Received April 30, 2019, accepted May 21, 2019, date of publication May 27, 2019, date of current version June 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919106

How to Stabilize a Competitive Mobile Edge Computing Environment: A Game Theoretic Approach

KEQIN LI, (Fellow, IEEE)
College of Information Science and Engineering, Hunan University, Changsha 410082, China
Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
e-mail: lik@newpaltz.edu

This work was supported in part by the National Natural Science Foundation of China under Grant 61876061, in part by the Key Program of National Natural Science Foundation of China under Grant 61432005, and in part by the National Key Research and Development Program of China under Grant 2018YFB1003401.

ABSTRACT There are two fundamental purposes in mobile edge computing, i.e., performance enhancement and cost reduction. By offloading computation tasks to a *mobile edge cloud* (MEC), a *user equipment* (UE), also called mobile user, mobile subscriber, or mobile device, can possibly reduce its average response time, which is the main performance measure, and can possibly reduce its average power consumption. Optimizing both performance and cost may be conflicting requirements. In this paper, we optimize the *cost-performance ratio* (CPR), i.e., the power-time product, which combines performance (average response time) and cost (average power consumption) into one quantity. A unique feature in mobile edge computing is the competitiveness of mobile users, who are selfish in competing for resources in a mobile edge cloud. We take a game theoretic approach to the stabilization of a competitive mobile edge computing environment. The main contributions of the paper are summarized as follows. 1) We consider a mobile edge computing environment with multiple UEs and a single MEC. We establish an M/G/1 queueing model for the UEs and an M/G/m queueing model for the MEC. The UEs are entirely heterogeneous in terms of task characteristics, computation and communication speeds, and power consumption models for both computation and communication. 2) We analytically derive the average response time and the average power consumption of each UE and the MEC, so that cost-performance ratio optimization can be studied mathematically and rigorously. 3) We establish a non-cooperative game framework to systematically study the stabilization of a competitive mobile edge computing environment. Our framework includes a set of seven non-cooperative games among the UEs and the MEC, each attempts to minimize its payoff function, i.e., its cost-performance ratio. These games are different in terms of the number of variables to play and which variables to play. 4) We develop efficient algorithms for each player to find the best response in each game. All these algorithms are the poly-log time in the length of an initial search interval and the accuracy requirement. We also develop an iterative algorithm to find the Nash equilibrium of the games. 5) We demonstrate the numerical examples of our algorithms and performance data of our games for the idle-speed model and the constant-speed model respectively.

INDEX TERMS Average power consumption, average response time, computation offloading, cost-performance ratio, mobile edge computing, Nash equilibrium, non-cooperative game, queueing model.

I. INTRODUCTION

A. MOTIVATION

There are two fundamental purposes in mobile edge computing, i.e., performance enhancement and cost reduction [17]. By offloading computation tasks to a *mobile edge cloud* (MEC), a *user equipment* (UE), also called mobile user, mobile subscriber, or mobile device, can possibly reduce its average response time, which is a main performance measure, and can possible reduce its average power consumption, which is a main cost measure. By reducing the average response time and reducing the average power consumption,

The associate editor coordinating the review of this manuscript and approving it for publication was Petros Nicopolitidis.

a UE creates an illusion of a mobile device with stronger computing power and longer battery lifetime [8], [12], [13].

There are three essential parameters that a UE can control or choose to enhance its performance and to reduce its cost. All these three parameters affect the average response time and the average power consumption, and their impact on performance and cost is sophisticated and deserves serious investigation. (1) The first parameter is the amount of task offloading. It seems that increasing the amount of offloading will decrease the average response time. However, this is true only for non-offloaded tasks, since the workload on a UE is reduced. The average response time of offloaded tasks may be increased if the communication time is too long and/or an MEC is overloaded. While increasing the amount of offloading may decrease the average power consumption for computation of a UE, it definitely increases the average power consumption for communication. (2) The second parameter is the server execution speed. While increasing the server execution speed of a UE decreases the average response time of non-offloaded tasks, it increases the average power consumption for computation. (3) The third parameter is the data communication speed. While increasing the data communication speed of a UE decreases the average response time of offloaded tasks, it increases the transmission power and the average power consumption for communication.

