Budget-Constrained Service Allocation Optimization for Mobile Edge Computing

Yan Ding, *Student Member, IEEE*, Kenli Li, *Senior Member, IEEE*, Chubo Liu, *Member, IEEE*, Zhuo Tang, and Keqin Li, *Fellow, IEEE*

Abstract—The service resource allocation strategy optimization problem has always been a hot issue in mobile edge computing (MEC). In this article, we formulate the problem as a long-term quality of service (QoS) improvement problem while satisfying the budget of MEC service provider (MSP). Since it is very unrealistic to accurately obtain the request information of user equipments (UEs) over a long time, we first transform the original problem into a series of real-time linear programing sub-problems by using Lyapunov optimization method, and propose a centralized algorithm to determine the resource allocation strategies. However, since the sub-problems are still NP-hard problems, it is a huge challenge to determine the strategies for all UEs with the centralized algorithm in a large scale MEC environment. Thus, we then formulate the sub-problem as an N players non-cooperative game, prove that there exists a Nash equilibrium, and develop two iterative algorithms to find the Nash equilibrium while determining the strategies. Experimental results show that the algorithms can take into account QoS and budget of MSP at the same time, and perform better compared to five other common schemes.

Index Terms—Budget-constrained service allocation, lyapunov optimization method, mobile edge computing, non-cooperative game, nash equilibrium

1 Introduction

1.1 Motivation

There is a variety of intelligent applications that have changed the lifestyle of human society, such as autonomous driving, virtual reality, augmented reality, and facial recognition [1]. Driving these changes is the use of various types of data, such as sound, image, temperature, and humidity in the real-world. The data is collected by the millions of user equipments (UEs), such as smart phone, smart watch, and other Internet of Things (IoT) devices. Then the data is transmitted to the cloud for processing [2], thus making the applications intelligent.

However, as the scale of data continues to increase, it becomes impractical to transmit all user-side data to the cloud for analysis. Otherwise, this causes the performance of the already-congested backbone network to be further reduced, resulting in the worse quality of experience (QoE) and quality of service (QoS) [3], [4]. To address the reality dilemma, in 2014, the European Telecommunications Standards Institute (ETSI) proposed a computing architecture that named as mobile edge computing (MEC). In MEC, MEC service provider (MSP) deploys some servers with limited resources at the edge of network. This service model responds to UEs in a timely manner because UEs can leverage the high-speed wireless communication technology and the nearer MEC servers for efficient requesting [5], [6].

With the help of lightweight virtualization technology, such as virtual machine [7], VirtualBox [8], and Docker [9], an MSP can quickly deploy the specific service instances of UEs on its MEC servers. In this paper, the service instance of a UE can be a basic execution environment responding to the UE's request, or various other resources that the UE needs. Although the MSP can deploy enough MEC servers for running various service instances to improve QoS, it is impractical due to the budget constraint of MSP in real-world. Take video caching in MEC as an example. Video caching is stored in the MEC servers to provide low-latency video delivery for UEs. The servers can not store the cache of all UEs for a long time due to their limited storage space. Moreover, the MSP pays (such as power charge, equipment maintenance cost, and employee salary) to keep its servers running. In general, according to the status of UEs and the servers, the MSP updates the cache policy every certain period of time, such as several minutes or tens of minutes [10]. Hence, the MSP must consider its budget when optimizing its service resource allocation strategy [11].

How to trade-off the requirements of UEs and the practical needs of MSP is a non-trivial problem. On the one hand,

- *Yan Ding, Kenli Li, Chubo Liu, and Zhuo Tang are with the College of Information Science and Engineering, Hunan University, Changsha, Hunan 410082, China, and also with the National Supercomputing Center, Changsha, Hunan 410082, China. E-mail: {ding, lkl, liuchubo, ztang}@hnu.edu.cn.*
- *Keqin Li is with the College of Information Science and Engineering, Hunan University, Changsha, Hunan 410082, China, and with the National Supercomputing Center in Changsha, Hunan 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, New York, NY 12561 USA. E-mail: lik@newpaltz.edu.*

Manuscript received 19 Apr. 2021; revised 21 Sept. 2021; accepted 5 Dec. 2021. Date of publication 9 Dec. 2021; date of current version 6 Feb. 2023.
This work was supported in part by the National Outstanding Youth Science Program of National Natural Science Foundation of China under Grant 61625202, in part by the Program of National Natural Science Foundation of China under Grants 62072165, 61876061, and U19A2058, and in part by the Open Research Projects of Zhejiang Lab under Grant 2020KE0AB01.
(Corresponding authors: Kenli Li and Keqin Li.)
Digital Object Identifier no. 10.1109/TSC.2021.3133547

the UEs want the MSP to improve QoS and QoE, and the MSP wants to control its cost when responding to the UEs. On the other hand, since the resources of the MEC servers are finite, the service allocation strategy of a UE will significantly affect the performance of other UEs' strategies [12]. Moreover, the UEs and the application types of the UEs are all heterogeneous. It is difficult for MSP to make a common resource allocation strategy that satisfies different demands of UEs.

When involving mobility in MEC, the resource allocation strategy optimization problem becomes even harder [13], [14]. MSP should migrate the service instances of UEs to follow the movement of the UEs. If the service allocation strategies are determined based on the current location and request information of UEs, it will easily lead to frequent operation and increase the long-term cost of MSP [15]. However, it is a huge challenge to accurately obtain the information of UEs over a long time, especially for the UEs with random mobility and request. Thus, it is critical to weigh the respective requirements of UEs and MSP, and optimize service allocation strategies that are acceptable to both parties.

1.2 Our Contributions

Based on the above discussions, in this paper, we research the budget-constrained service allocation strategy optimization problem. The problem is to allocate the service resources (i.e., MEC server and computing resource of the server) of MSP for UEs with mobility to improve the long-term QoS with budget constraint consideration. The main contributions of this paper are presented as follows.

- We formulate the optimization problem as a long-term QoS improvement problem while satisfying the budget of MSP, transform the original problem into a series of real-time linear programing sub-problems by using Lyapunov optimization method, and develop a Lyapunov optimization method based centralized algorithm to determine the resource allocation strategies for all UEs, which addresses the randomness of the UEs.
- To improve the performance of service allocation for UEs in a large scale MEC environment, we then formulate the sub-problem as an N players non-cooperative game and prove there exists a Nash equilibrium of the players. Moreover, we develop two iterative algorithms to find the Nash equilibrium of UEs and determine the service allocation strategies for each UE in a distributed manner .
- We conduct extensive simulation experiments to demonstrate the convergence of algorithms, the effectiveness of algorithms, and the impact of various key factors, such as the weight parameter of latency and the budget of MSP, on the cost and QoS, respectively.

The remainder of the paper is outlined as follows. Related work is reviewed in Section 2. The system model and problem formulations are presented in Section 3. Section 4 transforms the original problem into a series of real-time linear programing sub-problems by using Lyapunov optimization method. Section 5 formulates the sub-problem as a non-cooperative game, and develops algorithms to find the Nash equilibrium in detail. Extensive simulation experiments using the real-world data are conducted in Section 6. Conclusions and our future work are presented in Section 7.

2 Related Work

Service resource allocation refers to that MSP configures the service instances of UEs on its servers based on the spatial-temporal information of UEs, thus enabling the MSP has the capability to respond to the different requests of the UEs [16], [17]. The service resource allocation strategy optimization problem has always been a hot issue in MEC and has been extensively studied. For example, Hung *et al.* [3] optimized live video streaming service by developing two auction frameworks to decide the backhaul capacity and caching space allocation of UEs. Kiani *et al.* [18] designed the service allocation policy in an auction-based profit maximization manner. Nguyen *et al.* [19] presented a decentralized and revised content-centric networking-based MEC service allocation strategy to save the MEC resource. Chen *et al.* [20] studied the collaborative service placement in dense network. Xiang *et al.* [21] developed an optimization method to improve the performance of service deployment. Deng *et al.* [22], [23], [24] investigated the resource allocation optimization problem from the perspectives of performance optimization, load balancing, and resource pricing, respectively. Although the above work studied the problem from various perspectives, they all ignored the impact of UE mobility in real-world.

Unlike the relatively stable service allocation strategy of the cloud, the mobility of UEs significantly affects the service allocation strategy in MEC [25], [26]. Some researches assumed that the mobility of UEs follows a Markov Decision Process (MDP) [27], [28]. To deploy the service instances of UEs in advance to improve QoS and QoE, some work investigated the problem based on the assumption that the UEs' future locations can be predicted [29], [30]. However, accurately predicting the UE mobility characteristics or future location is a huge challenge in real-world, especially the movement and request of UEs are highly random.

Therefore, it is important to explore efficient service allocation strategy without strong (accurate) assumption of UE mobility. Some work ignored the short-term mobility of UEs, and instead studied the resource allocation strategy over a long time [15], [31], [32], [33]. Although the above work developed the strategy to optimize the cost of MSP while improving QoS over a long time, the researches [31], [32], [33] still required the strong prior knowledge to make the strategy, that is, Wang *et al.* [31] assumed that the cost of all possible service allocation strategies are known in advance and Chen *et al.* [32], [33] determined the strategy when the demand patterns of UEs are known. Meanwhile, the MSP does not allocate its resources without considering its own budget, but this fact is only considered by [15], [32], [33]. Moreover, MEC is also known as a distributed cloud, so the request of a UE can be served by multiple MEC servers simultaneously. However, except for [19], [20], all of the above work does not consider the distributed service.

In our paper, the problem is formulated as a long-term quality of service (QoS) improvement problem while

TABLE 1
The Comparison Between the Related Work and Our Work

Categories	Schemes	Short-Term or Long-Term Optimization	Whether to Need the Priori Knowledge of Mobility or Demand?	Whether to Consider the Budget of MSP?	Whether to Consider the Distributed Service?
No Mobility Consideration	Hung *et al.* [3]	Short-Term	Yes	No	No
	Kiani *et al.* [18]	Short-Term	Yes	No	No
	Nguyen *et al.* [19]	Short-Term	Yes	No	Yes
	Chen *et al.* [20]	Short-Term	Yes	No	Yes
	Xiang *et al.* [21]	Short-Term	Yes	No	No
	Deng *et al.* [22], [23], [24]	Short-Term	Yes	No	No
With Mobility Consideration	Ouyang *et al.* [15]	Long-Term	No	Yes	No
	Urgaonkar *et al.* [27]	Short-Term	Yes	No	No
	Zhang *et al.* [28]	Short-Term	Yes	No	No
	Wu *et al.* [29]	Short-Term	Yes	No	No
	Zhou *et al.* [30]	Short-Term	Yes	No	No
	Wang *et al.* [31]	Long-Term	Yes	No	No
	Chen *et al.* [32]	Long-Term	Yes	No	No
	Ours	Long-Term	No	Yes	Yes

satisfying the budget of MEC service provider (MSP). Since it is very unrealistic to accurately obtain the request information of user equipments (UEs) over a long time, we first transform the original problem into a series of real-time linear programing sub-problems by using Lyapunov optimization method, and propose a centralized algorithm to determine the resource allocation strategies. However, since the sub-problems are still NP-hard problems, it is a huge challenge to decide the strategies with the centralized algorithm in a large scale MEC environment. Thus, we then formulate the sub-problem as an N players non-cooperative game, prove that there exists a Nash equilibrium, and develop two iterative algorithms to find the Nash equilibrium while determining the strategies. Consequently, the best strategies for UEs can be determined slot by slot. Table 1 shows the differences between the related work and our work. The algorithms developed in this paper do not need to know the mobility characteristics and demand patterns of UEs in advance, which determine the strategy more universal in the real-world.

