy will be more complicated. Intuitively, for service deployment, a service provider of MEC can deploy enough servers to improve QoS and QoE. However, it is impractical in the real world due to

[*] Corresponding authors at: College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China.
[**] Corresponding author.
E-mail addresses: ding@hnu.edu.cn (Y. Ding), lkl@hnu.edu.cn (K. Li), liuchubo@hnu.edu.cn (C. Liu), ztang@hnu.edu.cn (Z. Tang), lik@newpaltz.com (K. Li).

https://doi.org/10.1016/j.jpdc.2020.12.006
0743-7315/© 2020 Elsevier Inc. All rights reserved.

the budget constraint of a service provider. Meanwhile, for saving energy, a UE's service in a server that does not respond to the UE will enter the sleep state. Therefore, after migrating the service, it needs additional waiting time for activating the service from the sleep state to serve the UE. Obviously, this is not suitable for many latency-sensitive tasks. Thus, if the mobility characteristics of UEs can be known in advance, we can deploy service to a UE accordingly in advance, thus improving QoS and QoE.

In addition, the UE mobility types are also diverse. Meanwhile, utilizing the mobility characteristics to optimize the cost and performance of UEs is only feasible in a short time. Accurately obtaining the mobility characteristics of the UEs over a long time in practice is a huge challenge, especially in a highly random environment like the MEC. In this paper, the short- and long-term are relative concepts, and represent the number of task offloading strategies made by UEs.

According to the above discussions, in this work, we investigate the following questions: (1) How to make the task offloading strategy to optimize the cost and performance of UEs with mobility? (2) How to optimize the offloading strategy by using the short-term mobility characteristics of UEs? (3) How to optimize the long-term offloading strategy without prior knowledge of mobility characteristics?

1.2. Our contribution

To address the above issues, we study the problems and optimize the short- and long-term cost and performance of UEs respectively. The main contributions of our work are as follows.

- We first formulate the long-term cost and performance optimization problems, respectively. To deal with the challenge of acquiring UEs' mobility characteristics over a long time, we then use a Lyapunov optimization method to decouple the original problems into two series of real-time optimization subproblems. Thus, the cost and performance of UEs can be optimized based on their current states.
- We develop the algorithms based on the Lyapunov optimization method, which do not require any prior knowledge of the mobility characteristics to optimize the long-term cost and performance of UEs. Meanwhile, the algorithms proposed in this paper not only make the offloading decision, but also decide the resource allocation and service migration strategies.
- Extensive simulation experiments are conducted to evaluate the effectiveness of the algorithms in the short- and long-term. Moreover, we explore the impact of various key parameters on the cost and performance of UEs through the experiments.

The rest of the paper is outlined as follows. Section 2 briefly reviews the related research of task offloading strategy optimization, and highlights the characteristics of this paper. Section 3 demonstrates the system model and problem formulations. Section 4 studies the energy minimization problem and develops the algorithms to optimize the cost of UEs. Section 5 studies the latency minimization problem and develops the algorithms to optimize the performance of UEs. Section 6 conducts the experiments to evaluate the effectiveness of the algorithms. Section 7 concludes this paper.

2. Related work

According to the optimization objective, the research of task offloading optimization can be divided into three categories, i.e., energy-optimal (EO), latency-optimal (LO), and others. EO focuses on the energy consumption or harvesting optimization problems [11], and has been extensively studied. For example, Li [13] formulated UEs and MEC servers as queueing models, and developed algorithms by using the Lagrange multiplier method to minimize the energy consumption of UEs. Chen et al. [7] developed an approach to determining how much energy should be harvested at UEs. Cao et al. [5] maximized the saving energy of UEs while satisfying the UEs' latency requirements. Tout et al. [29] proposed a centralized selective and multi-objective algorithm to optimize the energy consumption of UEs. LO investigates the latency minimization problems [13]. For example, Yang et al. [33] minimized the average computation time of UEs through a heuristic algorithm. Li [12] developed a non-cooperative game theoretic algorithm to optimize the latency of UEs. The third category studies the optimization of other objectives. For example, Chen et al. [6] minimized the weighted sum of the energy consumption and computational time for multiple users with multiple wireless channels. Bhattacharya et al. [3] studied QoE improvement from four aspects, including completion time, energy consumption, monetary cost, and security. Yang et al. [34] compressed the transferred data size to reduce the transmission cost during the task offloading process.

Although the above work studied the computation offloading strategy optimization problem from different optimization goals, they assumed that UEs remain stationary and ignored the impact of mobility on the cost and performance of UEs. Moreover, except for [13], the other work only determined whether to offload the tasks of UEs to MEC servers, but did not decide the resources (i.e., CPU frequency and transmission power) allocation strategy for the UEs.

Several researchers have addressed the issue of UE mobility in the short-term. According to the mobility characteristic, we classify existing research into the following three categories: (1) For UE with random mobility, we know anything about the UE's movement regularity, and can only make the strategy based on the current location of the UE. For example, Taleb et al. [28] proposed a Markov decision process based algorithm to optimize the strategy. (2) For UE with predictable mobility, we can predict some future locations of the UE, and make the strategy by using the current and future locations of the UE. For example, Wu et al. [32] and Plachy et al. [23] developed the location prediction method respectively. (3) For UEs with fully known mobility, we know everything about the UE's movement regularity and its all future locations in advance. Therefore, we can make the strategy based on the whole movement path of the UE. For example, Wang et al. [31] optimized the cost of UEs based on their mobility regularities. Under the assumption that the task has been uploaded to the servers, the above work studied the impact of UE mobility on the task offloading optimization. Although Yu et al. [35] made the task offloading decision, but the resource allocation strategies are not considered in their work. Moreover, all the above work did not consider the impact of different mobility characteristics on the cost and performance of UEs.

Also, there is work that optimized the offloading strategy over a long time. Shen et al. [24] minimized the total energy consumption over a long time by reducing the number of service migrations. Sun et al. [26] minimized the average delay over multiple tasks of a UE while satisfying the energy consumption constraint. Ouyang et al. [22] investigated the cost-performance tradeoff of UEs in the long-term. Although the above work investigated the long-term cost and performance optimization problem, their assumption that all UEs stay in a certain area for a long time is too strong. Meanwhile, these work not only failed to study the impact of different mobility characteristics on the task offloading strategy, but also ignored the advantages of optimizing the strategy of UEs staying in a certain area for a short time by using their mobility characteristics. In addition, the above work

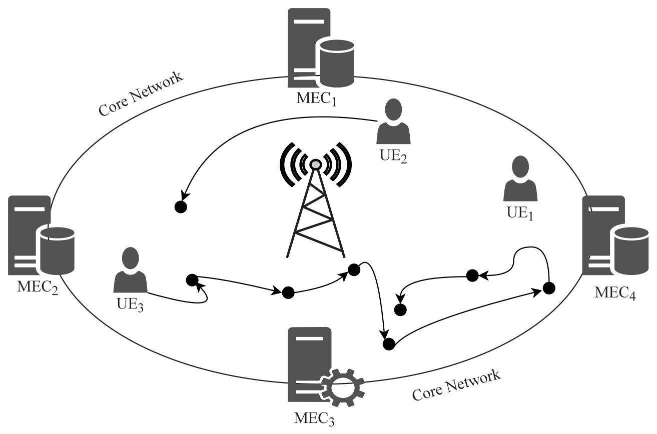


Fig. 1. An example scenario investigated in this paper.

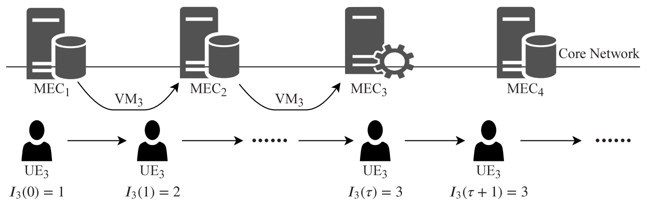


Fig. 2. An example of service migration.

did not involve the development of resource allocation strategies for UEs.

To address the above limitations, in our preliminary work [9], we analyzed different characteristics of UE mobility, and developed several greedy strategy based task offloading algorithms to optimize the strategy in a short-term based on these mobility characteristics. However, it is a huge challenge to obtain the mobility characteristics of UEs over a long time. Therefore, it is necessary and meaningful to investigate the long-term offloading strategy optimization problem, which solves the issue of UE mobility. This work significantly extends our preliminary work [9]. In this paper, we formulate the long-term cost and performance optimization problems respectively, and use a Lyapunov optimization method to decouple the original problems into two series of real-time optimization subproblems. Then, we develop the algorithms based on the Lyapunov optimization method, which do not require any prior knowledge of the mobility characteristics to optimize the long-term cost and performance of UEs. The algorithms proposed in this paper not only make the offloading decision, but also decide the resource allocation and service migration strategies. Moreover, we explore the impact of various key parameters on the cost and performance of UEs through extensive experiments.

3. System model and problem formulation

3.1. System model

The scenario studied in this paper is a time slot system. UEs move on a two-dimensional plane and execute a task at each time slot $\tau \in \{0, 1, 2, \ldots\}$ [23,31,32]. We assume that the time interval between the two successive time slots is long enough to complete a task. UE$_i$ represents ith UE, where $1 \le i \le N$. The service of UE$_i$ is represented by ith virtual machine (VM), i.e., VM$_i$. The location of UE$_i$ is $\big(x_i(\tau), y_i(\tau)\big)$, where $x_i(\tau)$, $y_i(\tau)$ are the abscissa and ordinate of UE$_i$ at τ. As shown in Fig. 1, we use some dots to represent the locations of UEs. Moreover, UEs have their own mobility characteristics. In the figure, UE$_1$ has no dot, which means that it moves in a random manner and we know nothing about its mobility regularity except for its current location. UE$_2$ has one dot, which means that it moves in a certain regularity and its location at $\tau + 1$ can be predicted at τ. UE$_3$ has a set of dots, which means that it moves in a given route and we know everything about its mobility regularity and its locations at all time slots in advance. $a_i(\tau)$ represents an offloadable task of UE$_i$ at τ. The number of CPU clock cycles required to complete $a_i(\tau)$ is denoted by $w_i(\tau)$ (cycles). The data size per CPU clock cycle of $a_i(\tau)$ is denoted by $\delta_i(\tau)$ (bits per cycle).

MEC$_j$ indicates an MEC server, where $1 \le j \le M$. We assume that high-speed data transmission between the MEC servers is implemented through the backbone network. (x_j, y_j) represents MEC$_j$'s location, where x_j, y_j are the abscissa and ordinate of the server. The deployment location of VM$_i$ at τ is denoted by $I_i(\tau) = j$. The binary variable $\lambda_{i,j}(\tau) \in \{0, 1\}$ represents whether $a_i(\tau)$ is uploaded to MEC$_j$. If $a_i(\tau)$ is executed by MEC$_j$, then $\lambda_{i,j}(\tau) = 1$, otherwise $\lambda_{i,j}(\tau) = 0$. $\lambda_{i,0}(\tau) = 1$ indicates that $a_i(\tau)$ will be executed by UE$_i$ itself. Because $a_i(\tau)$ can only be processed by one entity, thus $\sum_{j=0}^{M} \lambda_{i,j}(\tau) = 1$. We use $\upsilon_{i,j',j}(\tau) \in \{0, 1\}$ to represent whether to migrate the service of UE$_i$ form MEC$_{j'}$ to MEC$_j$ at τ. If $\upsilon_{i,j',j}(\tau) = 1$, then VM$_i$ will be migrated from MEC$_{j'}$ to MEC$_j$. We let $\upsilon_{i,j,j}(\tau) = 0$. Meanwhile, we assume that there is only one server that can deploy VM$_i$ at τ, thus $\sum_{j=1}^{M} \upsilon_{i,j',j}(\tau) \le 1$.

It can be seen from Fig. 2 that if VM$_3$ is deployed in MEC$_1$ at time slot τ, the increase in distance between UE$_3$ and the MEC server leads to the increase in transmission delay, thus degrading QoE and QoS. Thus, as shown in the figure, to provide seamless service for UE$_3$, VM$_3$ should be migrated among the MEC servers to follow the UE's movement. However, as shown in Fig. 1, when UE$_3$ moves back and forth between two locations, if VM$_3$ follows the UE to move back and forth between the two MEC servers, it will cause frequent service migration. The frequent service migration can also lead to an increase in energy consumption and latency. As shown in Figs. 1 and 2, when UE$_3$ moves back and forth between MEC$_3$ and MEC$_4$, because we know the mobility characteristic of UE$_3$, we can keep VM$_3$ at MEC$_3$, thereby reducing the number of unnecessary service migrations and improving QoE. This paper studies the offloading strategy optimization problem with different mobility characteristics, and develops algorithms based on these characteristics to improve QoE and QoS.

3.2. Communication model

We use $p_{i,max}$ Watt (W) to represent the maximum transmission power of UE$_i$. According to Shannon's theorem [21], in the channel interfered by Gaussian white noise, the maximum transmission rate is determined by $W_i \log_2(1 + \Upsilon)$, where W_i is the transmission channel bandwidth and Υ is the signal-to-noise ratio of the channel. Following the signal-to-noise ratio used in [5,6,11], we adopt the Rayleigh fading channel model [25]. Therefore, the signal-to-noise ratio is $\Upsilon = p_{i,j}(\tau)h_i^2 / \big(d_{i,j}^{\omega_i}(\tau)N_i\big)$, where $p_{i,j}(\tau), h_i$, $d_{i,j}(\tau) = \sqrt{\big(x_i(\tau) - x_j\big)^2 + \big(y_i(\tau) - y_j\big)^2}$, ω_i, N_i are the transmission power of UE$_i$ to upload $a_i(\tau)$ to MEC$_j$, the transmission channel fading coefficient, the distance between UE$_i$ and MEC$_j$ at τ, the channel path loss exponent, and the channel white Gaussian noise, respectively. Moreover, we also overlook the receiving latency of task result. Thus, the transmission rate of $a_i(\tau)$ from UE$_i$ to MEC$_j$ can be formulated as

$$R_{i,j}(\tau) = W_i \log_2\left(1 + \frac{p_{i,j}(\tau)h_i^2}{d_{i,j}^{\omega_i}(\tau)N_i}\right). \tag{1}$$

3.3. Computation model

3.3.1. Local computation model

The maximum CPU frequency of UE$_i$ is denoted by $f_{i,max}$ (cycles per second). Moreover, we assume that the UE can adjust its CPU frequency according to its demands. $f_i(\tau) \in [0, f_{i,max}]$ represents the actual CPU frequency of UE$_i$ when $a_i(\tau)$ is executed by the UE. The local computational time of $a_i(\tau)$ is

$$t_i^l(\tau) = \frac{w_i(\tau)}{f_i(\tau)}. \tag{2}$$

Based on [36], then we have the local energy consumption of $a_i(\tau)$, i.e.,

$$e_i^l(\tau) = \kappa_i w_i(\tau) f_i^2(\tau), \tag{3}$$

where κ_i is the coefficient factor of UE$_i$'s chip architecture.