Similarly, there is one essential parameter, i.e., the server execution speed, that an MEC can control or choose to enhance its performance and to reduce its cost. It is clear that increasing the server execution speed decreases the average response time, but increases the average power consumption.

Optimizing both performance and cost may be conflicting requirements. There are different ways to deal with the performance and cost tradeoff, for instances, minimization of average response time with average power consumption constraint, and minimization of average power consumption with average response time constraint [21], and joint performance and cost optimization [7]. In this paper, we optimize the *cost-performance ratio* (CPR), i.e., the power-time product, which combines performance (average response time) and cost (average power consumption) into one quantity. A UE can optimize its CPR by proper choice of its computation offloading strategy and computation/communication speeds. An MEC can optimize its CPR by proper choice of its server execution speed.

A unique feature in mobile edge computing is competitiveness of mobile users, who are selfish in competing for resources in a mobile edge cloud. Therefore, collective optimization of the overall performance and/or cost of all mobile users is not interesting to anyone. Optimization of computation offloading strategy and computation/communication speeds should be carried out for each UE individually and separately, while other UEs are also doing so. In addition, an MEC can also join such a non-cooperative game. Each player in the game attempts to find his best response to the current situation by finding his best choice of the variables that minimize his CPR. We are interested in how the stable situation looks like, a situation where no one can reduce his CPR anymore and no one wants to make further change. The motivation of our investigation is to conduct a mathematical study of the above competitive and non-cooperative game and to show that such a stable situation does exist and can be found algorithmically, numerically, and efficiently.

B. SUMMARY OF CONTRIBUTIONS

In this paper, we take a game theoretic approach to stabilization of a competitive mobile edge computing environment. The main contributions of the paper are summarized as follows.

- We consider a mobile edge computing environment with multiple UEs and a single MEC. We establish an M/G/1 queueing model for the UEs and an M/G/m queueing model for the MEC. The UEs are entirely heterogeneous in terms of task characteristics, computation and communication speeds, and power consumption models for both computation and communication.
- We analytically derive the average response time and the average power consumption of each UE and the MEC, so that cost-performance ratio optimization can be studied mathematically and rigorously.
- We establish a non-cooperative game framework to systematically study stabilization of a competitive mobile edge computing environment. Our framework includes a set of seven non-cooperative games among the UEs and the MEC, each attempts to minimize its payoff function, i.e., its cost-performance ratio. These games are different in terms of the number of variables to play and which variables to play.
- We develop efficient algorithms for each player to find the best response in each game. All these algorithms are poly-log time in the length of an initial search interval and the accuracy requirement. We also develop an iterative algorithm to find the Nash equilibrium of the games.
- We demonstrate numerical examples of our algorithms and performance data of our games for the idle-speed model and the constant-speed model respectively.

The rest of the paper is organized as follows. In Section 2, we review related research. In Section 3, we establish mathematical models. In Section 4, we present power consumption models for both computation and communication. In Section 5, we establish a game formulation of a competitive mobile edge computing environment. In Section 6, we develop our algorithms. In Section 7, we demonstrate numerical examples and performance data. In Section 8, we summarize the paper and mention further research directions.

II. RELATED WORK

In this section, we review related research. Computation offloading in mobile edge computing has been a hot research topic in recent years, and extensive investigation has been conducted. The reader is referred to [2], [15], [17], [24], [27] for recent comprehensive surveys.

Several researchers have considered the important issue of performance and cost tradeoff in mobile edge computing. Mao *et al.* investigated the tradeoff between two critical but conflicting objectives in multi-user MEC systems, namely, the power consumption of mobile devices and the execution delay of computation tasks, by considering a stochastic optimization problem, for which, the CPU frequency, the transmit power, as well as the bandwidth allocation should be determined for each device in each time slot [25]. You *et al.* studied optimal resource allocation for a multi-user mobile-edge computation offloading system, where each user has one task, by minimizing the weighted sum of mobile energy consumption under the constraint on computation latency, with the assumption of negligible cloud computing and result downloading time [29]. Zhang *et al.* proposed a joint computation offloading and resource allocation optimization scheme, aiming to minimize the total cost (which includes energy consumption, monetary cost, and execution latency for both computation and communication) of all mobile users, where each user has one task [31]. Zhang *et al.* studied energy-efficient computation offloading mechanisms for MEC in 5G heterogeneous networks by formulating an optimization problem to minimize the energy consumption of an offloading system with multiple mobile devices, where each device has a computation task to be completed within certain delay constraint, and the energy cost of both task computing and file transmission are taken into consideration [32].