3 System Model and Problem Formulation

As shown in Fig. 1, we consider an MEC environment in which a set of MEC servers $\mathcal{M} \triangleq \{1, 2, \ldots, M\}$ serve a set of UEs $\mathcal{N} \triangleq \{1, 2, \ldots, N\}$. All MEC servers belong to one MSP. We use $\text{UE}_i \in \mathcal{N}$ and $\text{MEC}_j \in \mathcal{M}$ to represent ith UE ($1 \leq i \leq N$) and jth MEC server ($1 \leq j \leq M$) respectively.

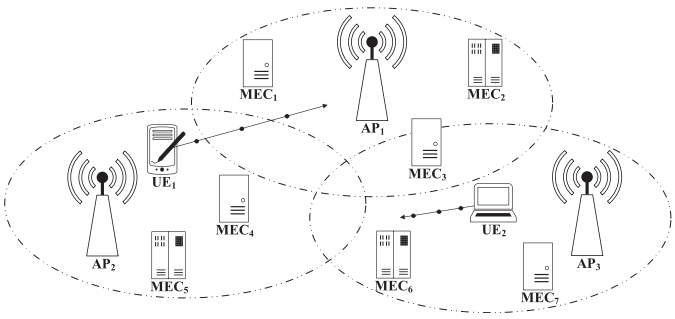

Fig. 1. A scenario example of MEC.

The communication between UE_i and MEC_j is implemented by the wireless access point (AP). Each AP can serve a set of UEs and MEC servers within a limited area. UEs request the MSP to execute their offloaded tasks for the purpose of minimizing the execution latency of the tasks. The MSP allocates its service resources to the UEs according to the location and request information of UEs, its own budget, and the status of each MEC server. In this paper, the service allocation strategy determines which MEC servers should respond to the UEs' requests and how many offloaded tasks of the UEs these servers should handle. The model that uses the number of tasks executedby MEC servers as the service allocation decision has been widely adopted by the existing work [15], [34]. Table 2 gives a list of the parameters and their definitions in this paper.

3.1 System Model

A time slot system is considered in this paper. UEs generate some tasks at each time slot $\tau \in \{0, 1, 2, \ldots, T-1\}$. The actual time interval of two consecutive slots can be either the same or different, such as at the time scale of second or minute [10], [31]. Meanwhile, when a UE's request is not completed, the UE remains stationary and does not send new requests [15], [31], [32]. The number of offloaded tasks generated by UE_i at τ is denoted by $\lambda_i(\tau)$. Moreover, a UE generates only one type of task at each time slot, i.e., the sizes of tasks generated by the UE at τ are equal. The MSP gathers all requests of UEs and determines the optimal service resource allocation strategies for each UE. To simplify the analysis, this paper assumes that the offloaded tasks can be further divided into some subtasks with any size and executed by multiple MEC servers simultaneously [35]. A typical application scenario is video analytics task, in which a video can be divided into a series of frames and the video analytics task (e.g., object detection) can be conducted on each frame separately [36]. Although, in reality, only some sizes may be accepted in the data partitioning, the solution in this paper could be served as a performance upper-bound of realistic resource allocation strategies.

$\lambda_{i,j}(\tau)$ is used to represent the number of offloaded tasks submitted by UE_i to MEC_j at τ. Thus $\lambda_i(\tau) = \sum_{j=1}^{M} \lambda_{i,j}(\tau)$. The

TABLE 2
Summary of Parameters

Parameters	Definition
	System Model
τ	a time slot, $0 \leq \tau \leq T-1$
UE_i	the ith UE, $1 \leq i \leq N$
MEC_j	the jth MEC server, $1 \leq j \leq M$
f_j	the computing resource of MEC_j, i.e., CPU frequency
$f_{i,j}(\tau)$	the computing resource of MEC_j allocated to UE_i at τ
C	the budget of MSP
$\lambda_i(\tau)$	the number of offloaded tasks generated by UE_i at τ
$\lambda_{i,j}(\tau)$	the number of offloaded tasks submitted by UE_i to MEC_j at τ
$\boldsymbol{\lambda_i(\tau)}$	$= \big(\lambda_{i,1}(\tau), \lambda_{i,2}(\tau), \lambda_{i,3}(\tau), \ldots, \lambda_{i,M}(\tau)\big)$, the resource allocation strategy of UE_i
$\boldsymbol{\Lambda_i}$	$= \big(\boldsymbol{\lambda_i}(0), \ldots, \boldsymbol{\lambda_i}(\tau), \ldots, \boldsymbol{\lambda_i}(T-1)\big)^T$, the strategies of UE_i over a period of time T
$\boldsymbol{\Lambda}$	$= \boldsymbol{\Lambda_1} \times \ldots \times \boldsymbol{\Lambda_N}$, the service resource allocation strategies for all UE_i over a period of time T
$A_i(\tau)$	$\triangleq \big(\lambda_i(\tau), a_i(\tau), \delta_i(\tau), l_i(\tau)\big)$, a request of UE_i at τ
$a_i(\tau)$	the data size of $A_i(\tau)$
$\delta_i(\tau)$	the processing density of $A_i(\tau)$
$l_i(\tau)$	$\triangleq \big(x_i(\tau), y_i(\tau)\big)$, the location of UE_i at τ
	QoS Model of UE
$r_{i,j}(\tau)$	the transmission rate from UE_i to MEC_j at τ
$t_{i,j,r}(\tau)$	the transmission latency from UE_i to MEC_j
$t_{i,j,m}(\tau)$	the service deployment latency of UE_i on MEC_j
$m_{i,j}(\tau)$	the delay for MEC_j migrating and activating the service instance of UE_i at τ
$t_{i,j,p}(\tau)$	the remote execution latency of UE_i's offloaded tasks executed by MEC_j
$Q_i(\tau)$	the latency that the MSP responds to UE_i's request
	Cost Model of MSP
$c_{i,j,p}(\tau)$	the execution energy required by MEC_j to execute the offloaded tasks of UE_i
$P_{i,j}$	$= bf_{i,j}^2(\tau)$, the rate at which the offloaded task is performed by MEC_j
b	a constant related to the chip
$c_{i,j,m}$	the deployment energy consumption required by MEC_j to respond to UE_i
$e_{i,j,m}(\tau)$	the energy consumption of MEC_j for migrating and activating the service instance of UE_i
$C_i(\tau)$	the energy consumption of MSP executing the tasks of UE_i at τ
$C(\tau)$	the energy consumption of MSP executing the tasks of all UEs at τ
	Lyapunov Optimization Method
$Z(\tau)$	the backlog of a discrete time queuing system of MSP defined over time slot τ
V	a weight parameter
$L\big(Z(\tau)\big)$	$= Z(\tau)^2/2$, the Lyapunov function
C_{max}	the maximum energy consumption of MSP for responding to UEs
B	$= \big(C_{max}^2 + \overline{C}^2 + 2C_{max} + 1\big)/2$
	Non-Cooperative Game Theory
$\mathbb{R}^M$	an euclidean space
K_i	the all possible strategies of UE_i
$\mathcal{K}$	$= K_1 \times K_2 \times \ldots \times K_N$, the all possible strategies of N players
$\boldsymbol{\Lambda_{-i}}$	$= (\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_{i-1}}, \boldsymbol{\lambda_{i+1}}, \ldots, \boldsymbol{\lambda_N})^T$, the strategies of $N-1$ players except for UE_i
$\xi_i(\boldsymbol{\lambda_i}, \boldsymbol{\Lambda_{-i}})$	the cost function of UE_i in a non-cooperative game
Ξ	$= \big(\xi_1(\boldsymbol{\lambda_1}, \boldsymbol{\Lambda_{-1}}), \ldots, \xi_N(\boldsymbol{\lambda_N}, \boldsymbol{\Lambda_{-N}})\big)$
$\mathcal{G}$	$= (\mathcal{K}, \Xi)$, a non-cooperative game
$\boldsymbol{\Lambda}^*$	$= (\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_N^*)^T$, the Nash equilibrium
$\mathbf{H}\big(\xi_i(\boldsymbol{\lambda_i}, \boldsymbol{\Lambda_{-i}})\big)$	the Hessian matrix of $\xi_i(\boldsymbol{\lambda_i}, \boldsymbol{\Lambda_{-i}})$
$\sigma_i, \mu_{i,j}, \omega_{i,j}$	the Lagrange multipliers
ζ	the number of Lagrange multiplier updates
$\gamma_i, \beta_i, \kappa_i$	the size of Lagrange multiplier update step
ψ	a constant that controls P3's solution precise
ϕ	the number of game rounds
Φ	the maximum number of game rounds
ϵ	the accuracy requirement of Algorithm 3

resource allocation strategy of UE_i can be represented by a vector $\boldsymbol{\lambda_i(\tau)} = \big(\lambda_{i,1}(\tau), \lambda_{i,2}(\tau), \lambda_{i,3}(\tau), \ldots, \lambda_{i,M}(\tau)\big)$. The strategies of UE_i over a time ($0 \leq \tau \leq T-1$) can be denoted by a matrix $\boldsymbol{\Lambda_i} = \big(\boldsymbol{\lambda_i}(0), \ldots, \boldsymbol{\lambda_i}(\tau), \ldots, \boldsymbol{\lambda_i}(T-1)\big)^T$. Fig. 2 shows an example of MSP responding to the requests of UEs.

The service resource allocation strategy of a UE is not only affected by the number of offloaded tasks, but also the data size, the task type, and the location of the UE. Thus, we use $A_i(\tau) \triangleq \big(\lambda_i(\tau), a_i(\tau), \delta_i(\tau), l_i(\tau)\big)$ to represent a request of UE_i at τ, where $a_i(\tau)$ (bit) is the data size of the request, $\delta_i(\tau)$

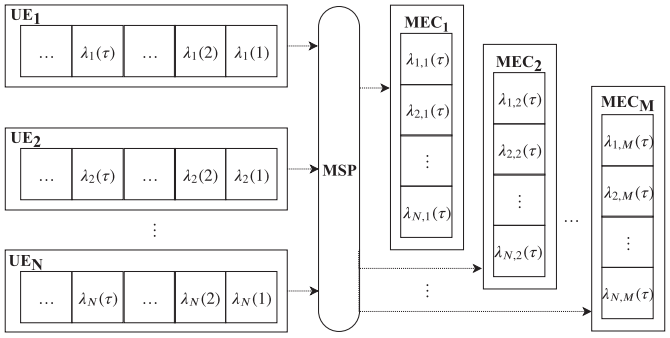

Fig. 2. An illustration of MSP responding to UEs' requests.

is the processing density (cycles/bit) of the request, and $l_i(\tau)$ is the location of UE_i at τ. The number of CPU cycles required to complete a request is equal to the product of processing density and data size, i.e., $a_i(\tau)\delta_i(\tau)$. We assume UE_i moves on a two-dimensional plane, thus the location of UE_i at τ can be denoted by a coordinate point $l_i(\tau) \triangleq \big(x_i(\tau), y_i(\tau)\big)$, where $x_i(\tau), y_i(\tau)$ are the abscissa and ordinate respectively. The location of MEC_j is $l_j \triangleq (x_j, y_j)$, where x_j, y_j are the abscissa and ordinate of MEC_j, respectively.