3.3.2. MEC server computation model

We use f_j (cycles per second) to denote the computing power of MEC$_j$. Thus, the computational time of $a_i(\tau)$ executed by MEC$_j$ is

$$t_i^j(\tau) = t_{i,j,t}(\tau) + t_{i,j,e}(\tau) + t_{i,j,w}(\tau) + v_{i,j',j}(\tau) t_{i,j,m}(\tau), \tag{4}$$

where $t_{i,j,t}(\tau) = w_i(\tau)\delta_i(\tau)/R_{i,j}(\tau)$, $t_{i,j,e}(\tau) = w_i(\tau)/f_j$, $t_{i,j,w}(\tau)$, $t_{i,j,m}(\tau) = m_i$ are the transmission delay of $a_i(\tau)$, the computational time of $a_i(\tau)$ executed by MEC$_j$, the average waiting delay of $a_i(\tau)$ in MEC$_j$, the migration delay of VM$_i$, respectively. In this paper, without loss of generality, we assume that the migration delay m_i is a constant related to task type of UE$_i$. Accordingly, the UE$_i$'s energy consumption of $a_i(\tau)$ executed by MEC$_j$ can be formulated as

$$e_i^j(\tau) = p_{i,0}\big(t_{i,j,e}(\tau) + t_{i,j,w}(\tau) + v_{i,j',j}(\tau) m_i\big) + p_{i,j}(\tau) t_{i,j,t}(\tau), \tag{5}$$

where $p_{i,0}$ (W) is the static power of UE$_i$. Based on the above definitions, the latency of $a_i(\tau)$ can be formulated as

$$t_i(\tau) = \left(\Big(1 - \sum_{j=1}^{M} \lambda_{i,j}(\tau)\Big) t_i^l(\tau) + \sum_{j=1}^{M} \lambda_{i,j}(\tau) t_i^j(\tau) \right). \tag{6}$$

The energy consumption of $a_i(\tau)$ can be formulated as

$$e_i(\tau) = \left(\Big(1 - \sum_{j=1}^{M} \lambda_{i,j}(\tau)\Big) e_i^l(\tau) + \sum_{j=1}^{M} \lambda_{i,j}(\tau) e_i^j(\tau) \right). \tag{7}$$

3.4. Problem formulation

3.4.1. Energy minimization problem

According to the above definitions, we can formulate the energy consumption minimization problem of UE$_i$ as the following:

$$\begin{aligned} \text{P1}: &\min_{V_i, \Lambda_i, P_i, F_i} \lim_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} e_i(\tau), \qquad (8)\\ s.t.\ & C_1: 0 \le f_i(\tau) \le f_{i,max},\\ & C_2: 0 \le p_{i,j}(\tau) \le p_{i,max},\\ & C_3: \sum_{j=0}^{M} \lambda_{i,j}(\tau) = 1,\ \lambda_{i,j}(\tau) \in \{0, 1\},\\ & C_4: \sum_{j \ne j'}^{M} v_{i,j',j}(\tau) \le 1,\ v_{i,j',j}(\tau) \in \{0, 1\},\\ & C_5: t_i(\tau) \le \bar{t}_{i,max}, \end{aligned}$$

where $\bar{t}_{i,max}$ is the maximum average time latency of UE$_i$'s task. V_i, Λ_i, P_i, F_i are the service migration strategies, the offloading decisions, the transmission power strategies, and the CPU frequency strategies of UE$_i$ at all $\tau \in [0, T-1]$. In P1, C_1, C_2 are the constraints of CPU frequency and transmission power, respectively. C_3 represents that $a_i(\tau)$ can only be processed by one entity at τ. C_4 represents that VM$_i$ can only be deployed at one MEC server at τ. C_5 is the latency constraint of a task.

3.4.2. Latency minimization problem

The latency minimization problem of UE$_i$ can be formulated as the following:

$$\begin{aligned} \text{P2}: &\min_{V_i, \Lambda_i, P_i, F_i} \lim_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} t_i(\tau), \qquad (9)\\ s.t.\ & C_1, C_2, C_3, C_4,\\ & C_6: e_i(\tau) \le \bar{e}_{i,max}, \end{aligned}$$

where $\bar{e}_{i,max}$ is the maximum average time energy consumption of UE$_i$'s task. C_6 is the energy constraint of a task.

It is easy to know that P1 and P2 are not only the long-term optimization problems, but also mixed integer programming problems and NP-hard problems [17].

4. Energy minimization problem

4.1. Greedy strategy based algorithms

It is easy to know that the optimal solutions of P1 and P2 cannot be obtained at one time, but needs to be continuously adjusted to accommodate the dynamics of UEs based on the long-term knowledge. Therefore, to solve the mixed integer programming and NP-hard problems, we can use the greedy strategy, i.e., making offloading strategy with minimum cost at each time slot. Then, we can solve the problems task by task. Meanwhile, we can also use the following theorem to transform the original multiple-dimensional optimization problem into 1-dimensional optimization problem [4].

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

If we know the deployment location of VM$_i$, i.e., Eq. (6) can be transformed to

$$t_i(\tau) = t_i^l(\tau) + \lambda_{i,j}(\tau)\big(t_i^j(\tau) - t_i^l(\tau)\big), \tag{10}$$

and Eq. (7) can be transformed to

$$e_i(\tau) = e_i^l(\tau) + \lambda_{i,j}(\tau)\big(e_i^j(\tau) - e_i^l(\tau)\big). \tag{11}$$

Moreover, if $\lambda_{i,j}(\tau)$ can be relaxed to be a continuous variable, i.e., $0 \le \lambda_{i,j}(\tau) \le 1$, P1 and P2 can be transformed to the standard linear programming problems. Next, we first assume that $I_i(\tau) = j$ and decouple P1 into a subproblem of $a_i(\tau)$. Thus, the subproblem of P1 is

$$\begin{aligned} \text{P3}: &\min_{\boldsymbol{s_i(\tau)}} e_i(\tau), \qquad (12)\\ s.t.\ & C_1, C_2, C_5,\\ & C_7: \lambda_{i,j}(\tau) \in [0, 1]. \end{aligned}$$

In P3, we use $\boldsymbol{s_i(\tau)} \triangleq \big(\lambda_{i,j}(\tau), f_i(\tau), p_{i,j}(\tau)\big)$ to represent a task offloading strategy set consists of offloading decision, CPU frequency, and transmission power. Thus, we can decompose subproblem P3 into two subproblems based on the offloading decision, i.e., $\lambda_{i,j}(\tau)$.

Based on Theorem 1, we can obtain the optimal task offloading decision, CPU frequency, transmission power, and service migration strategies according to the following theorem.

Theorem 2. *For energy minimization problem, the optimal strategies of $a_i(\tau)$ can be obtained from the following equations:*

$$f_i^*(\tau) = \min\{f_{i,max}, \tilde{f}_i(\tau)\}, \tag{13}$$

$$p_{i,j}^*(\tau) = \min\{p_{i,max}, p_{i,j,min}(\tau)\}, \tag{14}$$

$$\lambda_{i,j^*}^*(\tau) = \Phi\{e_i^{l^*}(\tau) > e_i^{j^*}(\tau)\}, \tag{15}$$

$$v_{i,j,j^*}^*(\tau) = \Phi\{e_i^{j}(\tau) > e_i^{j^*}(\tau)\}, \tag{16}$$

where $\tilde{f}_i(\tau) = w_i(\tau)/\bar{t}_{i,max}$, and $\Phi\{o\} \in \{0, 1\}$ is a boolean function. If o is true, then $\Phi\{o\} = 1$. Otherwise, $\Phi\{o\} = 0$. $p_{i,j,min}(\tau) = (2^{b_{i,j}(\tau)} - 1)/\psi_{i,j}(\tau)$ where $\psi_{i,j}(\tau) = h_i^2/(d_{i,j}^{\omega_i}(\tau)N_i)$ and $b_{i,j}(\tau) = w_i(\tau)\delta_i(\tau)/W_i\big(t_{i,r}(\tau) - t_{i,j,e}(\tau) - t_{i,j,w}(\tau) - v_{i,j',j}(\tau)m_i\big)$.

Proof. If $a_i(\tau)$ is executed by UE$_i$ itself, then $\lambda_{i,0}(\tau) = 1$. Plugging Eq. (2) into $C_5 : t_i(\tau) \le \bar{t}_{i,max}$, we have $f_i(\tau) \ge w_i(\tau)/\bar{t}_{i,max}$. Let $\tilde{f}_i(\tau) = w_i(\tau)/\bar{t}_{i,max}$. Thus we have $f_i(\tau) \ge \tilde{f}_i(\tau)$. That is, there is a minimal CPU frequency $\tilde{f}_i(\tau)$ that can satisfy the latency constraint C_5. According to $C_1 : f_i(\tau) \in [0, f_{i,max}]$, we can easily obtain the optimal CPU frequency strategy from $f_i^*(\tau) = \min\{f_{i,max}, \tilde{f}_i(\tau)\}$.

If $a_i(\tau)$ is executed by MEC$_j$, then $\lambda_{i,j}(\tau) = 1$. Plugging $t_{i,j,t}(\tau) = w_i(\tau)\delta_i(\tau)/R_{i,j}(\tau)$ into $C_5 : t_i(\tau) \le \bar{t}_{i,max}$, we have an inequality $w_i(\tau)\delta_i(\tau)/R_{i,j}(\tau) \le \bar{t}_{i,max}$. Then, plugging Eq. (1) into the inequality, we have $p_{i,j}(\tau) \ge (2^{b_{i,j}(\tau)} - 1)/\psi_{i,j}(\tau)$, where $\psi_{i,j}(\tau) = h_i^2/(d_{i,j}^{\omega_i}(\tau)N_i)$ and $b_{i,j}(\tau) = w_i(\tau)\delta_i(\tau)/W_i\big(t_{i,r}(\tau) - t_{i,j,e}(\tau) - t_{i,j,w}(\tau) - v_{i,j',j}(\tau)m_i\big)$. Thus, there is a minimal transmission power $p_{i,j,min}(\tau) = (2^{b_{i,j}(\tau)} - 1)/\psi_{i,j}(\tau)$ between UE$_i$ and MEC$_j$ in order to satisfy the latency constraint C_5. According to $C_2 : p_{i,j}(\tau) \in [0, p_{i,max}]$, we can easily obtain the optimal transmission power strategy from $p_{i,j}^*(\tau) = \min\{p_{i,max}, p_{i,j,min}(\tau)\}$.

Since $e_i(\tau)$ is a linear function w.r.t. $\lambda_{i,j}(\tau)$, we can obtain the offloading decision from $\lambda_{i,j}^*(\tau) = \Phi\{e_i^{l^*}(\tau) > e_i^{j^*}(\tau)\}$, where MEC$_{j^*}$ represents the optimal server.

When the cost of service migration is less than the benefit of service migration, the service migration operation is triggered. We can iterate all MEC servers and calculate the energy consumption of UE$_i$. Then, we can obtain the service migration strategy from $v_{i,j,j'}(\tau) = \Phi\{e_i^j(\tau) > e_i^{j'}(\tau)\}$. Moreover, if we regard MEC$_{j^*}$ with minimal cost as the optimal server, we can obtain the optimal service migration strategy from $v_{i,j,j^*}^*(\tau) = \Phi\{e_i^j(\tau) > e_i^{j^*}(\tau)\}$. □

According to Theorem 2, we have a corollary, i.e.,

Corollary 1. *For $a_i(\tau)$ and MEC$_j$, MEC$_j$ is an available server for UE$_i$ when $p_{i,j,min}(\tau) \le p_{i,max}$.*

Proof. It is easy to know that if $a_i(\tau)$ can be completed within the latency constraint, the minimal transmission power must satisfy $p_{i,j,min}(\tau) \le p_{i,max}$. Otherwise, the delay of the task executed by MEC$_j$ violates the constraint C_5. □

In fact, we can use Corollary 1 to check the feasibility of an MEC server for executing $a_i(\tau)$. Thus, we can determine an available MEC server set of $a_i(\tau)$ in advance, i.e., $\mathbf{M}_i(\tau)$, to reduce the server scale that needed to be searched. Then, we can get the optimal transmission power strategies for UE$_i$ transmitting $a_i(\tau)$ to each MEC server (MEC$_j \in \mathbf{M}_i(\tau)$). Next, the corresponding cost for MEC$_j$ executing the task can be calculated based on Eq. (5). Meanwhile, we regard MEC$_{j^*}$ with minimal cost as the optimal server. The optimal CPU frequency and corresponding cost can be easily obtained from Eqs. (3) and (13), respectively. By comparing the optimal cost of local execution and server execution, we get the optimal offloading decision $\lambda_{i,j}^*(\tau)$. If $\lambda_{i,j}^*(\tau) = 1$, the service migration is triggered when $I_i(\tau) \ne j^*$. Hence, we obtain the optimal offloading strategy of $a_i(\tau)$.