The game theoretical approach has been employed to study computation offloading strategies of multiple users. Cao and Cai investigated the problem of multi-user computation offloading for cloudlet based mobile cloud computing in a multi-channel wireless contention environment, by formulating the multi-user computation offloading decision making problem as a non-cooperative game, where each mobile device user has one computation task with the same number of CPU cycles and attempts to minimize a weighted sum of execution time and energy consumption [4]. Chen formulated a decentralized computation offloading decision making problem among mobile device users as a decentralized computation offloading game, where each mobile device user has a computationally intensive and delay sensitive task and minimizes a weighted sum of computational time and energy consumption [9]. Chen *et al.* studied the multi-user computation offloading problem for mobile-edge cloud computing in a multi-channel wireless interference environment, and showed that it is NP-hard to compute a centralized optimal solution, and hence adopted a game theoretic approach to achieving efficient computation offloading in a distributed manner [10]. Liu *et al.* built a cooperative game based framework for quality of service (QoS) guaranteed offloading in a multiple MECs environment, such that the number of tasks whose QoS requirements are satisfied is maximized, where both UEs and MECs are players, and each UE has one task [22]. Ma *et al.* researched computation offloading strategies of multiple users via multiple wireless access points by taking energy consumption and delay (including computing and transmission delay) into account, and presented a game-theoretic analysis of the computation offloading problem while mimicking the selfish nature of the individuals [23]. However, all the above works only consider the case of multiple users, where each user has only a single task.

For multiple users, where each has multiple tasks, Chen *et al.* constructed a non-cooperative game model to find an optimal computation offloading policy for each UE to minimize a weighted sum of energy consumption and time consumption [7]. However, the method adopted is discrete combinatorial optimization, not continuous stochastic optimization. Cardellini *et al.* considered a usage scenario where multiple non-cooperative mobile users share the limited computing resources of a close-by cloudlet and can selfishly decide to send their computations to any of the three tiers, i.e., a local tier of mobile nodes, a middle tier (cloudlets) of nearby computing nodes, and a remote tier of distant cloud servers [5]. However, the above study employed the M/M/1 queueing model, which is not able to capture the heterogeneity of mobile devices. Furthermore, the above study did not consider multiple heterogeneous MECs. In fact, all the above studies are for a single MEC.

There has been investigation concerning multiple MECs. Tran and Pompili studied the problem of joint task offloading and resource allocation in a multi-cell and multi-server MEC system in order to maximize users' task offloading gains, which are measured by the reduction in task completion time and energy consumption, by considering task offloading decision, uplink transmission power of mobile users, and computing resource allocation in the MEC servers [28]. However, this study did not use the game theoretic approach to dealing with competitive and selfish mobile users. Li *et al.* considered multiple heterogeneous mobile users competing for resources from multiple heterogeneous mobile edge clouds, where each UE and MEC is characterized by an M/G/1 queueing system, and used the game theoretic approach to finding the optimal computation offloading strategy for each mobile user when a mobile computing environment becomes stabilized [20]. However, the cost of energy consumption was not taken into consideration in the payoff function, but only the average response time.

Our research in this paper considers multiple heterogeneous UEs, each having an endless sequence of computational tasks, and a powerful multiserver MEC, with the goal of minimizing a combined metric of performance and cost, using both queueing theory and game theory.

III. QUEUEING MODELS

To rigorously investigate stabilization of a competitive mobile edge computing environment, we need to establish mathematical models. We consider a mobile edge computing environment with multiple UEs and a single MEC (see Figure 1), where there are n mobile user equipments, i.e., UE_1, $\text{UE}_2, \ldots$, UE_n, and a mobile edge cloud MEC.