3.2 QoS Model

In this paper, QoS is represented by the latency that the MSP responds to UEs' requests. The response delay consists of not only the transmission latency, the service deployment latency, but also the remote execution latency. The transmission latency refers to the time consumption of a UE for uploading its tasks to MEC servers. The deployment latency is caused by migrating the service instance between the MEC servers and activating the service instance from the sleep state. The remote execution latency is the computational time for the MEC servers to execute the offloaded tasks. Next, we present QoS model in detail.

3.2.1 Transmission Latency

We use $r_{i,j}(\tau)$ to represent the transmission rate from UE_i to MEC_j at τ. In the paper, the communication channel condition between UE_i and MEC_j is assumed to be constant [32]. Since the distance between UE_i and MEC_j affects the transmission rate, we use $\hat{r}_{i,j}$ to represent the rate function with regard to the locations of UE_i and MEC_j, i.e., $r_{i,j}(\tau) \triangleq \hat{r}_{i,j}\big(l_i(\tau), l_j\big)$. The specific rate function will be introduced in Section 6.1. The transmission latency (second) from UE_i to MEC_j is

$$t_{i,j,r}(\tau) = \frac{a_i(\tau)}{r_{i,j}(\tau)} \cdot \frac{\lambda_{i,j}(\tau)}{\lambda_i(\tau)}. \tag{1}$$

3.2.2 Service Deployment Latency

When UE_i leaves the service area of an MEC server, QoS is reduced due to the longer transmission delay. Therefore, the service instance migration mechanism is introduced into the MEC. Consequently, there is an additional latency for migrating service instances between the MEC servers. Since idle instances will enter the sleep state, waking up the instances from sleep state again will cause additional delay. We use $m_{i,j}(\tau)$ to represent the delay for MEC_j migrating and activating the service instance of UE_i at τ. $m_{i,j}(\tau)$ can be measured through the long-term experience [36]. Thus, the service deployment latency (second) of UE_i on MEC_j can be formulated as

$$t_{i,j,m}(\tau) = \mathbb{I}\{\lambda_{i,j}(\tau-1) = 0\}\mathbb{I}\{\lambda_{i,j}(\tau) > 0\} m_{i,j}(\tau), \tag{2}$$

where $\mathbb{I}\{o\}$ is an indicator function with regard to a boolean variable o. If $o = true$, $\mathbb{I}\{o\} = 1$. Otherwise, $\mathbb{I}\{o\} = 0$.

3.2.3 Remote Execution Latency

We use f_j to represent the computing resource of MEC_j (i.e., CPU frequency, measured by cycles/s). In this paper, we assume that the MSP adopts the weighted computing resource allocation model. The computing resource weight of MEC_j allocated to UE_i at τ is the proportion of offloaded tasks submitted by the UE among all tasks executed by the server. That is, MEC_j does not consider the types of tasks submitted by UEs when allocating its resources, but only focuses on the number of tasks [37]. To support resource allocation and other system operation mechanisms, such as fault-recovery [38], MEC_j reserves part of its computing resource. The amount of resources occupied by these mechanisms is assumed to be equal to the amount of resources occupied by one offloaded task. Hence, the total number of tasks responded by MEC_j is $\sum_{i'=1}^{N} \lambda_{i',j}(\tau) + 1$, where $\sum_{i'=1}^{N} \lambda_{i',j}(\tau)$ is the total number of tasks offloaded to MEC_j by all $\mathrm{UE}_{i'} \in \mathcal{N}$ at τ. Therefore, the computing resource of MEC_j allocated to UE_i at τ is $f_{i,j}(\tau) = \lambda_{i,j} f_j(\tau) / \big(\sum_{i'=1}^{N} \lambda_{i',j}(\tau) + 1\big)$. Therefore, the remote execution latency (second) of UE_i's offloaded task executed by MEC_j is

$$\begin{aligned} t_{i,j,p}(\tau) &= a_i(\tau)\delta_i(\tau)\frac{\lambda_{i,j}(\tau)}{\lambda_i(\tau) f_{i,j}(\tau)} \\ &= a_i(\tau)\delta_i(\tau)\frac{\big(\sum_{i'=1}^{N} \lambda_{i',j}(\tau) + 1\big) f_j}{\lambda_i(\tau)}. \end{aligned} \tag{3}$$

Based on the above definitions, QoS improvement model of UE_i can be formulated as minimizing the following

$$Q_i(\tau) = \sum_{j=1}^{M} \big(t_{i,j,r}(\tau) + t_{i,j,m}(\tau) + t_{i,j,p}(\tau)\big). \tag{4}$$

3.3 Cost of MSP

The MSP receives the long-term payment from UEs if the demands of the UEs are satisfied. However, the MSP not only wants to improve QoS, but also wants to control its cost. Thus, the MSP will trade-off the cost and benefit of responding to the UEs' requests. The cost types of MSP are diverse, such as CPU occupancy rate, memory occupancy rate, energy consumption and time consumption for responding to UEs. In this paper, however, we consider the energy consumption (Joule, J) as the cost of MSP. The reason is that energy consumption has covered all the above cost types [39]. Moreover, we do not consider fixed energy cost, such as leakage and static energy consumptions, and focus on the maintenance energy consumption. The maintenance energy consumption of MSP consists of the offloading task execution energy consumption and the service instance deployment energy consumption. The execution energy (Joule) required by MEC_j to execute the offloaded tasks of

UE_i can be formulated as

$$c_{i,j,p}(\tau) = t_{i,j,p}(\tau)P_{i,j}, \tag{5}$$

where power $P_{i,j}(\tau) = bf_{i,j}^2(\tau)$ (Watt) is the rate at which the offloaded task is performed by MEC_j, and $b = 5 \times 10^{-17}$ is the constant related to the chip [37], [40], [41].

The deployment energy consumption (Joule) required by MEC_j to respond to UE_i is

$$c_{i,j,m} = \mathbb{I}\{\lambda_{i,j}(\tau - 1) = 0\}\mathbb{I}\{\lambda_{i,j}(\tau) > 0\}e_{i,j,m}(\tau), \tag{6}$$

where $e_{i,j,m}(\tau)$ (Joule, J) is the energy consumption of MEC_j for migrating and activating the service instance of UE_i. $e_{i,j,m}(\tau)$ is a constant related to the service instance type of UE_i, and can also be obtained by the MSP through the long-term experience [36].

According to the above formulations of the two energy consumption types, the cost (Joule) of MSP executing the tasks of UE_i at τ is

$$C_i(\tau) = \sum_{j=1}^{M}(c_{i,j,p}(\tau) + c_{i,j,m}(\tau)). \tag{7}$$

3.4 Problem Formulation

In the paper, the problem can be described as the MSP finds the optimal service resources allocation strategies for UEs to improve the long-term QoS within its energy consumption budget. Therefore, based on the above definitions, the problem can be formulated as

$$\begin{aligned}
&\mathrm{P1}: \min_{\mathbf{\Lambda}} \lim_{T\to\infty} \frac{1}{T}\sum_{\tau=0}^{T-1}\sum_{i=1}^{N} Q_i(\tau)\\
&s.t.\ C1: \lim_{T\to\infty} \frac{1}{T}\sum_{\tau=0}^{T-1}\sum_{i=1}^{N} C_i(\tau) \le \overline{C},\\
&\quad C2: \lambda_i(\tau) = \sum_{j=1}^{M}\lambda_{i,j}(\tau), i \in [1, N],\\
&\quad C3: 0 \le \lambda_{i,j}(\tau) \le \lambda_i, i \in [1, N], j \in [1, M],
\end{aligned} \tag{8}$$

where $\mathbf{\Lambda} = \mathbf{\Lambda_1} \times \ldots \times \mathbf{\Lambda_N}$ is the service resource allocation strategies for all $UE_i \in \mathcal{N}$ over a period of time (i.e., $0 \le \tau \le T-1$), and $\overline{C}$ is the cost constraint of MSP, i.e., budget.

4 Problem Transformation and Lyapunov Optimization Method Based Algorithm

4.1 Background Information

Consider a discrete time queuing system $Z(\tau)$. The arrivals and departures of $Z(\tau)$ is affected by a schedule action taken at each time slot τ. To describe the congestion state of system, Lyapunov optimization method defines the Lyapunov function as the sum of squares of backlog in the queue at τ, i.e., $L\big(Z(\tau)\big) = Z(\tau)^2/2$. The method also defines the Lyapunov drift $\Delta(\tau) = L\big(Z(\tau+1)\big) - L\big(Z(\tau)\big)$ to describe the change of Lyapunov function from τ to $\tau+1$. $Z(\tau)$ and $\Delta(\tau)$ help to ensure the long-term constraint is met. In a queuing system, if we want to optimize an objective function $\mathcal{P}(\tau)$ while maintaining the system stability, the schedule actions are decided at each slot τ to greedily minimize the drift-plus-penalty, i.e., $\Delta(\tau) + V\mathcal{P}(\tau)$, to consistently push $Z(\tau)$ to a lower congestion state. In the drift-plus-penalty expression, V is a non-negative control parameter that is chosen as desired.

Compared with other optimization approaches (such as convex programming and duality theory), one advantage of Lyapunov optimization method is that the method can unify the objective function and the long-term constraint into an equation (i.e., drift-plus-penalty), and enables the problem to be solved based on the current information, without requiring prior knowledge at other times. The another advantage of the method is that its performance bound can be demonstrated explicitly. For example, for each time slot, Theorem 2 shows the gap between the optimal average response delay obtained by solving P2 and the optimal solution of P1 is bounded by $O(1/V)$. Moreover, the average energy consumption of MSP is bounded by $O(V)$. For a detailed introduction of Lyapunov optimization method, the reader is referred to reference [42].

P1 is defined as the problem of minimizing the average response delay under the budget constraint over a long-term. The optimal solution of P1 can be calculated by continuously updating to follow the dynamic of UEs over a long time information (e.g., UEs' mobility characteristics and requests, and the computing resource of server). However, it is unrealistic to accurately obtain these knowledge over a long time [15]. Fortunately, we can regard the long-term budget constraint of MSP in P1 (i.e., $C1$) as a queue stability control problem, and transform the original long-term optimization problem into a series of linear programing sub-problems (i.e., P2) by using the Lyapunov optimization method. That is, we optimize the drift-plus-penalty (i.e., $\Delta(\tau) + V\mathcal{P}(\tau)$) of the system at every time slot. Thus, the strategy can be determined based only on the current information, and does not require prior knowledge at other times. Then, we can make the best strategies for UEs slot by slot. Moreover, it also allows us to make the strategies without considering the specific mobility characteristics and demand patterns of UEs. Next, we detail the problem transformation process by using the Lyapunov optimization method.

4.2 Problem Transformation

We define $Z(\tau)$ as the backlog of a discrete time queuing system, where $\tau \in \{0, 1, 2, \ldots, T-1\}$. The next time slot backlog of the queue $Z(\tau+1)$ is derived by the current energy consumption $C(\tau) = \sum_{i=1}^{N} C_i(\tau)$ and the budget $\overline{C}$ based on the following dynamic equation:

$$Z(\tau+1) = \max\{Z(\tau) - \overline{C} + C(\tau), 1\}. \tag{9}$$

The backlog of the queue can be represented by the additional energy needed to execute tasks. As shown in Equation (9), we assume that MSP reserves one J of energy in the queue for some emergency situations. In this paper, the initial queue backlog is one (i.e., $Z(0) = 1$). In fact, $Z(\tau)$ is another form of constraint $C1$. If $Z(\tau)$ can be proven to be mean rate stable, i.e., the expectation of average queue backlog $\lim_{T\to\infty} \mathbb{E}\{Z(\tau)/T\} = 0$, the average constraint $C1$ can be satisfied [42].