Remark 1. In this paper, although the CPU frequency is assumed to be a continuous variable, the algorithms proposed in the paper can be easily adapted to discrete CPU frequencies. For instance, let us consider CPU frequency $f_i(\tau) \in \mathcal{CPU} \triangleq \{f_{i,1}, f_{i,2}, \ldots, f_{i,max}\}$, where $f_{i,1} < f_{i,2} < \cdots < f_{i,max}$ are the possible CPU frequency values of UE$_i$. For given $a_i(\tau)$ and $\bar{t}_{i,max}$, the optimal CPU frequency is determined. For the short-term energy-minimization problem, to meet the latency constraint of $a_i(\tau)$, the minimal CPU frequency is $f_{i,min}(\tau) = w_i(\tau)/\bar{t}_{i,max}$. If $f_{i,min}(\tau) > f_{i,max}$, we can conclude that UE$_i$ is not capable to complete the task $a_i(\tau)$ within the latency constraint. That is, the task should be offloaded to the MEC servers for processing. If $f_{i,min}(\tau) \le f_{i,max}$, the task can be executed locally. Moreover, if $f_{i,min}(\tau) \in \mathcal{CPU}$, then $f_{i,min}(\tau)$ is the optimal CPU frequency strategy, i.e., $f_i^*(\tau) = f_{i,min}(\tau)$. If $f_{i,min}(\tau) \notin \mathcal{CPU}$ and $f_{i,min}(\tau) < f_{i,max}$, although we cannot obtain the optimal CPU frequency strategy directly, it can be confirmed that the suboptimal CPU frequency $f_i^*(\tau)$ should satisfy $f_i^*(\tau) \ge f_{i,min}(\tau)$. Hence, the suboptimal CPU frequency is the smallest element in $\mathcal{CPU}$ that is greater than $f_{i,min}(\tau)$, i.e., $f_i^*(\tau) = \min\{f_i(\tau) | f_i(\tau) \ge f_{i,min}(\tau), f_i(\tau) \in \mathcal{CPU}\}$.

It should be noted that $f_i^*(\tau)$ may be a suboptimal solution of P3. However, $f_i^*(\tau)$ is the best CPU frequency strategy that UE$_i$ can really adopt.

It is feasible to obtain the mobile characteristics of UEs in a short-term. Thus, we can use the mobility characteristics to optimize the strategies of the UEs. Next, we detail the three task offloading algorithms based on different mobility characteristics.

Algorithm 1 EO-RM: energy-optimal algorithm for UE$_i$ with random mobility.

Input: $\bar{t}_{i,max}, f_{i,max}, p_{i,max}, I_i(0) = j, W_i, h_i, N_i, \omega_i, w_i$, and δ_i, for all $\tau \in [0, T-1]$.
Output: Λ_i^*, F_i^*, P_i^*, V_i^*, and I_i^*.

```
1: while $\tau < T$ do
2:   Obtain $\mathbf{M}_i(\tau)$ from Corollary 1;
3:   Calculate $f_i^*(\tau)$, $p_{i,j}(\tau)$, $e_i^l(\tau)$ and $e_i^j(\tau)$;
4:   Record the current minimal energy consumption $e_i^m \leftarrow e_i^j(\tau)$;
5:   for MEC$_{j'} \in \mathbf{M}_i(\tau)$ do
6:     Calculate $p_{i,j'}(\tau)$ and $e_i^{j'}(\tau)$;
7:     if $e_i^{j'}(\tau) < e_i^m(\tau)$ then
8:       Update migration strategy $v_{i,I_i(\tau-1),j'}^*(\tau) \leftarrow 1$;
9:       Update service location $I_i^*(\tau) \leftarrow j'$;
10:      Update energy consumption $e_i^m \leftarrow e_i^{j'}(\tau)$;
11:    end if
12:  end for
13:  Update optimal offloading decision $\lambda_{i,I_i(\tau)}^*(\tau)$ and transmission power $p_{i,I_i(\tau)}^*(\tau)$;
14:  $\tau \leftarrow \tau + 1$;
15: end while
16: return $\Lambda_i^*$, $F_i^*$, $P_i^*$, $V_i^*$, and $I_i^*$.
```

4.1.1. The algorithm for UEs with random mobility

For UE$_i$ with random mobility, we make the strategy based on the UE's current informations, i.e., location and task informations. For the energy minimization problem, Algorithm 1 shows an energy-optimal algorithm to decide task offloading strategies for UE$_i$ with random mobility. The algorithm is named EO-RM. When

$a_i(\tau)$ and MEC$_j$ are given, $f_i(\tau)$, $p_{i,j}(\tau)$, $e_i^l(\tau)$, and $e_i^j(\tau)$ can be obtained accordingly. Then, UE$_i$ iterates $\mathbf{M_i}(\tau)$ to try to find MEC$_{j^*}$ while making the service migration strategy. That is, UE$_i$ finds an optimal MEC server with minimum execution energy consumption among MEC$_j \in \mathbf{M_i}(\tau)$ at each time slot. The complexity of the algorithm is $O\big(\sum_{\tau=0}^{T-1}|\mathbf{M_i}(\tau)|\big)$, where $|\mathbf{M_i}(\tau)|$ is the number of MEC servers in this set.

Algorithm 2 EO-PM: energy-optimal algorithm for UE$_i$ with predictable mobility.

Input: $\bar{t}_{i,max}, f_{i,max}, p_{i,max}, I_i(0) = j, W_i, h_i, N_i, \omega_i, w_i$, and δ_i, for all $\tau \in [0, T-1]$.
Output: Λ_i^*, F_i^*, P_i^*, V_i^*, and I_i^*.

```
1: while $\tau < T$ do
2:   Obtain $\mathbf{M_i}(\tau)$ and $\mathbf{M_i}(\tau+1)$ from Corollary 1;
3:   Obtain $\boldsymbol{s_i}(\tau)$, $\boldsymbol{s_i}(\tau+1)$, $I_i(\tau) = j_\tau$, and $I_i(\tau+1) = j_{\tau+1}$ from Algorithm 1;
4:   if $\lambda_{i,j_\tau}(\tau) = \lambda_{i,j_{\tau+1}}(\tau+1) = 1$ and $j_\tau \neq j_{\tau+1}$ then
5:     for MEC$_j \in \mathbf{M_i}(\tau) \cap \mathbf{M_i}(\tau)$ do
6:       if $e_i^j(\tau)+e_i^j(\tau+1) < e_i^{j_\tau}(\tau)+e_i^{j_{\tau+1}}(\tau+1)$, $t_i^j(\tau) \le \bar{t}_{i,max}(\tau)$ and $t_i^j(\tau+1) \le \bar{t}_{i,max}$ then
7:         Update service location at $\tau$, i.e., $I_i(\tau) \leftarrow j$;
8:         Update service location at $\tau+1$, i.e., $I_i(\tau+1) \leftarrow j$;
9:         Update offloading decision, CPU frequency, transmission power, and service migration strategies at two successive time slots, i.e., $\boldsymbol{s_i}(\tau)$, $\boldsymbol{s_i}(\tau+1)$, $\upsilon_{i,I_i(\tau-1),j}(\tau)$, and $\upsilon_{i,j_\tau,j}(\tau+1)$;
10:      end if
11:    end for
12:  end if
13:  Obtain $\boldsymbol{s_i^*}(\tau)$, $\boldsymbol{s_i^*}(\tau+1)$, $\upsilon^*_{i,I_i(\tau-1),j^*}(\tau)$, and $\upsilon^*_{i,I_i(\tau),j^*}(\tau+1)$;
14:  $\tau \leftarrow \tau+2$;
15: end while
16: return $\Lambda_i^*$, $F_i^*$, $P_i^*$, $V_i^*$ and $I_i^*$.
```

4.1.2. The algorithm for UEs with predictable mobility

For UE$_i$ with predictable mobility, we can predict some future locations of the UE, and make the strategy by using the UE's current and future locations. We assume that the UE's location at $\tau+1$ can be predicted exactly at τ [35]. Therefore, we can reformulate the subproblem of P1 as

$$\text{P4}: \min_{\boldsymbol{s_i}(\tau),\boldsymbol{s_i}(\tau+1)} e_i(\tau)+e_i(\tau+1), \quad (17)$$
$$s.t.\ C_1, C_2, C_4, C_5, C_7.$$

Let $I_i(\tau) = j_\tau$ and $I_i(\tau+1) = j_{\tau+1}$ be the VM deployment policies for $a_i(\tau)$ and $a_i(\tau+1)$, which are gotten from Algorithm 1. Moreover, there is a common MEC server executing the two successive tasks, which may further reduce the cost of UE$_i$. The rational of the assumption revealed by the following theorem.

Theorem 3. *If $j_\tau \neq j_{\tau+1}$ and $\lambda_{i,j_\tau}(\tau) = \lambda_{i,j_{\tau+1}}(\tau+1) = 1$, there may be a new optimal strategy for two tasks, i.e., $I_i'(\tau) = I_i'(\tau+1) = j^*$ and $\lambda_{i,j^*}(\tau) = \lambda_{i,j^*}(\tau+1) = 1$, where MEC$_{j^*} \in \mathbf{M_i}(\tau)\cap\mathbf{M_i}(\tau+1)$. Otherwise, the original strategies are the optimal strategies for $a_i(\tau)$ and $a_i(\tau+1)$.*

Proof. If $I_i(\tau) \neq I_i(\tau+1)$, $\lambda_{i,j_\tau}(\tau) = \lambda_{i,j_{\tau+1}}(\tau+1) = 1$, and there is a new optimal strategy for the two tasks, i.e., $\lambda_{i,I_i'(\tau)}(\tau) = \lambda_{i,I_i'(\tau+1)}(\tau+1) = 1$, where $I_i'(\tau) = j'_\tau$, $I_i'(\tau+1) = j'_{\tau+1}$, and $j'_\tau \neq j'_{\tau+1}$. Then, we have an inequality, i.e., $e_i^{j_\tau}(\tau)+e_i^{j_{\tau+1}}(\tau+1) > e_i^{j'_\tau}(\tau)+e_i^{j'_{\tau+1}}(\tau+1)$, where MEC$_{j'_\tau} \in \mathbf{M_i}(\tau)$ and MEC$_{j'_{\tau+1}} \in \mathbf{M_i}(\tau+1)$.

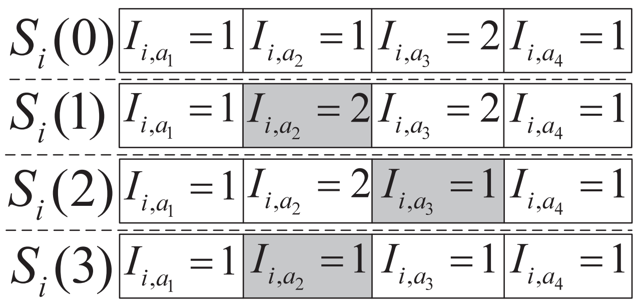
Algorithm 3 EO-KM: energy-optimal algorithm for UE$_i$ with fully known mobility.

Input: $\bar{t}_{i,max}, f_{i,max}, p_{i,max}, S_i(0) = j, W_i, h_i, N_i, \omega_i, w_i, \delta_i, \epsilon_i$, and Γ_i, for all $\tau \in [0, T-1]$.
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 $\sum_{k=1}^{T-1} e_i(k) - \Sigma_t > \epsilon_i$ and $\gamma < \Gamma_i$ do
5:   $\phi \leftarrow I_i(0)$;
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 $k+1-\rho > 0$ 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 $e_i^{I_{i,a_k}}(k') + \sum_{\xi=k'+1}^{k} e_i(\xi) < \sum_{\xi=k'}^{k} e_i(\xi)$ and $t_i^{I_{i,a_k}}(k') \le \bar{t}_{i,max}$ then
14:            Update service location $I_{i,a_{k'}} \leftarrow I_{i,a_k}$;
15:            Update offloading decision, CPU frequency and transmission power strategies, i.e., $\boldsymbol{s_{i,a_k}}$, and service migration strategy $\upsilon_{i,j',j,a_k}$;
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:  $\Sigma_t \leftarrow \sum_{k=1}^{T-1} e_i(k)$;
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$_{I_{i,a_k}} \in \mathbf{M_i}(k+1)$ then
28:     if $e_i^{I_{i,a_k}}(k) + e_i^{I_{i,a_k}}(k+1) + e_i^{I_{i,a_k}}(k+2) < \sum_{k'=k}^{k+2} e_i(k')$ and $t_i^{I_{i,a_k}}(k+1) < \bar{t}_{i,max}$ then
29:       Update service location $I_{i,a_{k+1}} \leftarrow I_{i,a_k}$;
30:       Update offloading decision, CPU frequency and transmission power strategies, i.e., $\boldsymbol{s_{i,a_{k+1}}}$, and service migration strategy $\upsilon_{i,j',j,a_k}$;
31:     end if
32:   end if
33: end for
34: return $\Lambda_i^*$, $F_i^*$, $P_i^*$, $V_i^*$, and $I_i^*$.
```

However, we know that $e_i^{j_\tau}(\tau) \le e_i^{j'_\tau}$ and $e_i^{j_{\tau+1}}(\tau+1) \le e_i^{j'_{\tau+1}}(\tau+1)$, which contradicts with the premise. Thus, $j'_\tau = j'_{\tau+1} = j^*$ should be true (i.e., the two tasks are executed at a common MEC server MEC$_{j^*}$) when there is a new strategy for the two tasks. Moreover, if MEC$_{j^*} \notin \mathbf{M_i}(\tau)\cap\mathbf{M_i}(\tau+1)$, which means that one of the tasks violates the latency constraint. Therefore, if $j_\tau = j_{\tau+1}$ and $\lambda_{i,j_\tau}(\tau) = \lambda_{i,j_{\tau+1}}(\tau+1) = 1$, then MEC$_{j_\tau}$ is the optimal execution location for the two tasks.

If $\lambda_{i,j_\tau}(\tau) \neq \lambda_{i,j_{\tau+1}}(\tau+1)$, we assume that $\lambda_{i,0}(\tau) = 1$ and $\lambda_{i,j_{\tau+1}}(\tau+1) = 1$, we have inequalities $e_i^l(\tau) \le e_i^j(\tau)$ ($\forall$MEC$_j \in \mathbf{M_i}(\tau)$) and $e_i^{j_{\tau+1}}(\tau+1) \le e_i^j(\tau+1)$ ($\forall$MEC$_j \in \mathbf{M_i}(\tau+1)$). Thus, if we have $e_i^l(\tau) + e_i^{j_{\tau+1}}(\tau+1) > e_i^{j'_\tau}(\tau) + e_i^{j'_{\tau+1}}(\tau+1)$, which



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

Fig. 3. An illustration process of Algorithm 3.

contradicts with the premise. Based on the above, we have the conclusion. □

For the energy minimization problem, Algorithm 2 shows an energy-optimal algorithm to decide task offloading strategies for UE$_i$ with predicted mobility. The algorithm is named EO-PM. We can first obtain the optimal strategies of $a_i(\tau)$ and $a_i(\tau+1)$ from Algorithm 1, respectively. The services of two successive tasks are rescheduled in MEC$_j \in \mathbf{M}_i(\tau) \cap \mathbf{M}_i(\tau+1)$ according to Theorem 3. Meanwhile, the task offloading strategies of the two tasks are updated accordingly. The complexity of the algorithm is $O\left(\sum_{\tau=0}^{T-1} |\mathbf{M}_i(\tau)|\right)$.

4.1.3. The algorithm for UEs with fully known mobility

For UE$_i$ with fully known mobility, we know everything about the UE's movement regularity and its all future locations in advance. For the energy minimization problem, Algorithm 3 shows an energy-optimal algorithm to decide task offloading strategies for UE$_i$ with fully known mobility. The algorithm is named EO-KM. An example process of Algorithm 3 is illustrated in Fig. 3. The initial Λ_i, F_i, P_i, V_i, and I_i are obtained from Algorithm 2. As shown in the figure, we use $S_i(0)$ to represent the initial locations of UE$_i$'s service. Since the whole movement and task informations of UE$_i$ is know, to avoid confusion with the former two real-time algorithms, we use k instead of τ to represent the subscript of parameters. We iterate $a_k \in A_i$ to update the task strategies of UE$_i$, where a_k ($k \in [1, T]$) is kth time slot task of UE$_i$ and A_i represents the task set of the UE at all time slots (i.e., $|A_i| = T$). $I_{i,a_k} \in [1, M]$ denotes the service location of $a_k \in A_i$. $\boldsymbol{s_{i,a_k}}$ is an offloading strategy (including offloading decision, CPU frequency strategy, and transmission power strategy) of a_k. $\mathbf{M_{i,a_k}}$ is the available MEC server set of a_k. $v_{i,j',j,a_k} \in \{0, 1\}$ is the service migration strategy of a_k.

There are two iterations for updating the strategies of UE$_i$ in Algorithm 3. The first iteration (Lines 4–25) is migrating the service of the UE in advance. Let $I_{i,a_\rho} = j$, we use a_ρ ($\rho \in [1, T]$) and a_k to represent the first task and the last task among the successive tasks executed by MEC$_j$, respectively. Thus, the number of the successive tasks executed by MEC$_j$ is $k + 1 - \rho$. If $k+1-\rho > 0$, we try to migrate VM$_i$ ahead to reduce the UE's cost. As shown in Fig. 3, if we migrate VM$_i$ from MEC$_1$ to MEC$_2$ when a_2 is executed, and the cost of $k+1-\rho$ tasks can be reduced, we will adjust VM$_i$'s location and UE$_i$'s task offloading strategies accordingly. Hence, we obtain $S_i(1)$. If the reduced cost is less than ϵ_i or the iteration amount exceeds the maximum number of iterations Γ_i, the early migration process is terminated. Therefore, we have $S_i(2)$.

The second iteration (Lines 26–33) is avoiding migrating the service of the UE. If $I_{i,a_k} \neq I_{i,a_{k+1}}$, $I_{i,a_k} = I_{i,a_{k+2}}$ and MEC$_{I_{i,a_{k+1}}} \in \mathbf{M_{i,a_k}}$, we try not to migrate VM$_i$ when a_{k+1} is executed. As shown in $S_i(2)$, if a_2 is executed by MEC$_1$, the UE$_i$'s cost can be reduced. We can adjust VM$_i$'s deployment strategy and UE$_i$'s task offloading strategies accordingly. Finally, we obtain the optimal strategies. The complexity of the algorithm is $O\left(\sum_{\tau=0}^{T-1} |\mathbf{M}_i(\tau)| + \Gamma_i T^2\right)$.

4.2. Lyapunov optimization based algorithm

4.2.1. The background of Lyapunov optimization method

Lyapunov optimization is a method of using a Lyapunov function to optimal a dynamic system, and has low computational complexity and quantifiable worst-case performance [18]. The method has been widely used for task scheduling in queueing networks [14,20]. In this section, we present the background of the Lyapunov optimization method.

Consider a queueing system that operates in discrete time with unit time slots $\tau \in \{1, 2, 3, \ldots\}$, and let $q(\tau)$ be the workload of a new arrival task at τ. The workload $q(\tau)$ will be stored in a queue $Q(\tau)$ to be scheduled. Meanwhile, the system is described by the queue backlog of $Q(\tau)$. A schedule action is taken at every time slot τ, which affects the arrivals and departures of $Q(\tau)$. The method defines a function $L\big(Q(\tau)\big)$ as the square of backlog multiplied by 1/2, i.e., $L\big(Q(\tau)\big) = Q(\tau)^2/2$. The function is named the Lyapunov function, and can be used to measure the system congestion. Next, the method defines the Lyapunov drift $\Delta(\tau) = L\big(Q(\tau+1)\big) - L\big(Q(\tau)\big)$ as the difference in the Lyapunov function between two successive time slots. If schedule actions are made at every time slot to greedily minimize $\Delta(\tau)$, then $Q(\tau)$ is consistently pushed towards a lower congestion state, which intuitively maintains system stability. It should be noted that the specific meaning of system stability varies according to different problem definitions. For example, in this paper, the system stability means that the energy consumption and latency of UEs remain at a certain level.

For a schedule system, we want to minimize an objective function $\mathcal{F}\big(q(\tau)\big)$ while ensuring the system stability. Instead of taking actions to minimize $\Delta(\tau)$, the actions are taken to minimize the drift-plus-penalty function at every time slot τ. The drift-plus-penalty function is formulated as $\Delta(\tau) + V\mathcal{F}\big(q(\tau)\big)$, where V is a non-negative weight parameter that indicates the importance of how much we emphasize the optimization objective.

It can be known that the Lyapunov optimization method only requires the knowledge of current information. Therefore, we can use the Lyapunov optimization method to transform P1 and P2 into two series of online minimization subproblems respectively, which addresses the challenge of acquiring UE's mobility characteristics over a long time.

4.2.2. The algorithm

The greedy strategy based algorithms require that the mobility type of UEs is known in advance. However, it is unrealistic to obtain the mobility characteristics of UEs over a long time. Fortunately, the long-term constraints (i.e., C_5, C_6) in the problems can be regarded as the queue stability control problem respectively [20]. The Lyapunov optimization method provides an efficient approach to decouple the long-term optimization problem. Next, we detail the transform process.

Let $Q_i(\tau)$ represent a virtual discrete time queueing system of UE$_i$ defined over time slot τ. In the paper, $Q_i(0) = 0$. The future state of the queue is derived by the current computational time $t_i(\tau)$ and the average latency constraint $\bar{t}_{i,max}$ according to the dynamic equation

$$Q_i(\tau+1) = \max\{Q_i(\tau) - \bar{t}_{i,max} + t_i(\tau), 0\}. \tag{18}$$

The virtual queue $Q_i(\tau)$ is the backlog at τ and can be represented the additional time required to process the tasks. Thus, $Q_i(\tau)$ is used to enforce the strategies meet the constraint C_5. We use Lyapunov optimization method to transform P1, then have the following theorem.

Algorithm 4 EO-LY: energy-optimal algorithm for UE$_i$ based on the Lyapunov optimization method.

Input: $\bar{t}_{i,max}, f_{i,max}, p_{i,max}, I_i(0), W_i, h_i, N_i, \omega_i, w_i$, and δ_i, for all $\tau \in [0, T-1]$.
Output: Λ_i^*, F_i^*, P_i^*, V_i^*, and I_i^*.