Based on the above definitions, we transform P1 into a series of real-time linear programing sub-problems (i.e., P2) by using Lyapunov optimization method, and have Theorem 1.

Theorem 1. *For each time slot, the sub-problem of P1 can be formulated as the following problem:*

$$\text{P2}: \min_{\mathbf{\Lambda}} V\sum_{i=1}^{N} Q_i(\tau) + Z(\tau)\left(\sum_{i=1}^{N} C_i(\tau)+1\right),$$
$$s.t.\ C2, C3, \tag{10}$$

where $V > 0$ is the trade-off parameter between the cost of MSP and QoS.

Proof. In this queuing system, the Lyapunov function is $L\big(Z(\tau)\big)= Z(\tau)^2/2$. Moreover, the Lyapunov drift is $L\big(Z(\tau+1)\big)-L\big(Z(\tau)\big)= \big(Z(\tau+1)^2 - Z(\tau)^2\big)/2 \le \big(C(\tau)^2+ \overline{C}^2+1 \quad +2C(\tau)\big)/2 - \big(C(\tau)+1\big)\overline{C} + Z(\tau)\big(C(\tau)-\overline{C}+1\big)$. Thus, we have

$$L\big(Z(\tau+1)\big)-L\big(Z(\tau)\big)\le Z(\tau)\big(C(\tau)-\overline{C}+1\big)+B, \tag{11}$$

where $B = \big(C_{max}^2+\overline{C}^2+2C_{max}+1\big)/2$, and C_{max} is the maximum energy consumption of MSP for responding to UEs, which is determined by the MSP. The conditional Lyapunov drift is

$$\Delta\big(Z(\tau)\big) \triangleq \mathbb{E}\big\{L\big(Z(\tau+1)\big)-L\big(Z(\tau)\big)|Z(\tau)\big\}. \tag{12}$$

According to Equations (11), (12), and the law of iteration expectation [42], we have

$$\begin{aligned}\mathbb{E}\big\{\Delta\big(Z(\tau)\big)\big\} &= \mathbb{E}\big\{L\big(Z(\tau+1)\big)-L\big(Z(\tau)\big)\big\}\\ &\le B+\mathbb{E}\big\{Z(\tau)\big\}\big(C(\tau)-\overline{C}+1\big).\end{aligned} \tag{13}$$

And then, according to the law of telescoping sums and $Z(0)=1$ [42], we get

$$\mathbb{E}\big\{L\big(Z(T)\big)\big\}-\frac{1}{2}\le BT+\sum_{\tau=0}^{T-1}\mathbb{E}\big\{Z(\tau)\big\}\big(C(\tau)-\overline{C}+1\big). \tag{14}$$

Thus, when $T\to\infty$, we rearrange the above inequality as

$$\lim_{T\to\infty}\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}\big\{Z(\tau)\big\}\le \lim_{T\to\infty}\frac{1}{T}\chi = 0, \tag{15}$$

where $\chi = (BT+1/2-\mathbb{E}\big\{L\big(Z(T)\big)\big\})/\sum_{\tau=0}^{T-1}\big(\overline{C}-C(\tau)+1\big)$. Thus $Z(\tau)$ is mean rate stable, that is $C1$ can be satisfied over a long time [42].

Accordingly, the Lyapunov drift-plus-penalty function is $\Delta(Z(\tau))+V\sum_{i=1}^{N}Q_i(\tau)\le Z(\tau)\mathbb{E}\big\{C(\tau)-\overline{C}+1|Z(\tau)\big\}+B+V\sum_{i=1}^{N}Q_i(\tau)$. Therefore, we get

$$\Delta\big(Z(\tau)\big)+VQ(\tau)\le Z(\tau)\big(C(\tau)+1\big)+B+VQ(\tau), \tag{16}$$

where $Q(\tau)=\sum_{i=1}^{N}Q_i(\tau)$. Therefore, if we optimize QoS over a long time while satisfying the energy consumption budget of MSP, we can minimize $\Delta\big(Z(\tau)\big)+VQ(\tau)$ slot by slot. Equivalently, we can minimize $VQ(\tau)+Z(\tau)\big(C(\tau)+1\big)$. So we have the theorem. □

4.3 Performance Analysis and the Algorithm

P2 can be described as that we want to stabilize the queue $Z(\tau)$ while making the average response delay of all UEs $Q(\tau)$ close to a minimal delay $q^*\ge 0$. We assume that the expectation of $Q(\tau)$ is lower bounded by a finite value $q_{min}\ge 0$, for all $\tau\in\{0,\dots,T-1\}$, we have

$$\mathbb{E}\{Q(\tau)\}\ge q_{min}. \tag{17}$$

The following theorem gives the bound of average response delay performance and energy consumption (i.e., the backlog of $Z(\tau)$), i.e., the gap between the solution obtained by solving P2 and the optimal solution of P1.

Theorem 2. *There are constants $B\ge 0, V>0, \rho\ge 0$, and $q^*\ge q_{min}$, for all $\tau\in\{0,\dots,T-1\}$. The average response delay of $Q(\tau)$ is bounded by*

$$\lim_{T\to\infty}\sup\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}\{Q(\tau)\}\le q^*+\frac{B}{V}. \tag{18}$$

The average energy consumption of MSP, i.e., the backlog of $Z(\tau)$ is bounded by

$$\lim_{T\to\infty}\sup\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}\{Z(\tau)\}\le\frac{B+V(q^*-q_{min})}{\rho+1}. \tag{19}$$

Proof. We assume that there is a non-negative ρ to make the constraint $C1$ true. Thus we have $\overline{C}-C\ge\rho$. Plugging this into Equation (14) yields

$$\Delta\big(Z(\tau)\big)\le B-(\rho+1)\mathbb{E}\{Z(\tau)\}. \tag{20}$$

Plugging the inequalities (17) and (20) into the inequality (16) can yield the following inequality:

$$\Delta\big(Z(\tau)\big)+VE\{Q(\tau)\}\le B+Vq^*-(\rho+1)\mathbb{E}\{Z(\tau)\}. \tag{21}$$

For the inequality (21), taking expectations of both sides and using the law of iteration expectations [42], we have an inequality $\mathbb{E}\{L\big(Z(\tau+1)\big)\}-\mathbb{E}\{L\big(Z(\tau)\big)\}+V\mathbb{E}\{Q(\tau)\}\le B+Vq^*-(\rho+1)\mathbb{E}\{Z(\tau)\}$. Then, summing the inequality over $\tau\in\{0,1,\dots,T-1\}$ for $T>1$ and using the law of telescoping sums [42], we can obtain an inequality $\mathbb{E}\{L\big(Z(T)\big)\}-\mathbb{E}\{L\big(Z(0)\big)\}+V\sum_{\tau=0}^{T-1}\mathbb{E}\{Q(\tau)\}\le(B+Vq^*)T-(\rho+1)\sum_{\tau=0}^{T-1}\mathbb{E}\{Z(\tau)\}$. Plugging $Z(0)=1$ into the inequality and rearranging the terms of the inequality, we can easily get the following two inequalities:

$$\begin{aligned}\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}\{Q(\tau)\}\le& q^*+\frac{B}{V}-\frac{(\rho+1)\sum_{\tau=0}^{T-1}\mathbb{E}\{Z(\tau)\}}{VT}\\ &+\frac{1/2-\mathbb{E}\{L\big(Z(T)\big)}{VT},\end{aligned} \tag{22}$$

$$\frac{1}{T}\sum_{\tau=0}^{T-1}\mathbb{E}\{Z(\tau)\} \leq \frac{B+Vq^*-V\mathbb{E}\{Q(\tau)\}}{\rho+1} + \frac{1}{2T(\rho+1)} - \frac{\mathbb{E}\{L(Z(T))\}}{T(\rho+1)} \leq \frac{B+V(q^*-q_{min})}{\rho+1} + \frac{1}{2T(\rho+1)} - \frac{\mathbb{E}\{L(Z(T))\}}{T(\rho+1)}. \quad (23)$$

Taking limits of the inequalities (22) and (23) as $T \to \infty$ can prove the theorem. □

Theorem 1 unifies the average response delay of UEs and the cost of MSP into an equation, and enables the problem to be solved based on the current information, without requiring prior knowledge at other times. Thus, we omit τ of parameters in the following content. Meanwhile, for each time slot, Theorem 2 shows the gap between the optimal average response delay obtained by solving P2 and the optimal solution of P1 is bounded by $O(1/V)$. Moreover, the average energy consumption of MSP is bounded by $O(V)$. Based on Theorems 1 and 2, it is easy to know that the solution of P2 is an approximate optimal solution of P1 in every time slot.

Algorithm 1. LY: Lyapunov Optimization Method Based Algorithm for Finding $\mathbf{\Lambda}^*$

Input: $A_i(\tau), m_{i,j}(\tau), e_{i,j,m}(\tau), f_j$, for $1 \leq i \leq N$ and $1 \leq j \leq M$, $\overline{C}$, $Z(\tau)$, and b.
Output: $\mathbf{\Lambda}^*$.
1: **for** $\tau \in \{0, 1, \ldots\}$ **do**
2: Obtain the optimal resource allocation strategies through $\mathbf{\Lambda^*(\tau)} = \arg\min_{\mathbf{\Lambda}} VQ(\tau) + Z(\tau)\big(C(\tau)+1\big)$;
3: Update $Z(\tau+1)$ by Equation (9);
4: **end for**
5: return $\mathbf{\Lambda}^*$.

Algorithm 1 shows that the optimal strategies of UEs at each time slot τ can be obtained when solving P2. The algorithm does not need to know the mobility characteristics and demand patterns of UEs in advance. However, since P2 is an NP-hard problem [15], it is a huge challenge to solve P2 with a centralized algorithm especially in a large scale network. Moreover, there is a fact that all UEs want to have as little latency as possible, which indicates that the MSP needs to optimize service allocation strategy for each UE.

Fortunately, game theory provides an efficient way to solve P2 while determining the strategies for UEs. Thus, we then formulate the sub-problem as an N players non-cooperative game, prove that there exists a Nash equilibrium, and develop two algorithms to find the Nash equilibrium while determining the strategies. The algorithms are iterative algorithms and determine the best strategy for each UE in a distributed manner, thereby reducing the time complexity of obtaining the solution of P2. In each iteration, each UE is greedy and wants MSP to adjust its service resource allocation strategy to minimize its service latency. The iteration will continue until a strategy set acceptable to all UEs is obtained, that is, the strategies of UEs can no longer continue to be updated to benefit the UEs. According to Definition 2 proposed in Section 5.1, we can know that although the Nash equilibrium solution is not the optimal strategy set of P2, the solution consists of the best strategies of all UEs [37].

5 Non-Cooperative Game Based Algorithms

5.1 The Preliminary of Game

In this section, we first give some definitions about the N players non-cooperative game. There are N players (i.e., N UEs) in a game and all players want to minimize their cost (i.e., improve their QoS). The ith player UE$_i$ makes a strategy $\boldsymbol{\lambda_i} = (\lambda_{i,1}, \lambda_{i,2}, \ldots, \lambda_{i,M}) \in K_i \subseteq \mathbb{R}^M$, where K_i (i.e., the all possible strategies of UE$_i$) is closed and convex, for all $1 \leq i \leq N$. $\mathcal{K} = K_1 \times K_2 \times \ldots \times K_N$ indicates the set of the all possible strategies of N players. According to the former definition, $\mathbf{\Lambda}$ is the strategy set of all players. We use $\mathbf{\Lambda_{-i}}$ to represent the strategies of $N-1$ players except for UE$_i$, i.e., $\mathbf{\Lambda_{-i}} = (\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_{i-1}}, \boldsymbol{\lambda_{i+1}}, \ldots, \boldsymbol{\lambda_N})^T$. Each player has a cost function $\xi_i(\boldsymbol{\lambda_i}, \mathbf{\Lambda_{-i}}) \in \mathbb{R}$. Meanwhile, the cost function ξ_i is continuously differentiable in $\mathbf{\Lambda}$. Next, we can give the definition of non-cooperative game.

Definition 1. *There is a game with N UEs defined by $\mathcal{G} = (\mathcal{K}, \Xi)$, where $\Xi = \big(\xi_1(\boldsymbol{\lambda_1}, \mathbf{\Lambda_{-1}}), \ldots, \xi_N(\boldsymbol{\lambda_N}, \mathbf{\Lambda_{-N}})\big)$. Every player wants to make a strategy $\boldsymbol{\lambda_i} \in K_i$ to minimize its cost function $\xi_i(\boldsymbol{\lambda_i}, \mathbf{\Lambda_{-i}})$. Based on Theorem 1, when the strategies of all UEs are given except for UE$_i$, the cost function of UE$_i$ can be formulated as*

$$\xi_i(\boldsymbol{\lambda_i}, \mathbf{\Lambda_{-i}}) = V\left(Q_i + \sum_{k\neq i}^{N} Q_k\right) + Z\left(C_i + \sum_{k\neq i}^{N} C_k + 1\right). \quad (24)$$

The cost minimization problem of UE$_i$ can be formulated as

$$\mathrm{P3}: \min_{\boldsymbol{\lambda_i}} \xi_i(\boldsymbol{\lambda_i}, \mathbf{\Lambda_{-i}}) \quad s.t.\ C2, C3. \quad (25)$$

Then, we name the game as an N players non-cooperative game.

Definition 2. *$\mathbf{\Lambda}^* = (\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_N^*)^T$ is the strategy set of N UEs. If $\xi_i(\boldsymbol{\lambda}_i^*, \mathbf{\Lambda}_{-i}^*) \leq \xi_i(\boldsymbol{\lambda_i}, \mathbf{\Lambda}_{-i}^*)$, for all $1 \leq i \leq N$, $\mathbf{\Lambda}^*$ is the Nash equilibrium of the N players non-cooperative game $\mathcal{G} = (\mathcal{K}, \Xi)$.*

How do we know whether a game has the Nash equilibrium? Thanks to the following theorem, we can get the answer of the question [37].

Theorem 3. *If $\xi_i(\boldsymbol{\lambda_i}, \mathbf{\Lambda_{-i}})$ is a convex function with regard to $\boldsymbol{\lambda_i}$ when $\mathbf{\Lambda_{-i}}$ is given, for all $1 \leq i \leq N$, there is a Nash equilibrium of the N players non-cooperative game $\mathcal{G} = (\mathcal{K}, \Xi)$.*

Hence, we can first prove that $\xi_i(\boldsymbol{\lambda_i}, \mathbf{\Lambda_{-i}})$ has the property of Theorem 3. According to Theorem 4, we can use the non-cooperative game theory to solve P2.

Theorem 4. *The N players non-cooperative game $\mathcal{G} = (\mathcal{K}, \Xi)$ has a Nash equilibrium.*

Proof. According to Equations (1), (2), (3), and (4) and (24), we have

$$\frac{\partial \xi_i}{\partial \lambda_{i,j}} = bZ\frac{a_i\delta_i f_j(\lambda_{i,j}^2 + 2\lambda_{i,j}\lambda_{-i,j} + 2\lambda_{i,j})}{\lambda_i(\lambda_{i,j}+\lambda_{-i,j}+1)^2} + V\frac{a_i}{r_{i,j}\lambda_i} + V\frac{a_i\delta_i}{\lambda_i f_j}, \tag{26}$$

where $\lambda_{-i,j} = \sum_{i=1}^{N}\lambda_{i,j} - \lambda_{i,j}$, and

$$\frac{\partial^2 \xi_i}{\partial \lambda_{i,j}^2} = 2bZ\frac{a_i\delta_i f_j(\lambda_{-i,j}+1)^2}{\lambda_i(\lambda_{i,j}+\lambda_{-i,j}+1)^3}. \tag{27}$$

Furthermore, it is easy to know that

$$\frac{\partial^2 \xi_i}{\partial \lambda_{i,j}^2} \geq 0, \tag{28}$$

and

$$\frac{\partial^2 \xi_i}{\partial \lambda_{i,j}\partial \lambda_{i,j'}} = 0, \tag{29}$$

for all $1 \leq j, j' \leq M$, where $j' \neq j$.

Thus, the Hessian matrix

$$\mathbf{H}\big(\xi_i(\boldsymbol{\lambda_i}, \boldsymbol{\Lambda_{-i}})\big) = \left[\frac{\partial^2 \xi_i}{\partial \lambda_{i,j}\partial \lambda_{i,j'}}\right]_{M\times M} \tag{30}$$

is positive semidefinite on the interior set of $\mathcal{K}$. Thus, $\xi_i(\boldsymbol{\lambda_i}, \boldsymbol{\Lambda_{-i}})$ is a convex function [43]. Based on Theorem 3, we have the conclusion. □

5.2 The Best Resource Allocation Algorithm for UE$_i$

For non-cooperative game, the MSP allocates resource for one UE under the condition that the strategies of other UEs are given. Meanwhile, since $\xi_i(\boldsymbol{\lambda_i}, \boldsymbol{\Lambda_{-i}})$ is a convex function, the best resource allocation for a UE can be obtained through the convex optimization method, i.e., the Lagrange multiplier method. The Lagrange function of P3 is

$$L_i(\boldsymbol{\lambda_i}, \sigma_i, \boldsymbol{\mu_i}, \boldsymbol{\omega_i}) = \xi_i(\boldsymbol{\lambda_i}, \boldsymbol{\Lambda_{-i}}) + \sigma_i g(\boldsymbol{\lambda_i}) + \boldsymbol{\mu_i}\boldsymbol{h}(\boldsymbol{\lambda_i}) + \boldsymbol{\omega_i}\boldsymbol{s}(\boldsymbol{\lambda_i}), \tag{31}$$

where $g(\boldsymbol{\lambda_i}) = \lambda_{i,1} + \lambda_{i,2} + , \ldots, + \lambda_{i,M} - \lambda_i$, $\boldsymbol{h}(\boldsymbol{\lambda_i}) = (\lambda_{i,1} - \lambda_i, \ldots, \lambda_{i,M} - \lambda_i)$, $\boldsymbol{s}(\boldsymbol{\lambda_i}) = (-\lambda_{i,1}, \ldots, -\lambda_{i,M})$, $\boldsymbol{\mu_i} = (\mu_{i,1}, \ldots, \mu_{i,M})^T$, $\boldsymbol{\omega_i} = (\omega_{i,1}, \ldots, \omega_{i,M})^T$, and $\sigma_i, \mu_{i,j} \geq 0, \omega_{i,j} \geq 0$ for $1 \leq i \leq N, 1 \leq j \leq M$ are the Lagrange multipliers.

Therefore, we can use the Karush-Kuhn-Tucker (KKT) conditions to determine the optimal resource allocation strategy of UE$_i$, i.e., the optimal solution of P3. The KKT conditions of P3 are

$$\partial L_i(\boldsymbol{\lambda_i^*}, \sigma_i^*, \boldsymbol{\mu_i^*}, \boldsymbol{\omega_i^*})/\partial \lambda_{i,j}^* = 0, \tag{32}$$

$$\sigma_i^* g(\boldsymbol{\lambda_i^*}) = 0, \tag{33}$$

$$\boldsymbol{\mu_i^*}\boldsymbol{h}(\boldsymbol{\lambda_i^*}) = 0, \tag{34}$$

$$\boldsymbol{\omega_i^*}\boldsymbol{s}(\boldsymbol{\lambda_i^*}) = 0. \tag{35}$$

The optimal resource allocation strategy of MEC$_j$ for UE$_i$ can be obtained from

$$\frac{\partial L_i}{\partial \lambda_{i,j}} = \frac{\partial \xi_i}{\partial \lambda_{i,j}} + \sigma_i + \mu_{i,j} - \omega_{i,j} = 0, \tag{36}$$

where $\lambda_{i,j}^*$ can be calculated by using the following theorem.

Theorem 5. *$\lambda_{i,j}^* = \big(-\nu_{i,j} + \sqrt{\nu_{i,j}^2 - 4\theta_{i,j}\pi_{i,j}}\big)/2\theta_{i,j}$, where $\nu_{i,j} = 2bZa_i\delta_i r_{i,j}\lambda_{-i,j}f_j^2 + 2Va_i(\lambda_{-i,j}+1)(f_j + \delta_i r_{i,j}) + 2\lambda_i r_{i,j} f_j(\lambda_{-i,j}+1)(\sigma_i + \mu_i - \omega_{i,j})$, $\theta_{i,j} = bZa_i\delta_i r_{i,j}f_j^2 + Va_i(f_j + \delta_i r_{i,j}) + \lambda_i f_j r_{i,j}(\sigma_i + \mu_i - \omega_{i,j})$, $\pi_{i,j} = bZa_i\delta_i r_{i,j}f_j^2 + Va_i(f_j + \delta_i r_{i,j})(\lambda_{-i,j}+1)^2 + \lambda_i r_{i,j}f_j(\sigma_i + \mu_i - \omega_{i,j})(\lambda_{-i,j}+1)^2$ are the coefficient of first-order term, the coefficient of second-order term, and constant term of Equation (36), respectively.*

Proof. Plugging Equation (26) into Equation (36) and rearranging the terms, we have

$$\begin{aligned} &bZa_i\delta_i r_{i,j}f_j^2(\lambda_{i,j}^2 + 2\lambda_{i,j}\lambda_{-i,j} + 1) \\ &+ Va_i(f_j + \delta_i r_{i,j})(\lambda_{i,j}+\lambda_{-i,j}+1)^2 \\ &+ \lambda_i r_{i,j}f_j(\sigma_i + \mu_i - \omega_{i,j})(\lambda_{i,j}+\lambda_{-i,j}+1)^2 \\ &= 0. \end{aligned} \tag{37}$$

Equation (37) is a quadratic equation w.r.t. $\lambda_{i,j}$ when other parameters are given. Rearranging the terms of equation, we easily obtain the coefficient of first-order term $\nu_{i,j}$, the coefficient of second-order term $\theta_{i,j}$, and constant term $\pi_{i,j}$. Based on Vieta theorem [44] and $\lambda_{i,j} \geq 0$, we have $\lambda_{i,j}^* = \big(-\nu_{i,j} + \sqrt{\nu_{i,j}^2 - 4\theta_{i,j}\pi_{i,j}}\big)/2\theta_{i,j}$. □

As shown in Theorem 5, $\lambda_{i,j}^*$ is related to the Lagrange multipliers. Next, based on the Slater's constraint [43], P3 can be solved by using the dual problem of P3. The dual problem of P3 can be formulated as

$$\begin{aligned} &\mathrm{P4}: \max_{\sigma_i, \boldsymbol{\mu_i}, \boldsymbol{\omega_i}} \min_{\boldsymbol{\lambda_i}} L_i(\boldsymbol{\lambda_i}, \sigma_i, \boldsymbol{\mu_i}, \boldsymbol{\omega_i}) \\ &s.t.\ C4: \boldsymbol{\mu_i}, \boldsymbol{\omega_i} \succeq 0. \end{aligned}$$

Then the sub-gradient method can be used to update the Lagrange multipliers while finding $\lambda_{i,j}^*$. The update functions of the Lagrange multipliers are

$$\sigma_i(\zeta+1) = \sigma_i(\zeta) + \gamma_i\frac{\partial L_i}{\partial \sigma_i}, \tag{39}$$

$$\mu_{i,j}(\zeta+1) = \max\left\{\mu_{i,j}(\zeta) + \beta_i\frac{\partial L_i}{\partial \mu_{i,j}}, 0\right\}, \tag{40}$$

$$\omega_{i,j}(\zeta+1) = \max\left\{\omega_{i,j}(\zeta) + \kappa_i\frac{\partial L_i}{\partial \omega_{i,j}}, 0\right\}, \tag{41}$$

where $\gamma_i, \beta_i, \kappa_i \in (0,1)$ are the size of update step. Hence, we can update the Lagrange multipliers iteratively until a feasible solution of P3 is obtained.