```
1: while $\tau < T$ do
2:   Calculate $f_i^*(\tau)$ and $e_i^{l*}(\tau)$;
3:   $e_i^m \leftarrow +\infty$;
4:   for $j \in [1, M]$ do
5:     Calculate $p_{i,j}^*(\tau)$ and $e_i^{j*}(\tau)$;
6:     if $e_i^j(\tau) < e_i^m(\tau)$ then
7:       Update service migration strategy $v_{i,I_i(\tau-1),j}^*(\tau) \leftarrow 1$;
8:       Update service location $I_i^*(\tau) \leftarrow j$;
9:       Update energy consumption $e_i^m \leftarrow e_i^{j'}(\tau)$;
10:    end if
11:  end for
12:  Update offloading decision $\lambda_{i,I_i(\tau)}^*(\tau)$ and transmission power $p_{i,I_i(\tau)}^*(\tau)$;
13:  $\tau \leftarrow \tau + 1$;
14: end while
15: return $\Lambda_i^*$, $F_i^*$, $P_i^*$, $V_i^*$, and $I_i^*$.
```

Theorem 4. *P1 is equivalent to the following problem*

$$\text{P5}: \min_{\mathbf{s}_i(\tau)} Z_i e_i(\tau) + Q_i(\tau) t_i(\tau), \tag{19}$$

$$s.t. \;\; C_1, C_2, C_4, C_7, \tag{20}$$

where $Z_i > 0$ is the weight parameter that indicates the importance of how much we emphasize energy consumption of UE$_i$.

Proof. Based on the definition of virtual queue, the Lyapunov function is $L\big(Q(\tau)\big) = Q_i(\tau)^2/2$. The change in Lyapunov function from one slot to the next slot is

$$\begin{aligned} L\big(Q_i(\tau+1)\big) - L\big(Q_i(\tau)\big) &= \frac{1}{2}\big(Q_i(\tau+1)^2 - Q_i(\tau)^2\big) \\ &\le \frac{1}{2}\big(t_i(\tau)^2 + \bar{t}_{i,max}^2\big) \\ &\quad + Q_i(\tau)\big(t_i(\tau) - \bar{t}_{i,max}\big) - t_i(\tau)\bar{t}_{i,max} \\ &\le B + Q_i(\tau)t_i(\tau) - Q_i(\tau)\bar{t}_{i,max}, \end{aligned} \tag{21}$$

where $B = \big(t_{i,max}^2 + \bar{t}_{i,max}^2\big)/2$ and $t_{i,max}$ is the maximum latency of task.

The conditional Lyapunov drift is

$$\Delta\big(Q_i(\tau)\big) \triangleq \mathbb{E}\big\{L\big(Q_i(\tau+1)\big) - L\big(Q_i(\tau)\big) | Q_i(\tau)\big\}, \tag{22}$$

where $\mathbb{E}$ represents the expectation. According to Eqs. (21), (22), and the law of iteration expectation [20], we have

$$\begin{aligned} \mathbb{E}\big\{\Delta\big(Q_i(\tau)\big)\big\} &= \mathbb{E}\big\{L\big(Q_i(\tau+1)\big) - L\big(Q_i(\tau)\big)\big\} \\ &\le B + \mathbb{E}\big\{Q_i(\tau)\big\}\big(t_i(\tau) - \bar{t}_{i,max}\big). \end{aligned} \tag{23}$$

And then, according to the law of telescoping sums [20] and $Q_i(0) = 0$, we have

$$\mathbb{E}\big\{L\big(Q_i(T)\big)\big\} \le BT + \sum_{\tau=0}^{T-1} \mathbb{E}\big\{Q_i(\tau)\big\}\big(t_i(\tau) - \bar{t}_{i,max}\big). \tag{24}$$

Hence, when $T \to \infty$, the above equation can be rearranged as

$$\lim_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\big\{Q_i(\tau)\big\} \le \lim_{T\to\infty} \frac{1}{T} \frac{BT - \mathbb{E}\big\{L\big(Q_i(T)\big)\big\}}{\sum_{\tau=0}^{T-1}\big(\bar{t}_{i,max} - t_i(\tau)\big)} = 0. \tag{25}$$

Based on Eqs. (24) and (25), $Z(\tau)$ is mean rate stable, that is the energy consumption constraint of MSP can be satisfied [20].

The Lyapunov drift-plus-penalty function is

$$\begin{aligned} \Delta\big(Q_i(\tau)\big) + Z_i e_i(\tau) &\le Q_i(\tau)\mathbb{E}\left\{t_i(\tau) - \bar{t}_{i,max} | Q_i(\tau)\right\} + B + Z_i e_i(\tau) \\ &\le B + Q_i(\tau)t_i(\tau) + Z_i e_i(\tau). \end{aligned} \tag{26}$$

Therefore, if we want to minimize the long-term energy consumption while satisfying the latency constraint C_5, we can minimize $\Delta\big(Q_i(\tau)\big) + Z_i e_i(\tau)$. Equivalently, we can minimize $Z_i e_i(\tau) + Q_i(\tau)t_i(\tau)$ and have the theorem. □

Theorem 4 unifies the energy consumption of UE$_i$ and the latency constraint of UE$_i$ into an equation. As shown in P5, the solution of P5 is an approximate optimal solution of P1. Meanwhile, the average time energy consumption deviates by at most $O(1/Z_i)$ from the optimal solution of P1, with the average queue backlog bounded of $O(Z_i)$ [20]. The optimal solutions $\lambda_{i,j^*}^*(\tau)$, $f_i^*(\tau)$, $p_{i,j}^*(\tau)$, and $v_{i,j',j^*}^*(\tau)$ of P5 can be easily obtained according to the following theorem.

Theorem 5. *For MEC$_j$ and UE$_i$, the optimal strategies of P5 can be obtained from the following equations:*

$$f_i^*(\tau) = \min\{f_{i,max}, \hat{f}_i(\tau)\}, \tag{27}$$

$$p_{i,j}^*(\tau) = \min\{p_{i,max}, \hat{p}_{i,j}(\tau)\}, \tag{28}$$

$$\lambda_{i,j^*}^*(\tau) = \Phi\{e_i^{l^*}(\tau) > e_i^{j^*}(\tau)\}, \tag{29}$$

$$v_{i,j',j^*}^*(\tau) = \Phi\{e_i^{j'}(\tau - 1) > e_i^{j^*}(\tau)\}, \tag{30}$$

where $\hat{f}_i(\tau) = \sqrt[3]{Q_i(\tau)/(2Z_i\kappa_i)}$, and $\hat{p}_{i,j}(\tau)$ is obtained from the following equation:

$$\begin{aligned} h\big(\hat{p}_{i,j}(\tau)\big) = Z_i \Big(&\big(1 + \hat{p}_{i,j}(\tau)\psi_{i,j}(\tau)\big)\ln\big(1 + \hat{p}_{i,j}(\tau)\psi_{i,j}(\tau)\big) \\ &- \hat{p}_{i,j}(\tau)\psi_{i,j}(\tau)\Big) - Q_i(\tau)\psi_{i,j}(\tau) = 0. \end{aligned} \tag{31}$$

Proof. Let $\xi_i(\tau) = Z_i e_i(\tau) + Q_i(\tau)t_i(\tau)$. Plugging Eqs. (10) and (11) into $\xi_i(\tau)$, we can get $\partial^2\xi_i(\tau)/\partial f_i^2(\tau) = 2Q_i(\tau)w_i(\tau)/f_i^3(\tau) + Z_i\kappa_i w_i(\tau) > 0$. It is easy to know that $\xi_i(\tau)$ is a convex function w.r.t. $f_i(\tau)$. Thus, we can get the optimal solution through $\partial\xi_i(\tau)/\partial\hat{f}_i(\tau) = 0$ and obtain $\hat{f}_i(\tau) = \sqrt[3]{Q_i(\tau)/(2Z_i\kappa_i)}$.

Let $r_{i,j}(\tau) = \log_2\big(1 + p_{i,j}(\tau)\psi_{i,j}(\tau)\big)$. We have

$$\begin{aligned} \frac{\partial\xi_i(\tau)}{\partial r_{i,j}(\tau)} &= \frac{-Q_i(\tau)w_i(\tau)\delta_i(\tau)}{W_i r_{i,j}(\tau)^2} + \frac{Z_i w_i(\tau)\delta_i(\tau)}{W_i\psi_{i,j}(\tau)} \\ &\quad \times \frac{2^{r_{i,j}(\tau)}\big(\ln 2 r_{i,j}(\tau) - 1\big) + 1}{r_{i,j}(\tau)^2}, \end{aligned} \tag{32}$$

$$\begin{aligned} \frac{\partial^2\xi_i(\tau)}{\partial r_{i,j}(\tau)^2} &= \frac{2Q_i(\tau)w_i(\tau)\delta_i(\tau)}{W_i r_{i,j}(\tau)^3} + \frac{Z_i w_i(\tau)\delta_i(\tau)}{W_i\psi_{i,j}(\tau)} \\ &\quad \times \frac{2^{r_{i,j}(\tau)}\big(\ln^2 2 r_{i,j}(\tau)^2 - 2\ln 2 r_{i,j}(\tau) + 2\big) - 2}{r_{i,j}(\tau)^3}. \end{aligned} \tag{33}$$

Let $\zeta_i\big(r_{i,j}(\tau)\big) = 2^{r_{i,j}(\tau)}\big(\ln^2 2 r_{i,j}(\tau)^2 - 2\ln 2 r_{i,j}(\tau) + 2\big) - 2$. We have

$$\frac{\partial\zeta_i\big(r_{i,j}(\tau)\big)}{\partial r_{i,j}(\tau)} = 2^{r_{i,j}(\tau)}\ln^3 2 r_{i,j}(\tau)^3 \ge 0. \tag{34}$$

Hence, we know that $\zeta_i\big(r_{i,j}(\tau)\big) \ge \zeta_i(0) = 0$. Accordingly, we easily know that $\partial^2\xi_i(\tau)/\partial r_{i,j}(\tau)^2 > 0$. Therefore, $\xi_i(\tau)$ is a convex function w.r.t. $r_{i,j}(\tau)$. Then, we can get the optimal value $r_{i,j}^*(\tau)$ through $\partial\xi_i(\tau)/\partial r_{i,j}(\tau) = 0$. Plugging $\hat{p}_{i,j}(\tau) = (2^{r_{i,j}^*(\tau)} - 1)/\psi_{i,j}(\tau)$ into Eq. (32), we can obtain Eq. (31). Thus, the optimal value $\hat{p}_{i,j}(\tau)$ is the solution of $h\big(\hat{p}_{i,j}(\tau)\big) = 0$ and can be obtained by using binary search method [4]. Based on the above optimal solutions, C_1, and C_2, we get the theorem. □

Remark 2. As mentioned in Section 4.2.1, if the schedule actions are made at every time slot to greedily minimize the drift-plus-penalty function $\Delta(\tau) + V\mathcal{F}\big(q(\tau)\big)$, then $Q(\tau)$ is consistently pushed towards a lower congestion state, which intuitively maintains system stability. As shown in Theorem 5, the CPU frequency strategy satisfies $\hat{f}_i(\tau) = \sqrt[3]{Q_i(\tau)/(2Z_i\kappa_i)}$, where Z_i and κ_i are constants. It can be easily known that $\hat{f}_i(\tau), Q_i(\tau)$ increase and decrease at the same time. Therefore, for the long-term energy minimization problem, if we use the discrete CPU model, we can first get the minimal CPU frequency through $\partial\xi_i(\tau)/\partial\hat{f}_i(\tau) = 0$, i.e., $f_{i,min}(\tau) = \sqrt[3]{Q_i(\tau)/(2Z_i\kappa_i)}$. Unlike the short-term energy minimization problem, if $f_{i,min}(\tau) > f_{i,max}$, we set the suboptimal CPU frequency is $f_i^*(\tau) = f_{i,max}$. This is because the Lyapunov optimization method focuses on the long-term optimization. Meanwhile, the method tolerates that the latency of strategy is greater than the latency constraint $\bar{t}_{i,max}$ in a certain range. If $f_{i,min}(\tau) \in \mathcal{CPU}$, then $f_{i,min}(\tau)$ is the optimal CPU frequency strategy, i.e., $f_i^*(\tau) = f_{i,min}(\tau)$. If $f_{i,min}(\tau) \notin \mathcal{CPU}$ and $f_{i,min}(\tau) < f_{i,max}$, not only to maintain the same increase and decrease between $Q_i(\tau)$ and $f_i(\tau)$, but also to maintain system stability, the suboptimal CPU frequency $f_i^*(\tau)$ must satisfy $f_i^*(\tau) \geq f_{i,min}(\tau)$. Hence, the suboptimal CPU frequency is the smallest element in $\mathcal{CPU}$ that is greater than $f_{i,min}(\tau)$, i.e., $f_i^*(\tau) = \min\{f_i(\tau)|f_i(\tau) \geq f_{i,min}(\tau), f_i(\tau) \in \mathcal{CPU}\}$.

Similarly, although $f_i^*(\tau)$ may be a suboptimal solution of P5, $f_i^*(\tau)$ is the best CPU frequency strategy that UE$_i$ can really adopt.

Based on Theorems 4 and 5, we develop the long-term energy-optimal algorithm for UE$_i$ based on the Lyapunov optimization method. The algorithm is named EO-LY. As shown in Algorithm 4, it does not require any prior knowledge of the mobility characteristics to minimize the long-term energy consumption of UEs. The complexity of the algorithm is $O(TM\mathcal{K})$, where $\mathcal{K}$ is the maximum number of binary search iterations for obtaining $\hat{p}_{i,j}(\tau)$.

5. Latency minimization problem

5.1. Greedy strategy based algorithms

Similar to P1, if $I_i(\tau) = j$, the subproblem of $a_i(\tau)$ is

$$\text{P6}: \min_{\boldsymbol{s_i}(\tau)} t_i(\tau), \qquad (35)$$
$$s.t. \quad C_1, C_2, C_6, C_7.$$

5.1.1. The optimal solution of the latency minimization problem

For latency minimization problem, we can also obtain the optimal task offloading decision, CPU frequency, transmission power, and service migration strategies based on Theorem 1. The optimal strategies of $a_i(\tau)$ can be gotten according to the following theorem.

Theorem 6. *For latency minimization problem, the optimal strategies of $a_i(\tau)$ can be obtained from the following equations:*

$$f_i^*(\tau) = \min\{f_{i,max}, \tilde{f}_i'(\tau)\}, \qquad (36)$$
$$p_{i,j}^*(\tau) = \min\{p_{i,max}, \tilde{p}_{i,j}'(\tau)\}, \qquad (37)$$
$$\lambda_{i,j^*}^*(\tau) = \Phi\{t_i^{l^*}(\tau) > t_i^{j^*}(\tau)\}, \qquad (38)$$
$$v_{i,j,j^*}^*(\tau) = \Phi\{t_i^{j}(\tau) > t_i^{j^*}(\tau)\}, \qquad (39)$$

where $\tilde{f}_i'(\tau) = \sqrt{\bar{e}_{i,max}/\big(\kappa_i w_i(\tau)\big)}$, and $\tilde{p}_{i,j}'(\tau)$ is the solution of the following equation:

$$g\big(\tilde{p}_{i,j}'(\tau)\big) = \pi_{i,j}(\tau)\log_2\big(1 + \tilde{p}_{i,j}'(\tau)\psi_{i,j}(\tau)\big) - \tilde{p}_{i,j}'(\tau) = 0. \qquad (40)$$

Proof. If $\lambda_{i,0}(\tau) = 1$, $a_i(\tau)$ is executed by UE$_i$. Plugging Eq. (3) into $C_6 : e_i(\tau) \leq \bar{e}_{i,max}$, we have $f_i(\tau) \leq \sqrt{\bar{e}_{i,max}/(\kappa_i w_i(\tau))}$. Let $\tilde{f}_i'(\tau) = \sqrt{\bar{e}_{i,max}/\big(\kappa_i w_i(\tau)\big)}$. Thus, we know that there is a maximum CPU frequency $\tilde{f}_i'(\tau)$ that can satisfy the energy constraint C_6. According to $C_1 : f_i(\tau) \in [0, f_{i,max}]$, we can easily obtain the optimal CPU frequency strategy from $f_i^*(\tau) = \min\{f_{i,max}, \tilde{f}_i'(\tau)\}$.