Algorithm 2. FL: Search Algorithm for Finding $\boldsymbol{\lambda_i}$

Input: $\sigma_i, A_i, m_{i,j}, e_{i,j,m}, f_j$, $\overline{C}$, and $\mu_{i,j}, \omega_{i,j}$, for all $1 \leq j \leq M$, $\overline{C}$, $Z(\tau)$, and b.
Output: $\boldsymbol{\lambda_i^*}$.

Initialize $\boldsymbol{\lambda_i} \leftarrow (0, \ldots, 0)_M$;
while $|\sum_{i=1}^{N} \lambda_{i,j} - \lambda_i| > \psi$ **do**
  Calculate $g(\boldsymbol{\lambda_i})$;
  Update Lagrange multipler, $\sigma_i \leftarrow \sigma_i + \gamma_i g(\boldsymbol{\lambda_i})$;
  for $j \leftarrow 1$ to M **do**
    while True **do**
      Calculate $h(\lambda_{i,j}), s(\lambda_{i,j})$;
      Update Lagrange multipler, $\mu_{i,j} \leftarrow \max\{\mu_{i,j} + \beta_i h(\lambda_{i,j}), 0\}$;
      Update Lagrange multipler, $\omega_{i,j} \leftarrow \max\{\omega_{i,j} + \kappa_i s(\lambda_{i,j}), 0\}$;
      Calculate $\lambda_{i,j}$ based on Theorem 5;
      if $0 \leq \lambda_{i,j} \leq \lambda_i$ **then**
        Update $\lambda_{i,j}$;
        break;
      end if
    end while
  end for
  Update strategy of UE$_i$, $\boldsymbol{\lambda_i} \leftarrow (\lambda_{i,1}, \ldots, \lambda_{i,M})$;
end while
Obtain the optimal strategy of UE$_i$, $\boldsymbol{\lambda_i^*} \leftarrow \boldsymbol{\lambda_i}$;
return $\boldsymbol{\lambda_i^*}$.

Algorithm 2 shows the FL algorithm of obtaining $\boldsymbol{\lambda_i^*}$. We initialize a set of the Lagrange multipliers and get a solution of P3, and then check whether the solution satisfies the KKT conditions of P3. If the solution does not satisfy the KKT conditions, the Lagrange multipliers are updated according to the Equations (39), (40), and (41) until a feasible solution is obtained. Then we can easily obtain $\lambda_{i,j}$ according to Theorem 5, for all $1 \leq j \leq M$. The time complexity of the sub-gradient method is $O(1/\psi^2)$, where ψ controls P3's solution precise [45]. Since Algorithm 2 needs to perform sub-gradient method M times to find $\boldsymbol{\lambda_i}$, we know that the time complexity of the algorithm is $O(M/\psi^2)$.

Algorithm 3. NE: Nash Equilibrium Calculating Algorithm

Input: $\sigma_i, A_i, m_{i,j}, e_{i,j,m}, f_j$, $\overline{C}$, Φ, $\mu_{i,j}, \omega_{i,j}$, and b, for all $1 \leq i \leq N, 1 \leq j \leq M$.
Output: $\boldsymbol{\Lambda^*}$.

Initialize $\boldsymbol{\Lambda} \leftarrow (\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_N})$;
while $\|\boldsymbol{\Lambda'} - \boldsymbol{\Lambda}\|_2 > \epsilon$ and $\phi \leq \Phi$ **do**
  for UE$_i \in \mathcal{N}$ **do**
    Obtain $\boldsymbol{\lambda_i}$ through Algorithm 2;
    Update strategy of UE$_i$, $\boldsymbol{\lambda_i'} \leftarrow \boldsymbol{\lambda_i}$;
  end for
  Update strategy of UEs, $\boldsymbol{\Lambda'} \leftarrow (\boldsymbol{\lambda_1'}, \ldots, \boldsymbol{\lambda_N'})$;
  Increase the number of game rounds, $\phi \leftarrow \phi + 1$;
end while
Obtain the Nash equilibrium of game, $\boldsymbol{\Lambda^*} \leftarrow \boldsymbol{\Lambda'}$;
return $\boldsymbol{\Lambda^*}$.

5.3 The Algorithm for Nash Equilibrium of UEs

In this section, the resource allocation algorithm is developed to find the Nash equilibrium of N UEs. As shown in Algorithm 3, for N UEs, we develop an iterative Algorithm NE and determine the strategy for each UE. In each round, MSP first allocates its resource to each UE by using Algorithm 2. Then, MSP adjusts the resource allocation strategies after the game between the UEs. The game between UEs refers to that, given the strategies of other $N-1$ UEs except for UE$_i$, the MSP attempts to adjust the strategy of UE$_i$ to reduce the service latency of the UE. The game terminates when the consequences of two successive rounds close enough or the number of game rounds exceeds the upper limit game round. Therefore, we regard the final strategy $\boldsymbol{\Lambda^*} = (\boldsymbol{\lambda_1^*}, \ldots, \boldsymbol{\lambda_N^*})$ as a Nash equilibrium of the game. In the algorithm, if $\|\boldsymbol{\Lambda'} - \boldsymbol{\Lambda}\|_2 = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M} |\lambda'_{i,j} - \lambda_{i,j}|^2} \leq \epsilon$, we can get the Nash equilibrium of N UEs, which means that all UEs will accept the fact that their strategies can no longer be adjusted to benefit themselves. We introduce Φ as the maximum number of game rounds to control the complexity of Algorithm 3. Meanwhile, Φ can be determined by the termination condition ϵ of the algorithm. Since the number of times that Algorithm 3 calls Algorithm 2 is ΦN at most, it is easy to know that the time complexity of the algorithm is $O(\Phi NM/\psi^2)$.

6 Experiments and Analysis

6.1 Experimental Setting

Due to the lack of real data of the system model formulated in this paper, we simulate an environment in which the parameters used in the experiments are real-world values obtained from some other different work [36]. Meanwhile, we additionally introduce a random quantity in the parameters to reflect the heterogeneous characteristics of MEC. We simulate 1000 time slots for our scenario, and generate $N = 10$ UEs and $M = 50$ MEC servers. The specific parameters settings of the UEs and the MEC servers are as follows. $f_j = r_a \times 10^{10}$ cycles/s [36], $\lambda_i(\tau) = 3 + 0.1r_a$, $\delta_i(\tau) = 1000 + 10r_a$ cycles/bit [46], $a_i(\tau) = 10475760 + 100r_a$ bits, $f_i(\tau) = 2.5 \times 10^8 + 10000r_a$ cycles/s [36], $m_{i,j}(\tau) = 3.1 + 0.1r_a$ s [47], and $e_{i,j,m}(\tau) = 1 + 0.1r_a$ J [15], where r_a is a random integer variable taken from [1,5]. UEs are assumed to be moving on a 100×100 square meters two-dimensional plane, i.e., $-100 \leq x_i(\tau), y_i(\tau) \leq 100$. The initial locations of UEs are (0,0). UE$_i$ moves along the main diagonal of the plane and its speed is a random integer variable $\eta_i \in [3, 5]$ m/s. All MEC servers are placed on the diagonal of two-dimensional plane, and their intervals are $2\sqrt{2}$ m.

The maximum transmission rate of UE$_i$ at τ is denoted by $r_{i,max}(\tau) = 100 + 0.1r_a$ Mbps [47]. Meanwhile, the maximum service distance of an MEC server is 10 m. In other words, when the distance $\big(d_{i,j}(\tau)\big)$ between UE$_i$ and MEC$_j$ is greater than 10 m, the MSP will not allocate this server to the UE. Moreover, there is a decay factor $\alpha_{i,j}(\tau) = \min\{10/d_{i,j}(\tau), 1\}$ of transmission rate related to $d_{i,j}(\tau)$. Thus, the actual transmission rate from UE$_i$ to MEC$_j$ is $r_{i,j}(\tau) = \alpha_{i,j}(\tau) r_{i,max}(\tau)$. To minimize the energy consumption, MSP will also not allocate MEC$_j$ to UE$_i$ when $\alpha_{i,j}(\tau) < \alpha$, where $\alpha \in [0, 1]$ is a factor related to a server's service area set by the MSP. By setting α, the MSP can improve QoS or reduce its cost. In addition, we set $\alpha = 0.5$, $\overline{C} = 20000$ J, $V = 2000$, $\psi = 0.1$, $\epsilon = 1$, and $\Phi = 200$.

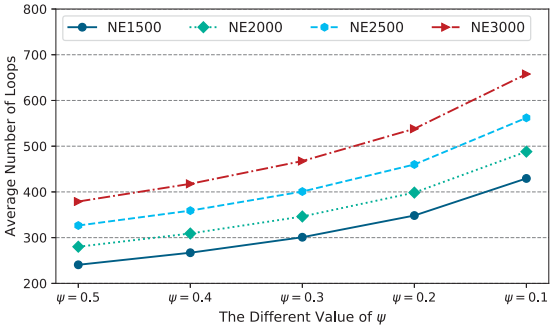

Fig. 3. The impact of ψ on the average number of loops required to find λ_i^*.

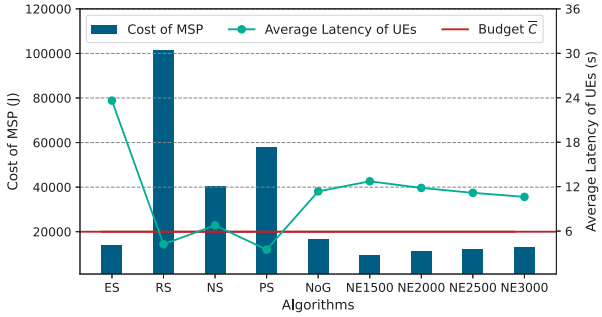

Fig. 5. The comparison between the algorithms.

6.2 Experiments and Results

6.2.1 The Convergence of Algorithms

Fig. 3 shows that the number of loops required by Algorithm FL to find λ_i^*. In the figure, NE1500 represents Algorithm NE with $V = 1500$, and other similar symbols (i.e., NE2000, NE2500, and NE3000) represent similar meaning. The figure shows that the number of loops increases roughly linearly with ψ. Since ψ controls the precise of P3's solution, as the precise improves, the number of search loops increases. Meanwhile, as shown in Fig. 4, since ϵ controls the termination precise of Algorithm NE, thus increasing the number of game rounds as ϵ decreases. Moreover, we can see that the number of iterations of Algorithms FL and NE both increase as V becomes larger. The reason is the algorithms with bigger V pay more attention to improving QoS, thus increasing the number of loops to find λ_i^*, and the number of game rounds to find the Nash equilibrium.

6.2.2 The Effectiveness of Algorithm NE

The following five algorithms are regarded as the baselines to evaluate the effectiveness of Algorithm NE: (1) ES: Each MEC server receives an equal amount of UE$_i$'s offloaded tasks, i.e., the tasks of the UE are evenly distributed to the available MEC servers; (2) RS: Each UE is served by a random MEC server; (3) NS: Each UE is served by an MEC server closest to itself; (4) PS: UEs are served by the most powerful server of MSP; (5) NoG: The service allocation strategy of a UE is made by using Algorithm FL, which means that there is no game between UEs.

In fact, although distributed computing is not necessarily faster than centralized computing, it can reduce the cost of servers [48]. As shown in Fig. 5, on the premise that MSP pays a high cost, Algorithms RS, NS, and PS greatly reduce the average latency of UEs. However, the energy consumption of RS, NS, and PS has exceeded the budget constraint of MSP (i.e., $\overline{C} = 20000$). Moreover, V indicates the importance of how much UEs emphasize QoS. As UEs value QoS (i.e., latency) more and more, the MSP will allocate more resources to the UEs. Comparing NE1500, NE2000, NE2500 and NE3000, it can be seen that as V increases, the average latency of UEs decreases while increasing the energy consumption of MSP.

Fig. 6 shows a game between UEs. As shown in the figure, when $V = 2000$ and $V = 2500$, Algorithm 3 focuses more on reducing the cost of MSP. It can be seen that after each round of the game, the average latency of UEs gradually increases, and the cost of MSP gradually decreases. When $V = 3500$ and $V = 5000$, Algorithm 3 focuses more on optimizing the average latency of UEs. In the second round of the game, reducing the average latency of UEs increases the cost of MSP (even exceeding the budget constraint of MSP). However, the algorithm quickly corrects this trend, and improves QoS while controlling the cost of MSP. It is not difficult to see that the optimization of QoS has increased the cost of MSP.

According to the above discussions, Algorithm NE can improve the long-term QoS while ensuring that the MSP budget constraint is met. In addition, MSP can determine the priority of different requests based on V to provide differentiated services. Compared with other methods, the algorithm proposed in this paper is more flexible and has better performance.

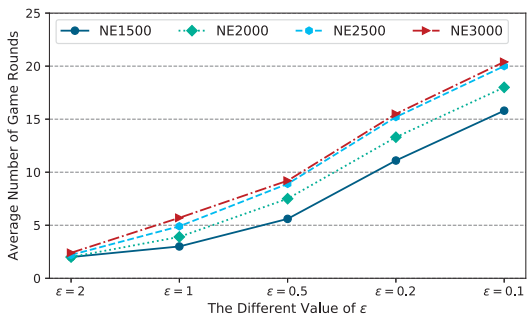

Fig. 4. The impact of ϵ on the average number of game rounds.

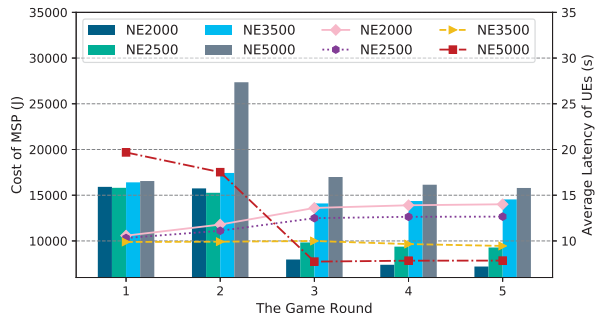

Fig. 6. The game between UEs.

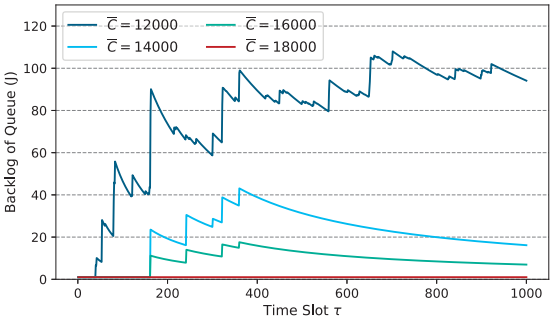

Fig. 7. The backlog of $Z(\tau)$ with different $\overline{C}$.

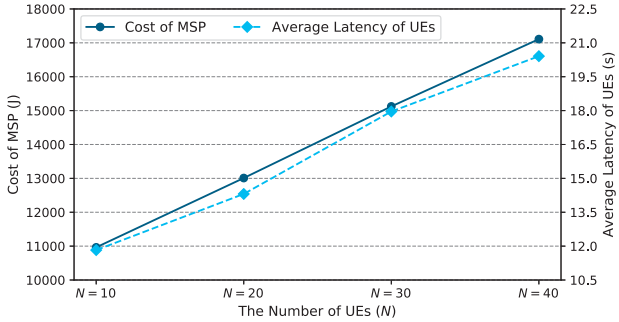

Fig. 9. The impact of UE amount on the average latency of UEs and the cost of MSP.

6.2.3 The Stability of Queue $Z(\tau)$

As shown in Fig. 7, when the MSP's budget can meet the UE's request, the backlog of its queue is always one. Although the backlog of $Z(\tau)$ is not always one when the energy consumption for MSP responding to the requests of UEs exceeds its budget $\overline{C}$. However, we know that the backlog of $Z(\tau)$ is always bounded by $O(V)$ according to Theorem 2. Moreover, the curves in the figure shows that the average backlog gradually becomes stable and follows $\lim_{T\to\infty}\mathbb{E}\{Z(\tau)/T\}=0$ under different $\overline{C}$. That indicates that Algorithm NE will satisfy the cost budget of MSP in the long-term. It can also be seen form Fig. 8 that the average energy consumption of MSP with different $\overline{C}$ can always satisfy its budget in the long-term. Moreover, we see that the MSP can increase its budget to minimize the average latency of UEs while improving QoS.

6.2.4 The Impact of N

As shown in Fig. 9, as N increases, the average latency of UEs and the cost of MSP increases. The reason lies in that the service resources of the MSP are finite, thus increasing the average latency and cost when the number of UEs increasing. Fig. 10 shows the impact of the number of UEs on the average latency for the UEs to determine the strategies, i.e., the overhead of the proposed algorithms. Since the algorithms determine the strategies in a distributed manner, an increase in the number of UEs and MEC servers does not necessarily increase the average latency to determine the strategies. Thus, we can conclude that the proposed algorithms are still effective as the scale of data increases. Moreover, as shown in the figure, it can be seen that the average latency in determining the strategies is less than five seconds. Experimental results show that the algorithms proposed in this paper are not suitable for millisecond services. However, not all application scenario requires the MSP to determine the service allocation strategies within milliseconds, such as edge caching. In edge caching, the MSP updates the cache policy every certain period of time, such as several minutes or tens of minutes [10]. Thus, the developed algorithms can be applied to decide cache policy in MEC, as well as the scenario where the delay is not particularly strictly required in the real-world, such as the second-level service and minute-level service. Moreover, in this paper, UE_i will not initiate a new request when its former request is not completed. As shown in Figs. 9 and 10, the latency that MSP responds to UE's request is typically tens of seconds, and is longer than the delay in determining a strategy. Thus, the average latency for UEs to determine a strategy is acceptable.

6.2.5 The Impact of α

A smaller value of α means that more MEC servers are involved in responding to the UEs' requests, and the service range of the servers is larger. In the real-world, MSP can trade-off its cost and QoS by controlling α to determine the strategies that are acceptable to both parties. As shown in Fig. 11, compared with other cases, when $\alpha=0.6$, the average latency of UEs is less. The reason is that reducing α causes some MEC servers far away from UEs to execute the UEs' requests, thereby increasing the transmission latency and reducing QoS. It

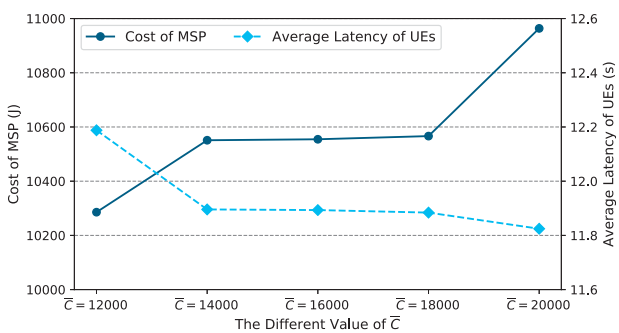

Fig. 8. The impact of budget $\overline{C}$ on the average latency of UEs and the cost of MSP.

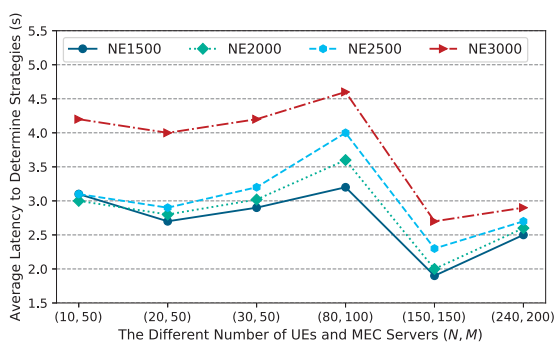

Fig. 10. Average latency for UEs to determine strategies under the different scenario (i.e., different number of UEs and MEC servers).

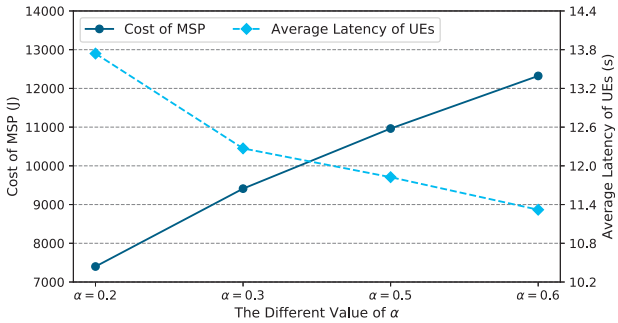

Fig. 11. The impact of α on the average latency of UEs and the cost of MSP.

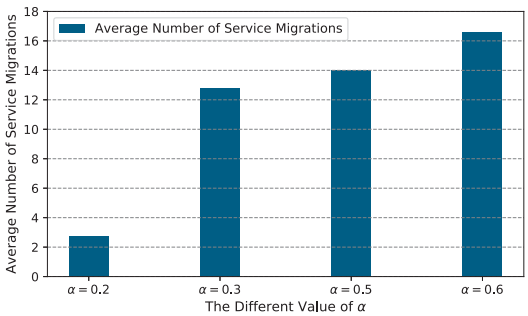

Fig. 12. The impact of α on the average number of service migrations.

can also be known that the cost of MSP increases with α increases. As shown in Fig. 12, this is because the smaller service range leads to frequent service migrations, which increases the cost of MSP for deploying service instances to follow the movement of UEs. Based on the computing resource allocation model, i.e., Equation (3), reducing the service range of a server is equivalent to increasing the number of tasks that the server needs to handle, thus also resulting in an increase in MSP cost.