If $\lambda_{i,j}(\tau) = 1$, $a_i(\tau)$ is executed by MEC$_j$. Plugging Eq. (5) into C_6, we have the following inequality

$$\pi_{i,j}(\tau)\log_2\big(1 + p_{i,j}(\tau)\psi_{i,j}(\tau)\big) \geq p_{i,j}(\tau), \qquad (41)$$

where $\pi_{i,j}(\tau) = W_i\Big(\bar{e}_{i,max} - p_{i,0}\big(t_{i,j,e}(\tau) + t_{i,j,w}(\tau) + v_{i,j',j}(\tau)t_{i,j,m}(\tau)\big)\Big)/w_i(\tau)\delta_i(\tau)$. We then introduce

$$g\big(p_{i,j}(\tau)\big) = g_1\big(p_{i,j}(\tau)\big) - g_2\big(p_{i,j}(\tau)\big), \qquad (42)$$

where $g_1\big(p_{i,j}(\tau)\big) = \pi_{i,j}(\tau)\log_2\big(1 + p_{i,j}(\tau)\psi_{i,j}(\tau)\big)$ and $g_2\big(p_{i,j}(\tau)\big) = p_{i,j}(\tau)$. As can be seen in Fig. 4, if $g_1(p_{i,max}) > g_2(p_{i,max})$, $p_{i,j}^*(\tau) = p_{i,max}$. Otherwise, $p_{i,j}^*(\tau) = \tilde{p}_{i,j}'(\tau)$, where $\tilde{p}_{i,j}'(\tau)$ is the solution of $g\big(p_{i,j}(\tau)\big) = 0$. Moreover, because $g'\big(p_{i,j}(\tau)\big) = \pi_{i,j}(\tau)\psi_{i,j}(\tau)/\Big(\ln 2\big(1 + p_{i,j}(\tau)\psi_{i,j}(\tau)\big)\Big) - 1$, then $g'\big(p_{i,j}(\tau)\big) \leq 0$ and $g\big(p_{i,j}(\tau)\big)$ is a monotonic non-increasing function w.r.t. $p_{i,j}(\tau)$ when $p_{i,j}(\tau) \geq \big(\pi_{i,j}(\tau)\psi_{i,j}(\tau)/\ln 2 - 1\big)/\psi_{i,j}(\tau)$. Therefore, we can obtain $\tilde{p}_{i,j}'(\tau)$ by using binary search method [4]. According to $C_2 : p_{i,j}(\tau) \in [0, p_{i,max}]$, we get the optimal transmission power strategy from $p_{i,j}^*(\tau) = \min\{\tilde{p}_{i,j}'(\tau), p_{i,max}\}$.

Since $t_i(\tau)$ is a linear function w.r.t. $\lambda_{i,j}(\tau)$, we can obtain the offloading decision from $\lambda_{i,j}^*(\tau) = \Phi\{t_i^{l^*}(\tau) > t_i^{j^*}(\tau)\}$.

When the cost of service migration is less than the benefit of service migration, the service migration operation is triggered. We can iterate all MEC servers and find the optimal MEC server with minimal latency MEC$_{j^*}$. Then, we can obtain the optimal service migration strategy from $v_{i,j,j^*}^*(\tau) = \Phi\{t_i^{j}(\tau) > t_i^{j^*}(\tau)\}$. □

Remark 3. Similar to the short-term energy minimization problem, if we use discrete CPU frequencies, for the short-term latency minimization problem, we can first obtain the maximal CPU frequency from $\tilde{f}_{i,max}'(\tau) = \sqrt{\bar{e}_{i,max}/\big(\kappa_i w_i(\tau)\big)}$. If $\tilde{f}_{i,max}'(\tau) > f_{i,max}$, we can conclude that $f_{i,max}$ will not exceed the energy consumption constraint. Thus, we have the suboptimal CPU frequency $f_i^*(\tau) = f_{i,max}(\tau)$. Moreover, if $\tilde{f}_{i,max}'(\tau) \in \mathcal{CPU}$, then $\tilde{f}_{i,max}'(\tau)$ is the optimal CPU frequency strategy, i.e., $f_i^*(\tau) = \tilde{f}_{i,max}'(\tau)$. If $\tilde{f}_{i,max}'(\tau) \notin \mathcal{CPU}$ and $\tilde{f}_{i,max}'(\tau) < f_{i,max}$, although we cannot obtain the optimal CPU frequency strategy directly, it can be confirmed that the suboptimal CPU frequency $f_i^*(\tau)$ should satisfy $f_i^*(\tau) \leq \tilde{f}_{i,max}'(\tau)$. Hence, the suboptimal CPU frequency strategy is the largest element in $\mathcal{CPU}$ that is less than $\tilde{f}_{i,max}'(\tau)$, i.e., $f_i^*(\tau) = \max\{f_i(\tau)|f_i(\tau) \leq \tilde{f}_{i,max}'(\tau), f_i(\tau) \in \mathcal{CPU}\}$.

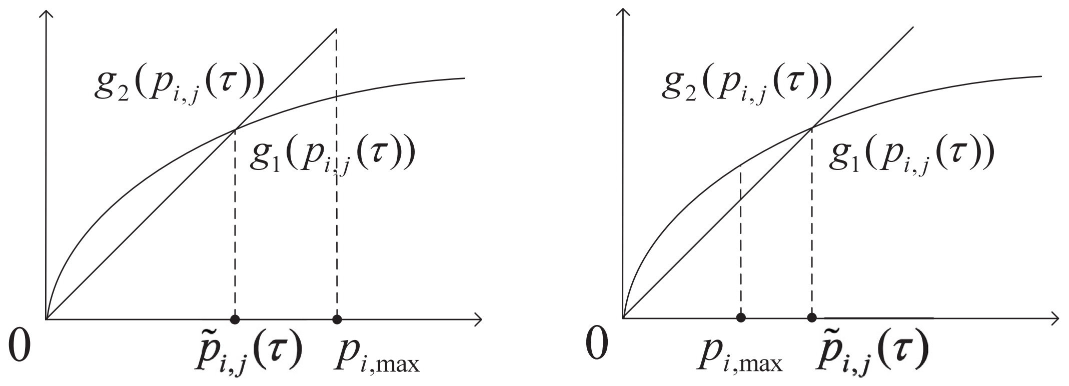
Similarly, although $f_i^*(\tau)$ may be a suboptimal solution of P6, $f_i^*(\tau)$ is the best CPU frequency strategy that UE$_i$ can really adopt.

5.1.2. The algorithms for UEs with different mobility characteristics

Similarity, we can use the mobility characteristics to optimize the strategy of UE. We can adopt Algorithms 1, 2, and 3 to optimize the latency based on the UE's mobility characteristics. However, in the algorithms, we replace the energy $e_i(\tau)$ with latency $t_i(\tau)$. Moreover, we name the algorithms as LO-RM, LO-PM, and LO-KM, respectively.

5.2. Lyapunov optimization based algorithm

Same as P1, we introduce virtual queue $Q_i'(\tau+1) = \max\{Q_i'(\tau) - \bar{e}_{i,max} + e_i(\tau), 0\}$. We also assume that $Q_i'(0) = 0$. Then, we have the following theorem.



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

Fig. 4. The illustrations of $p^*_{i,j}(\tau)$.

Theorem 7. *P2 is equivalent to the following problem*

$$\text{P7}: \min_{\mathbf{s_i}(\tau)} Z_i' t_i(\tau) + Q_i'(\tau) e_i(\tau), \tag{43}$$

$$s.t. \;\; C_1, C_2, C_4, C_7, \tag{44}$$

where $Z_i' > 0$ is the weight parameter that indicates the importance of how much we emphasize latency of UE$_i$.

Proof. Similar to Theorem 4, we first obtain Lyapunov function, conditional Lyapunov drift, and Lyapunov drift-plus-penalty function of P2. Then, we can get the conclusion with the help of the law of telescoping sums. Due to the space of paper, the detailed proof is omitted. □

The solution of P7 is an approximate optimal solution of P2. Meanwhile, the average time service latency deviates by at most $O(1/Z_i')$ from the optimal solution of P2, with the average queue backlog bounded of $O(Z_i')$ [20]. The optimal solutions $\lambda^*_{i,j^*}(\tau)$, $f_i^*(\tau)$, $p^*_{i,j}(\tau)$, and $v^*_{i,j',j^*}(\tau)$ of P7 can be obtained according to the following theorem.

Theorem 8. *For MEC$_j$ and UE$_i$, the optimal strategies of P7 can be obtained from the following equations:*

$$f_i^*(\tau) = \min\{f_{i,max}, \hat{f}_i'(\tau)\}, \tag{45}$$

$$p^*_{i,j}(\tau) = \min\{p_{i,max}, \hat{p}'_{i,j}(\tau)\}, \tag{46}$$

$$\lambda^*_{i,j^*}(\tau) = \Phi\{t_i^{l^*}(\tau) > t_i^{j^*}(\tau)\}, \tag{47}$$

$$v^*_{i,j',j^*}(\tau) = \Phi\{t_i^{j'}(\tau-1) > t_i^{j^*}(\tau)\}, \tag{48}$$

where $\hat{f}_i'(\tau) = \sqrt[3]{Q_i'(\tau)/(2Z_i'\kappa_i)}$, and $\hat{p}'_{i,j}(\tau)$ is obtained from the following equation:

$$h\big(\hat{p}'_{i,j}(\tau)\big) = Z_i' \Big(\big(1+\hat{p}'_{i,j}(\tau)\psi_{i,j}(\tau)\big)\ln\big(1+\hat{p}'_{i,j}(\tau)\psi_{i,j}(\tau)\big) - \hat{p}'_{i,j}(\tau)\psi_{i,j}(\tau) \Big) - Q_i'(\tau)\psi_{i,j}(\tau) = 0. \tag{49}$$

Proof. Compared with Theorem 4, we can easily find that the multipliers of $e_i(\tau)$ and $t_i(\tau)$ are reversed actually. We replace the task latency with task energy consumption as the backlog of the virtual queue that we want to make stable. Moreover, the above operations do not change the convex property of P7. Therefore, we can adjust the place of related parameters (i.e., Z_i' and $Q_i'(\tau)$) of $\hat{f}_i'(\tau)$ and $h\big(\hat{p}'_{i,j}(\tau)\big)$, thus obtaining the optimal solutions of P7. Therefore, we get the theorem. □

Similarity, we can replace energy consumption used in Algorithms 4 with latency as the optimization objective to make the strategy for UEs. The algorithm is named as LO-LY accordingly.

Remark 4. Similar to the long-term energy minimization problem, if we use discrete CPU frequencies, for the long-term latency minimization problem, we can first get the minimal CPU frequency from Theorem 8, i.e., $f_{i,min}(\tau) = \sqrt[3]{Q_i'(\tau)/(2Z_i'\kappa_i)}$. If $f_{i,min}(\tau) > f_{i,max}$, we set the suboptimal CPU frequency is $f_i^*(\tau) = f_{i,max}$. If $f_{i,min}(\tau) \in \mathcal{CPU}$, then $f_{i,min}(\tau)$ is the optimal CPU frequency strategy, i.e., $f_i^*(\tau) = f_{i,min}(\tau)$. If $f_{i,min}(\tau) \notin \mathcal{CPU}$ and $f_{i,min}(\tau) < f_{i,max}$, the suboptimal CPU frequency is the smallest element in $\mathcal{CPU}$ that is greater than $f_{i,min}(\tau)$, i.e., $f_i^*(\tau) = \min\{f_i(\tau)|f_i(\tau) \ge f_{i,min}(\tau), f_i(\tau) \in \mathcal{CPU}\}$.

Similarly, although $f_i^*(\tau)$ may be a suboptimal solution of P7, $f_i^*(\tau)$ is the best CPU frequency strategy that UE$_i$ can really adopt.

6. Simulation experiments and results analysis

6.1. Experiment setting

In the experiment, referring to [12], we generate $N = 3$ UEs and $M = 200$ MEC servers according to the following parameters: $p_{i,max} = 3$ W, $p_{i,0} = 0.01$ W, $f_{i,max} = 5 \times 10^8$ cycles/s, $\bar{t}_{i,max} = 1$ s, $\bar{e}_{i,max} = 1$ J, $w_i(\tau)$ is a random value taken from $\{10, 20, 30, 40\}$, $\delta_i(\tau) = 1048576$ bits/cycle, $\kappa_i = 10^{-10}$, $W_i = 10^{11}$ Hz, $h_i = 10^{-3}$, $\omega_i = 2$, $N_i = 10^{-9}$, $m_i = 10^{-6}$ s, $x_i(0) = y_i(0) = 0$. The computing power of MEC$_j$ is given as $f_j = 10^9$ cycles/s. Without loss of generality, we set $t_{i,j,w}(\tau) = 0$. UEs move on a 400×400 square meters two-dimensional plane and $-200 \le x_i(\tau), y_i(\tau) \le 200$. To simulate the movement of UEs, we introduce a random variable $u(\tau) \in \{-1, 0, 1\}$ to indicate UE remains stationary, or moves 1 meters (m) forward or backward. The three UEs adopt the different transportation means, including walk, bike, and motorcycle. The horizontal and vertical speeds (i.e., $v_{i,x}$ and $v_{i,y}$) of UEs are as follows: $v_{1,x} = v_{1,y} = 1$ m/s, $v_{2,x} = v_{2,y} = 3$ m/s, and $v_{3,x} = v_{3,y} = 5$ m/s. The location of UE$_i$ at $\tau + 1$ is $\big(x_i(\tau+1), y_i(\tau+1)\big)$, where $x_i(\tau+1) = \max\big\{\min\{x_i(\tau-1) + v_{i,x}u(\tau), 200\}, -200\big\}$ and $y_i(\tau+1) = \max\big\{\min\{y_i(\tau-1) + v_{i,y}u(\tau), 200\}, -200\big\}$. The MEC servers are located on the diagonal of the plane, and their horizontal and vertical coordinate intervals are the multiple of 4 m. In addition, due to different magnitude of energy and latency. For energy minimization problem, we simulate $T = 10^5$ time slots. For latency minimization problem, we simulate $T = 10^9$ time slots.

We use the following four task offloading schemes as baselines: (1) LM: UE$_i$ executes all tasks locally by using $f_{i,max}$. (2) LR: UE$_i$ executes all tasks locally by using the CPU strategies proposed in this paper. (3) MM: All tasks will be offloaded to the MEC servers with $p_{i,max}$. (4) MR: All tasks will be offloaded to the MEC servers for executing by using the transmission power strategies proposed in this paper.

Table 1
The average number of iterations comparison of UE$_i$ using different algorithms.

Algorithms	Energy minimization ($T = 10^5$)			Latency minimization ($T = 10^9$)		
	UE$_1$	UE$_2$	UE$_3$	UE$_1$	UE$_2$	UE$_3$
EO/LO-RM	200	200	200	200	200	200
EO/LO-PM	205	211	218	212	233	221
EO/LO-KM	314	257	239	319	367	254
EO/LO-LY4	10,875	11,234	10,341	12,503	12,503	12,233
EO/LO-LY6	11,884	11,406	11,188	12,232	12,232	12,504
EO/LO-LY8	13,063	11,651	11,582	11,501	11,501	11,503

Table 2
The cost and performance comparison of UE$_i$ using different algorithms.

Algorithms	Energy/cost (J, $T = 10^5$)			Latency/performance (s, $T = 10^9$)		
	UE$_1$	UE$_2$	UE$_3$	UE$_1$	UE$_2$	UE$_3$
LM	1E+09	1E+09	1E+09	8E−08	8E−08	8E−08
LR	1.587E−03	1.587E−03	1.587E−03	6.35E−02	6.35E−02	6.35E−02
MM	1.145E−04	1.145E−05	1.14E−05	4.852E−06	4.852E−06	4.836E−06
MR	4.053E−07	4.031E−07	3.917E−07	4.208E−06	4.271E−06	4.384E−06
EO/LO-RM	3.945E−07	3.93E−07	3.819E−07	4.208E−06	4.271E−06	4.384E−06
EO/LO-PM	3.943E−07	3.925E−07	3.814E−07	4.188E−06	4.249E−06	4.384E−06
EO/LO-KM	3.941E−07	3.925E−07	3.814E−07	4.181E−06	4.249E−06	4.384E−06
EO/LO-LY4	3.182E−06	1.527E−06	4.258E−06	8.178E−05	8.168E−05	8.062E−05
EO/LO-LY8	3.995E−07	9.073E−07	9.139E−07	1.032E−05	1.053E−05	1.034E−05
EO/LO-LY12	8.569E−08	8.66E−08	8.642E−08	4.208E−06	4.271E−06	4.324E−06

6.2. The convergence of the algorithms

Table 1 shows the average number of iterations required for UE$_i$ using different algorithms to obtain its optimal strategy. The number of iterations refers to the number of loops required to search $\lambda^*_{i,j}(\tau), f^*_i(\tau), p^*_{i,j}(\tau)$, and $\upsilon^*_{i,j',j}(\tau)$. In the table, EO-LY4 represents EO-LY ($Z_i = 10^4$) and LO-LY4 is LO-LY ($Z'_i = 10^4$). Other similar symbols indicate similar meanings. As shown in the table, for LO-RM/PM/KM, since the transmission power is obtained by using binary search method, the average number of iterations is more than what EO-RM/PM/KM needs. LO/EO-LY needs more iterations than LO/EO-RM/PM/KM. The reason lies in that LO/EO-LY does not determine the available MEC servers set in advance and should iterate all servers at each time slot. Meanwhile, the value of $h\big(\hat{p}_{i,j}(\tau)\big)$ or $h'\big(\hat{p}'_{i,j}(\tau)\big)$ is larger than $g\big(p_{i,j}(\tau)\big)$, so LO/EO-LY takes more iterations to make transmission power strategy when using the binary search method. Therefore, we can also find that the value of Z_i or Z'_i affects the number of iterations. For LO-LY, $h'\big(\hat{p}'_{i,j}(\tau)\big)$ decreases as Z'_i increases, thus the algorithm with smaller Z'_i requires less iterations. However, for EO-LY, increasing the value of Z_i has opposite effects on the number of iterations.

6.3. The effectiveness of algorithms in the long-term

As shown in Table 2, although LM achieves the minimal latency among all the algorithms, the energy consumption of LM exceeds the energy consumption constraint of UEs. It can be seen from the table that compared with the four baselines, the four algorithms proposed in this paper perform better. Moreover, if the future location of the UE can be known in advance, the service migration strategy can be pre-determined to further minimize the cost. Thus, compared with EO/LO-RM, EO/LO-PM/KM can further reduce the energy and latency of UEs by using their mobility characteristics.

Z_i and Z'_i are the weight parameter that indicates the importance of how much we emphasize the energy and latency of UE$_i$, respectively. Thus, we can see that the energy and latency of UE$_i$ decreases as Z_i and Z'_i increases. Moreover, as shown in Fig. 5, it can be seen that the number of service migrations decreases as Z_i or Z'_i increases, thus reducing the service migration amount, energy, and latency of UEs.

Next, we analyze the impact of $\bar{t}_{i,max}$ and $\bar{e}_{i,max}$ on the energy and latency, respectively. Let us take UE$_3$ as an example. From the perspective of the curves in Fig. 6, as the constraints tighten, the energy and latency gradually increase. However, EO/LO-LY performs better than EO/LO-RM/PM/KM.

It is well known that the short-term and long-term are relative concepts. Thus, in this paper, energy minimization problem is a long-term optimization problem when $T \geq 10^5$, and latency minimization problem is a long-term optimization problem when $T \geq 10^9$. Based on the above, we can conclude that EO/LO-LY performs better than EO/LO-RM/PM/KM in the long-term.

In addition, as shown in the above tables and figures, we find that the transportation means affects the velocity, resulting in different movement distance and $\mathbf{M}_i(\tau)$. Therefore, the transportation mean affects the effectiveness of the algorithms.

6.4. The effectiveness of algorithms in the short-term

As can be seen from Fig. 7(a), the energy consumption of UE$_i$ using EO-RM/PM/KM increases as T increases. In addition, in Fig. 7(b), the latency of UE$_i$ increases as T increases. The reason is that the movement of UE$_i$ increases the number of service migrations, thereby increasing the energy and latency. Let us take UE$_3$ as an example. For EO-LY, the energy consumption of the UE decreases as T increases. However, as shown in Fig. 8(a), the latency of UE$_3$ exceeds the $\bar{t}_{3,max}$ when $T < 10^5$. Thus, EO-RM/PM/KM performs better than EO-LY in the short-term (i.e., $T \leq 10^5$). Since the energy consumption constraint, it can also be seen from Figs. 7(b) and 8(b) that LO-RM/PM/KM performs better than LO-LY in the short-term (i.e., $T \leq 10^9$).

6.5. The impact of other variables

6.5.1. The impact of δ_i

The data size per CPU clock cycle of $a_i(\tau)$ is denoted by $\delta_i(\tau)$ (bits per cycle). This means that the increase of $\delta_i(\tau)$ directly affects the latency and energy consumption during the data transmission process. It can be seen from Figs. 9(a) and 9(b) that as the increasing of δ_i, the average energy consumption and average latency per time slot are also increasing. Therefore, the optimal execution location for UE's task with high δ_i is not always in MEC servers, but sometimes in local.

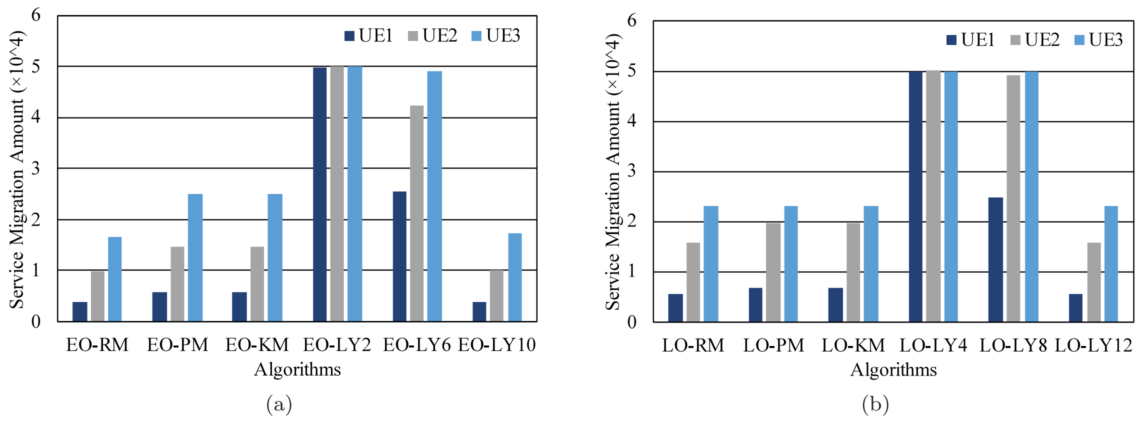


Fig. 5. (a) The impact of Z_i on the number of service migrations. (b) The impact of Z'_i on the number of service migrations.

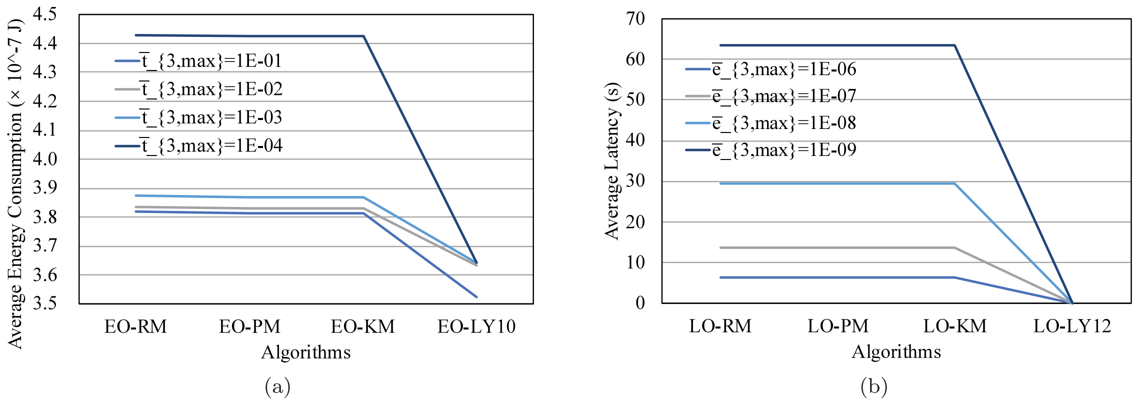


Fig. 6. (a) The impact of $\bar{t}_{3,max}$ on the energy consumption. (b) The impact of $\bar{e}_{3,max}$ on the latency.

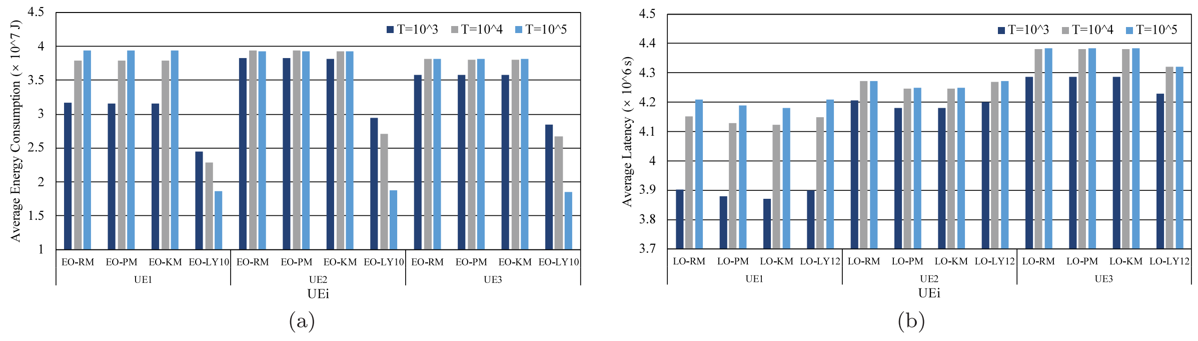


Fig. 7. (a) The impact of T on the energy consumption. (b) The impact of T on the latency.

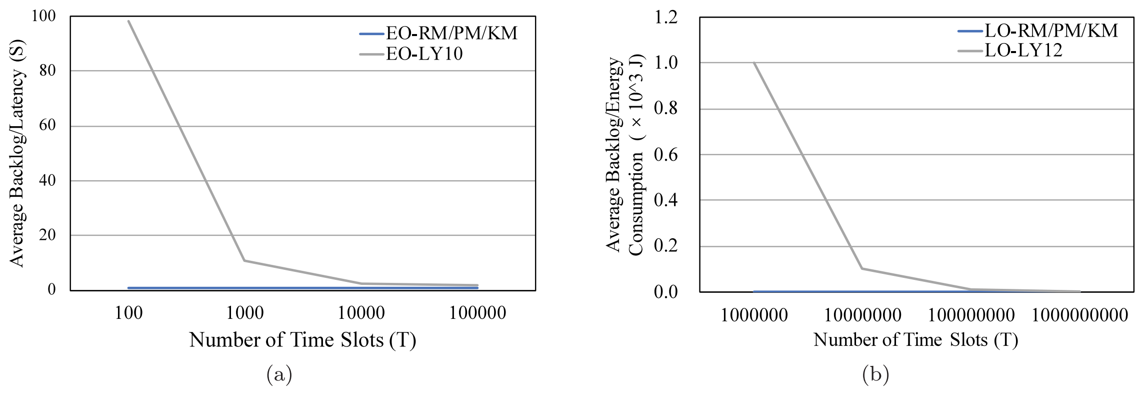


Fig. 8. (a) The impact of T on the average backlog of $Q_3(\tau)$. (b) The impact of T on the average backlog of $Q'_3(\tau)$.

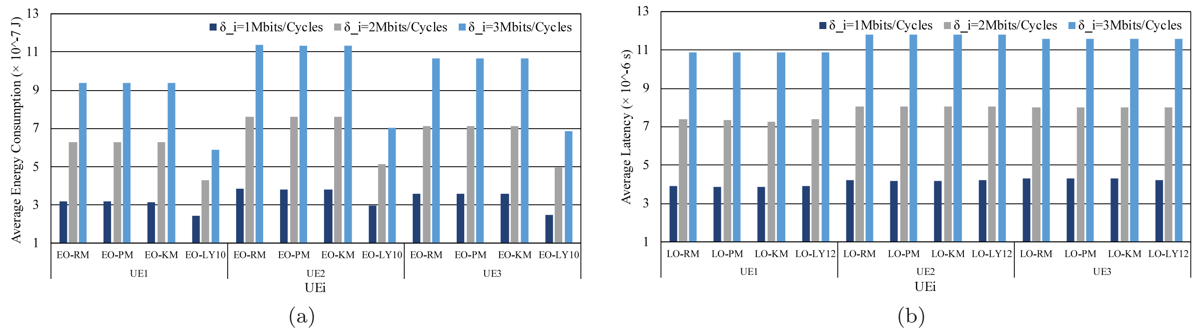


Fig. 9. (a) The impact of δ_i on the energy consumption. (b) The impact of δ_i on the latency.

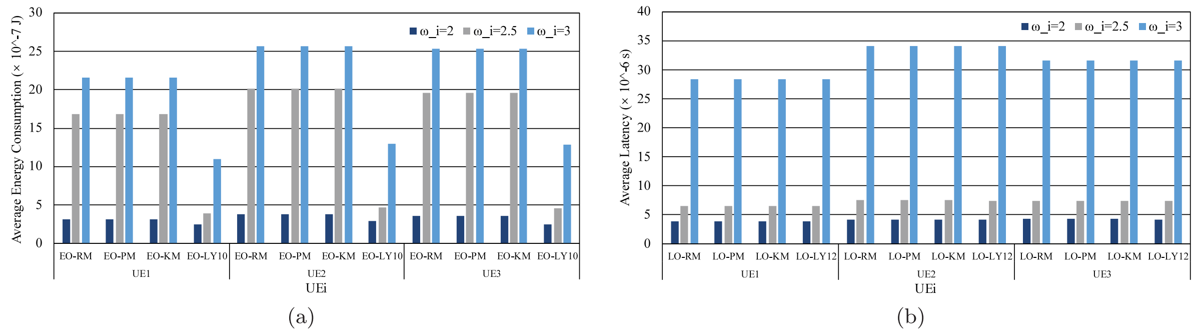


Fig. 10. (a) The impact of ω_i on the energy consumption. (b) The impact of ω_i on the latency.

6.5.2. The impact of ω_i

As shown in Eq. (1), ω_i represents the communication channel path loss between UE$_i$ and MEC$_j$. In addition, with the increase of ω_i, the transmission rate between the UE and the server decreases, thus increasing the transmission delay for the UE uploading its tasks. It can be seen from Figs. 10(a) and 10(b) that the increase of ω_i causes an increase in energy consumption and latency of UEs. It can also be known from the figures that the stable and high-speed wireless communication greatly reduce the energy and latency for UEs performing various applications, thus improving QoS and QoE.

6.5.3. The impact of m_i

m_i represents the migration delay of VM$_i$ and directly affects the quality of seamless service providing to UEs with mobility. We can see from Figs. 11(a) and 11(b) that the decrease of m_i causes the decrease in energy consumption and latency of UEs. The lightweight virtualization technology adopted by MEC, such as virtual network function [8], can make the fast service migration a reality.

6.5.4. The impact of M

In MEC, the servers with limited resources are deployed proximity to UEs to save energy or latency of the UEs. Intuitively, an increase in the number of MEC servers within an area can improve QoE. The reason is that increasing the number of servers will allow a UE to choose servers closer to itself, thereby reducing the latency and energy consumption for transmitting task. It can be seen from Figs. 12(a) and 12(b) that as M increases, the energy consumption and latency of UEs decrease, which is consistent with the real world.

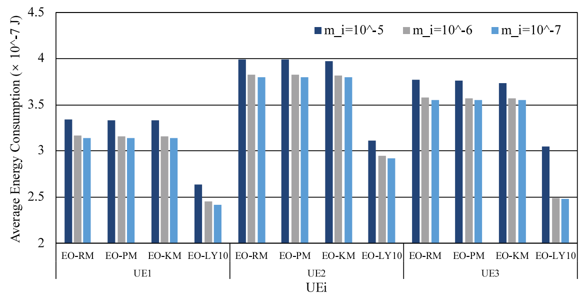
6.5.5. The impact of N

Although the paper assumes that there are no resource competitions between UEs, it is meaningful to explore the effectiveness of the proposed algorithms in a multiple UEs scenario. In this experiment, we create more UEs randomly based on the former parameter generation equations. As shown in Fig. 13, due to the different workload and moving speed of UEs, it seems that the change in the number of UEs causes the change in the average energy consumption and latency of the UEs. It should be noted that the increase of N does not necessarily mean an increase in average energy consumption or a decrease in average latency. The reason lies in that there is no competition of MEC server resources between UEs, that is, a UE's strategy is not affected by other UEs in this paper. However, constrained by the maximum energy or latency of UEs, it can be confirmed again from Figs. 13(a) and 13(b) that EO/LO-RM/PM/KM performs better than EO/LO-LY in the short-term. Moreover, it can also be known that EO/LO-LY performs better than EO/LO-RM/PM/KM in the long-term.

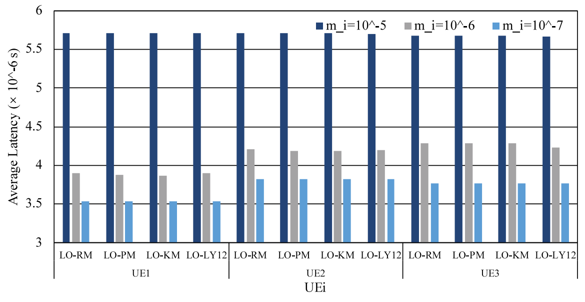
7. Conclusions

In this paper, we formulate an energy minimization and a latency minimization problems respectively, and develop algorithms to optimize the UE's cost and performance in the short- and long-term. We propose three mobility types depending on whether the mobility characteristics of UEs are known, and develop the greedy strategy based task offloading algorithms for the UEs to optimize their cost and performance in a short-term by using their mobility characteristics. To deal with the challenge of acquiring UE's mobility characteristics over a long time, we then use a Lyapunov optimization method to develop the algorithms that do not require any prior knowledge of the mobility characteristics to optimize the long-term cost and performance of UEs. Experimental results show that the greedy strategy based algorithms perform better than the Lyapunov optimization method based algorithms in the short-term. However, the Lyapunov optimization method based algorithms perform better than the greedy strategy based algorithms over the long-term, especially when the mobility characteristics of UE cannot be known in advance.

Deploying the algorithms on actual MEC system involves many challenging issues, such as caching, virtualization technology,

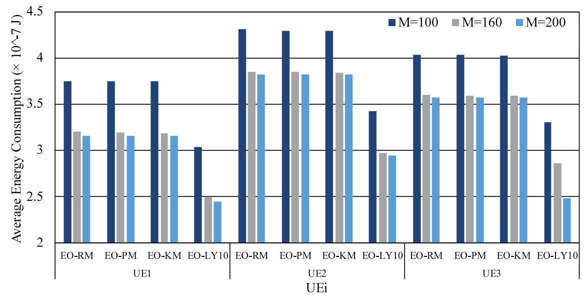


(a)

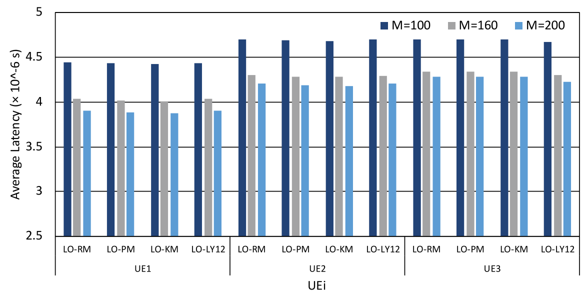


(b)

Fig. 11. (a) The impact of m_i on the energy consumption. (b) The impact of m_i on the latency.

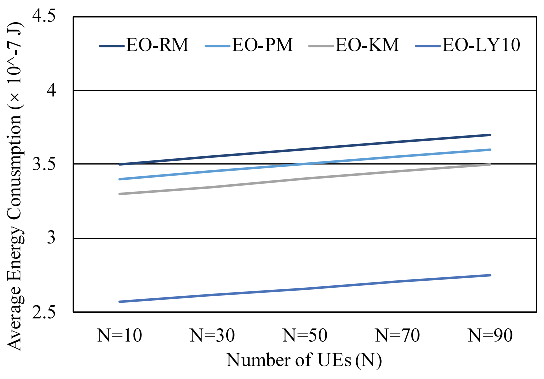


(a)

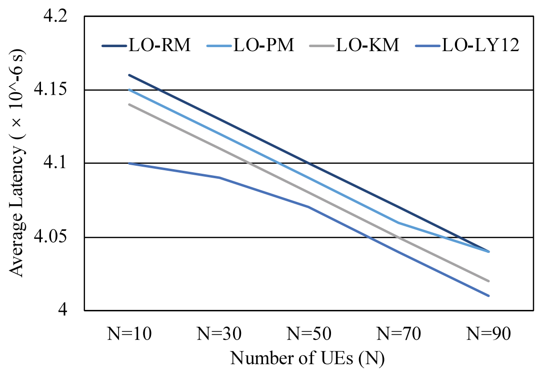


(b)

Fig. 12. (a) The impact of M on the energy consumption. (b) The impact of M on the latency.



(a)



(b)

Fig. 13. (a) The impact of N on the energy consumption. (b) The impact of N on the latency.

creating interactive protocols between the different entities, and ensuring the stability of communication channel. Since the study of the above issues is beyond the scope of our work, we evaluate the algorithms through simulation experiments. Moreover, this paper only investigates three simple mobility types. Thus, more complicated mobility characteristics should be further investigated in the future. Furthermore, the above issues should be considered in the system model, thus developing algorithms that can be deployed in the real world.