7 Conclusion and Future Work

To study the budget-constrained service allocation optimization problem for MSP, in this work, we formulate the problem as a long-term QoS improvement problem while satisfying the MSP's budget. We first transform the long-term QoS improvement problem into a set of real-time sub-problems and develop a Lyapunov optimization method based algorithm. To improve the performance of service allocation for UEs in a large scale MEC environment, we further formulate the sub-problems as a non-cooperative game, and develop the algorithms to find the Nash equilibrium and determine the best service allocation strategies for each UE. Experimental results show that the algorithms can take into account QoS and budget of MSP at the same time, and perform better compared to five other common schemes.

In the paper, we assume that the network condition of UEs are constant. However, the network environment of MEC is very complicated in reality. Moreover, as intelligent applications enter the millisecond era, the demand for the millisecond service allocation mechanisms is becoming more and more urgent. The developed algorithms cannot effectively cope with the situation where UEs upload tasks multiple times in a very short period of time (such as at the time scale of millisecond). Thus, in the future, we will investigate the millisecond service allocation mechanism in a more complex network environment to develop the service allocation algorithms that are more suitable for demands of the real-world.

Acknowledgments

The authors would like to thank the associate editor and anonymous reviewers for their comments which are very important to improve the quality of the manuscript.

References

[1] T. Bahreini, H. Badri, and D. Grosu, "Mechanisms for resource allocation and pricing in mobile edge computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 667–682, Mar. 2022.

[2] J. Hu, K. Li, C. Liu, J. Chen, and K. Li, "Coalition formation for deadline-constrained resource procurement in cloud computing," *J. Parallel Distrib. Comput.*, vol. 149, pp. 1–12, 2021.

[3] Y. Hung, C. Wang, and R. Hwang, "Optimizing social welfare of live video streaming services in mobile edge computing," *IEEE Trans. Mob. Comput.*, vol. 19, no. 4, pp. 922–934, Apr. 2020.

[4] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for dnn-based smart IoT systems in cloud-edge environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 683–697, Mar. 2022.

[5] A. Aissioui, A. Ksentini, A. M. Guéroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5302–5316, Jun. 2018.

[6] S. Chen, L. Jiao, F. Liu, and L. Wang, "EdgeDR: An online mechanism design for demand response in edge clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 2, pp. 343–358, Feb. 2022.

[7] Y. Xu, J. Li, Z. Lu, J. Wu, P. C. K. Hung, and A. Alelaiwi, "ARVMEC: Adaptive recommendation of virtual machines for IoT in edge-cloud environment," *J. Parallel Distrib. Comput.*, vol. 141, pp. 23–34, 2020.

[8] Orcal, "Virtualbox," 2010. Accessed: Dec. 2021. [Online]. Available: https://www.virtualbox.org/

[9] N. Zhao *et al.*, "Large-scale analysis of docker images and performance implications for container storage systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 4, pp. 918–930, Apr. 2021.

[10] Y. Guan, X. Zhang, and Z. Guo, "Prefcache: Edge cache admission with user preference learning for video content distribution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 4, pp. 1618–1631, Apr. 2021.

[11] L. Chen and J. Xu, "Budget-constrained edge service provisioning with demand estimation via bandit learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2364–2376, Oct. 2019.

[12] J. Hu, K. Li, C. Liu, and K. Li, "Game-based task offloading of multiple mobile devices with QoS in mobile edge computing systems of limited computation capacity," *ACM Trans. Embedded Comput. Syst.*, vol. 19, no. 4, pp. 29:1–29:21, 2020.

[13] W. Liu and Y. Shoji, "Edge-assisted vehicle mobility prediction to support V2X communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 10227–10238, Oct. 2019.

[14] Z. Liao, Y. Ma, J. Huang, J. Wang, and J. Wang, "HOTSPOT: A UAV-assisted dynamic mobility-aware offloading for mobile-edge computing in 3-D space," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10 940–10 952, Jul. 2021.

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

[16] J. Li, W. Liang, M. Huang, and X. Jia, "Reliability-aware network service provisioning in mobile edge-cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1545–1558, Jul. 2020.

[17] J. Guo, C. Li, Y. Chen, and Y. Luo, "On-demand resource provision based on load estimation and service expenditure in edge cloud environment," *J. Netw. Comput. Appl.*, vol. 151, pp. 1–14, 2020.
[18] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2082–2091, Dec. 2017.
[19] T. D. Nguyen, E. N. Huh, and M. Jo, "Decentralized and revised content-centric networking-based service deployment and discovery platform in mobile edge computing for IoT devices," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4162–4175, Jun. 2019.
[20] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377–390, Feb. 2021.
[21] Z. Xiang, S. Deng, J. Taheri, and A. Y. Zomaya, "Dynamical service deployment and replacement in resource-constrained edges," *Mobile Netw. Appl.*, vol. 25, no. 2, pp. 674–689, 2020.
[22] S. Deng *et al.*, "Optimal application deployment in resource constrained distributed edges," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1907–1923, May 2021.
[23] S. Deng, C. Zhang, C. Li, J. Yin, S. Dustdar, and A. Y. Zomaya, "Burst load evacuation based on dispatching and scheduling in distributed edge networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 1918–1932, Aug. 2021.
[24] S. Deng, Y. Chen, G. Chen, S. Ji, J. Yin, and A. Zomaya, "Incentive-driven proactive application deployment and pricing on distributed edges," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3096860.
[25] Z. Rejiba, X. Masip-Bruin , and E. Marín-Tordera, "A survey on mobility-induced service migration in the fog, edge, and related computing paradigms," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–33, Sep. 2019.
[26] G. Liu, Z. Xiao, G. Tan, K. Li, and A. T. Chronopoulos, "Game theory-based optimization of distributed idle computing resources in cloud environments," *Theor. Comput. Sci.*, vol. 806, pp. 468–488, 2020.
[27] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. S. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, 2015.
[28] X. Zhang, J. Zhang, Z. Liu, Q. Cui, X. Tao, and S. Wang, "MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3296–3309, Mar. 2020.
[29] C. Wu, L. Zhang, Q. Li, Z. Fu, W. Zhu, and Y. Zhang, "Enabling flexible resource allocation in mobile deep learning systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 2, pp. 346–360, Feb. 2019.
[30] Z. Zhou, X. Chen, W. Wu, D. Wu, and J. Zhang, "Predictive online server provisioning for cost-efficient IoT data streaming across collaborative edges," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2019, pp. 321–330.
[31] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.
[32] L. Chen, J. Xu, S. Ren, and P. Zhou, "Spatio-temporal edge service placement: A bandit learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8388–8401, Dec. 2018.
[33] L. Chen and J. Xu, "Budget-constrained edge service provisioning with demand estimation via bandit learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2364–2376, Oct. 2019.
[34] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.
[35] O. Muñoz, A. Pascual-Iserte , and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.
[36] M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Heterogeneous edge offloading with incomplete information: A minority game approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2139–2154, Sep. 2020.
[37] K. Li, "Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2019.2904680.
[38] X. Qiu, Y. Dai, Y. Xiang, and L. Xing, "Correlation modeling and resource optimization for cloud service with fault recovery," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 693–704, Jul.–Sep. 2019.
[39] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: Towards an integrated service provisioning model," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1116–1129, Jun. 2020.
[40] W. Yuan and N. Klara, "Energy-efficient CPU scheduling for multimedia applications," *ACM Trans. Comput. Syst.*, vol. 24, no. 3, pp. 292–331, Aug. 2006.
[41] B. Zhai, D. T. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proc. 41th Des. Autom. Conf., San Diego, CA, USA*, S. Malik, L. Fix, and A. B. Kahng, Eds. 2004, pp. 868–873.
[42] M. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan and Claypool, 2010.
[43] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[44] G. Grätzer, Universal Algebra. Berlin, Germany: Springer, 2008.
[45] A. Nedic, "Subgradient methods for convex minimization," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2002.
[46] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Processor-network speed scaling for energy-delay tradeoff in smartphone applications," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1647–1660, Jun. 2016.
[47] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proc. 2nd ACM/IEEE Symp. Edge Comput., San Jose / Silicon Valley, CA, USA*, J. Zhang, M. Chiang, and B. M. Maggs, Eds. 2017, pp. 11:1–11:13.
[48] C. Liu, K. Li, and K. Li, "Minimal cost server configuration for meeting time-varying resource demands in cloud centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2503–2513, Nov. 2018.

Yan Ding (Student Member, IEEE) is currently working toward the doctorate degree with Hunan University, Changsha, China. He has authored or coauthored five research papers in international conferences and journals, including *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Industrial Informatics*, *Journal of Parallel and Distributed Computing*, *Computers & Security*, and IEEE ISPA 2019. His research interests include parallel computing, fog computing, mobile edge computing, and artificial intelligence. He was the recipient of the Outstanding Paper Award in IEEE ISPA 2019. He is a student member of CCF.

Kenli Li (Senior Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. From 2004 to 2005, he was a visiting scholar with the University of Illinois at Urbana-Champaign. He is currently a full professor of computer science and technology with Hunan University, the dean of the College of Information Sciences and Engineering of Hunan University, and the director of National Supercomputing Center, Changsha. He has authored or coauthored more than 160 research papers in international conferences and journals, including *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *Journal of Parallel and Distributed Computing*, ICPP, and ICDCS. His research interests include parallel computing, high-performance computing, and grid and cloud computing. He is currently on the editorial board of *IEEE Transactions on Computers*. He is an outstanding member of CCF.

Chubo Liu (Member, IEEE) is currently an associate professor of computer science and technology with Hunan University. He has authored or coauthored more than 20 papers in journals and conferences, including *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Industrial Informatics*, ICPADS, HPCC, and NPC. His research interests include game theory, approximation and randomized algorithms, and cloud and edge computing. He was the recipient of the Best Paper Award in IFIP NPC 2019, Outstanding Paper Award in IEEE ISPA 2019, and the IEEE TCSC Early Career Researcher Award in 2019. He is a member of CCF.

Zhuo Tang is currently a professor with the College of Computer Science and Electronic Engineering, Hunan University. He has authored or coauthored almost 90 journal articles and book chapters. His research interests include distributed computing system, cloud computing, and parallel processing for Big Data, including distributed machine learning, security model, parallel algorithms, and resources scheduling and management. He is a member of ACM and CCF.

Keqin Li (Fellow, IEEE) is currently a SUNY distinguished professor of computer science with the State University of New York and a National distinguished professor with Hunan University, China. He has authored or coauthored more than 810 journal articles, book chapters, and refereed conference papers He holds over 60 patents announced or authorized by the Chinese National Intellectual Property Administration. His research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, Big Data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing. He is among the world's top ten most influential scientists in distributed computing based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an associate editor for *ACM Computing Surveys* and *CCF Transactions on High Performance Computing*. He was on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Services Computing*, and *IEEE Transactions on Sustainable Computing*. He was the recipient of several best paper awards.