CRediT authorship contribution statement

Yan Ding: Conceptualization, Methodology, Writing - original draft. **Kenli Li:** Methodology, Supervision, Writing - review & editing. **Chubo Liu:** Methodology, Writing - original draft. **Zhuo Tang:** Investigation, Validation. **Keqin Li:** Conceptualization, Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments have greatly helped to improve this manuscript. A preliminary version of the paper was presented on the 17th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2019), Xiamen, China, 16-18 December 2019. This work was partially supported by the National Outstanding Youth Science Program of National Natural Science Foundation of China (Grant No. 61625202), National Key Research and Development Program of China (2018YFB1701403), Programs of National Natural Science Foundation of China (Grant Nos. 61876061, 62072165, U19A2058, 61702170), and Postgraduate Scientific Research Innovation Project of Hunan Province (Grant No. CX20200435). This work was also sponsored by Zhijiang Lab, China (No. 2020KE0AB01).

References

[1] M. Aazam, S. Zeadally, K.A. Harras, Offloading in fog computing for iot: Review, enabling technologies, and research opportunities, Future Gener. Comput. Syst. 87 (2018) 278–289.

[2] M.V. Barbera, S. Kosta, A. Mei, J. Stefa, To offload or not to offload? the bandwidth and energy costs of mobile cloud computing, in: Proceedings of the IEEE INFOCOM, IEEE, Turin, Italy, 2013, pp. 1285–1293.

[3] A. Bhattacharya, P. De, A survey of adaptation techniques in computation offloading, J. Netw. Comput. Appl. 78 (2017) 97–115.

[4] Vandenberghe Boyd, Faybusovich, Convex Optimization, Cambridge University Press, Cambridge, U.K, 2004.

[5] S. Cao, X. Tao, Y. Hou, Q. Cui, An energy-optimal offloading algorithm of mobile computing based on hetnets, in: International Conference on Connected Vehicles and Expo, ICCVE, IEEE, Shenzhen, China, 2015, pp. 254–258.

[6] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (5) (2015) 2795–2808.

[7] W. Chen, D. Wang, K. Li, Multi-user multi-task computation offloading in green mobile edge cloud computing, IEEE Trans. Serv. Comput. 12 (5) (2019) 726–738.

[8] A. De Domenico, Y. Liu, W. Yu, Optimal virtual network function deployment for 5g network slicing in a hybrid cloud infrastructure, IEEE Trans. Wireless Commun. 1 (2020) 1–16, http://dx.doi.org/10.1109/TWC.2020.3017628.

[9] Y. Ding, C. Liu, K. Li, Z. Tang, K. Li, Task offloading and service migration strategies for user equipments with mobility consideration in mobile edge computing, in: 17th IEEE International Symposium on Parallel and Distributed Processing with Applications ISPA, IEEE, Xiamen, China, 2019, pp. 176–183.

[10] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, M. Satyanarayanan, Adaptive vm Handoff Across Cloudlets, Technical Report CMU-CS-15-113, 2015.

[11] W. Labidi, M. Sarkiss, M. Kamoun, Energy-optimal resource scheduling and computation offloading in small cell networks, in: 22nd International Conference on Telecommunications, ICT, IEEE, Sydney, Australia, 2015, pp. 313–318.

[12] K. Li, A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing, IEEE Trans. Sustain. Comput. 1 (2018) 1–10, http://dx.doi.org/10.1109/TSUSC.2018.2868655.

[13] K. Li, Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing, IEEE Trans. Sustain. Comput. 1 (2019) 1–14, http://dx.doi.org/10.1109/TSUSC.2019.2904680.

[14] Y. Li, S. Xia, m. Zheng, B. Cao, Q. Liu, Lyapunov optimization based trade-off policy for mobile cloud offloading in heterogeneous wireless networks, IEEE Trans. Cloud Comput. 1 (2019) 1–14, http://dx.doi.org/10.1109/TCC.2019.2938504.

[15] W. Lu, X. Meng, G. Guo, Fast service migration method based on virtual machine technology for MEC, IEEE Internet Things J. 6 (3) (2019) 4344–4354.

[16] L. Ma, S. Yi, Q. Li, Efficient service handoff across edge servers via docker container migration, in: J. Zhang, M. Chiang, B.M. Maggs (Eds.), Proceedings of the Second ACM/IEEE Symposium on Edge Computing, ACM, San Jose, Silicon Valley, SEC, CA, USA, 2017, pp. 11:1–11:13.

[17] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1628–1656.

[18] Y. Mao, J. Zhang, K.B. Letaief, A lyapunov optimization approach for green cellular networks with hybrid energy supplies, IEEE J. Sel. Areas Commun. 33 (12) (2015) 2463–2477.

[19] 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.

[20] M. Neely, Stochastic Network Optimization with Application to Communication and Queueing Systems, Morgan and Claypool Press, 2010.

[21] I. Nonaka, A dynamic theory of organizational knowledge creation, Organ. Sci. 5 (1) (1994) 14–37.

[22] T. Ouyang, Z. Zhou, X. Chen, Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing, IEEE J. Sel. Areas Commun. 36 (10) (2018) 2333–2345.

[23] J. Plachy, Z. Becvar, E.C. Strinati, Dynamic resource allocation exploiting mobility prediction in mobile edge computing, in: 27th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC, IEEE, Valencia, Spain, 2016, pp. 1–6.

[24] C. Shen, C. Tekin, M. Van Der Schaar, A non-stochastic learning approach to energy efficient mobility management, IEEE J. Sel. Areas Commun. 34 (12) (2016) 3854–3868.

[25] B. Sklar, Rayleigh fading channels in mobile digital communication systems. i. characterization, IEEE Commun. Mag. 35 (7) (1997) 90–100.

[26] Y. Sun, S. Zhou, J. Xu, EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2637–2646.

[27] T. Taleb, A. Ksentini, An analytical model for follow me cloud, in: 2013 IEEE Global Communications Conference, GLOBECOM, IEEE, Atlanta, GA, USA, 2013, pp. 1291–1296.

[28] T. Taleb, A. Ksentini, P.A. Frangoudis, Follow-me cloud: When cloud services follow mobile users, IEEE Trans. Cloud Comput. 7 (2) (2019) 369–382.

[29] H. Tout, C. Talhi, N. Kara, A. Mourad, Smart mobile computation offloading: Centralized selective and multi-objective approach, Expert Syst. Appl. 80 (2017) 1–13.

[30] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, K.K. Leung, Dynamic service placement for mobile micro-clouds with predicted future costs, IEEE Trans. Parallel Distrib. Syst. 28 (4) (2017) 1002–1016.

[31] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, R. Wang, User mobility aware task assignment for mobile edge computing, Future Gener. Comput. Syst. 85 (2018) 1–8.

[32] Q. Wu, X. Chen, Z. Zhou, L. Chen, Mobile social data learning for user-centric location prediction with application in mobile edge service migration, IEEE Internet Things J. 6 (5) (2019) 7737–7747.

[33] L. Yang, J. Cao, H. Cheng, Y. Ji, Multi-user computation partitioning for latency sensitive mobile cloud applications, IEEE Trans. Comput. 64 (8) (2015) 2253–2266.

[34] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, Y. Paek, Techniques to minimize state transfer costs for dynamic execution offloading in mobile cloud computing, IEEE Trans. Mob. Comput. 13 (11) (2014) 2648–2660.

[35] F. Yu, H. Chen, J. Xu, Dmpo: Dynamic mobility-aware partial offloading in mobile edge computing, Future Gener. Comput. Syst. 89 (2018) 722–735.

[36] W. Zhang, Y. Wen, K. Guan, K. Dan, H. Luo, D.O. Wu, Energy-optimal mobile cloud computing under stochastic wireless channel, IEEE Trans. Wireless Commun. 12 (9) (2013) 4569–4581.

Yan Ding received his B.S. degree in Software Engineering from North University of China in 2014, and M.S. degree in Computer Application Technology from Xinjiang University, Urumqi, China in 2018. He is currently pursuing the Ph.D. degree with the Hunan University, China. His research interests include mobile edge computing, data analysis, machine learning, and network security. He has published three papers in journals and conference, including IEEE Transactions on Industrial Informatics, Computers & Security, and the 17th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA 2019). He is a student member of IEEE and CCF.

Kenli Li received the Ph.D. degree in computer science from Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar at the University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently a full professor of computer science and technology at Hunan University, the dean of the College of Information Sciences and Engineering of Hunan University, and the director in the National Supercomputing Center in Changsha. His major research areas include parallel computing, high performance computing, and grid and cloud computing. He has published more than 160 research papers in international conferences and journals such as IEEE TOC, TPDS, JPDC, ICPP, ICDCS, etc. He is an outstanding member of CCF. He is a senior member of the IEEE and serves on the editorial board of the IEEE Transactions on Computers.

Chubo Liu received the B.S. degree and Ph.D. degree in computer science and technology from Hunan University, China, in 2011 and 2016, respectively. He is currently an associate professor of computer science and technology at Hunan University. His research interests are mainly in game theory, approximation and randomized algorithms, cloud and edge computing. He has published over 20 papers in journals and conferences such as the IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Cloud Computing, IEEE Transactions on Mobile Computing, IEEE Transactions on Industrial Informatics, IEEE Internet of Things Journal, ACM Transactions on Modeling and Performance Evaluation of Computing Systems, Theoretical Computer Science, ICPADS, HPCC, and NPC. He won the Best Paper Award in IFIP NPC 2019 and the IEEE TCSC Early Career Researcher (ECR) Award in 2019. He is a member of IEEE and CCF.

Zhuo Tang received the Ph.D. in computer science from Huazhong University of Science and Technology, China, in 2008. He is currently a professor of the

College of Computer Science and Electronic Engineering at Hunan University, and is the associate chair of the department of computing science. His majors are distributed computing system, cloud computing, and parallel processing for big data, including distributed machine learning, security model, parallel algorithms, and resources scheduling and management in these areas. He has published almost 50 journal articles and book chapters. He is a member of ACM and CCF.

Keqin Li is a SUNY Distinguished Professor of computer science in the State University of New York. He is also a Distinguished Professor of Chinese National Recruitment Program of Global Experts (1000 Plan) at Hunan University, China. He was an Intellectual Ventures endowed visiting chair professor at the National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China, during 2011-2014. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published over 630 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently serving or has served on the editorial boards of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Transactions on Services Computing, and IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.