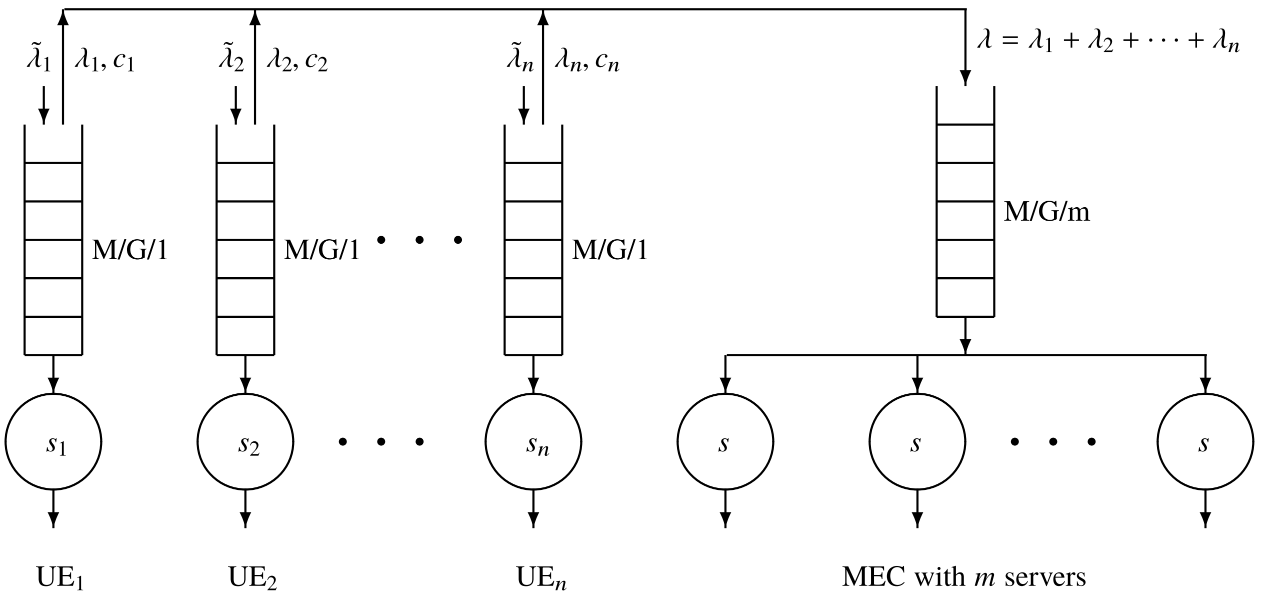

FIGURE 1. A mobile edge computing environment with multiple competitive UEs and a single MEC.

Throughout the paper, we use $\bar{y}$ to represent the expectation of a random variable y. Table 1 gives a list of the notations and their definitions used in this paper. (The main symbols of UE$_i$ are: $\tilde{\lambda}_i, \lambda_i, \bar{r_i}, \overline{r_i^2}, \bar{d_i}, \overline{d_i^2}, s_i, c_i, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$; and $x_i, \bar{x_i}, \overline{x_i^2}, \rho_i, W_i, T_{\mathrm{local},i}, T_{\mathrm{remote},i}, T_i, P_{d,i}, P_{\mathrm{comp},i}, P_{t,i}, P_{\mathrm{comm},i}, P_i, R_i$. The main symbols of the MEC are: m, s, ξ, α, P_s; and $\lambda, \bar{x}, \overline{x^2}, \sigma, C, \rho, W, \hat{W}, D, T, P_d, P, R$.)

A. THE USER MODEL: M/G/1

In this paper, each UE$_i$ is treated as an M/G/1 queueing system. That is, UE$_i$ is actually a server. Such a server allows task inter-arrival times to follow an exponential distribution and task execution times to follow an arbitrary probability distribution (a fairly general model without extra assumptions). There is a Poisson stream of computation tasks with arrival rate $\tilde{\lambda}_i$ (measured by the number of arrival tasks per unit of time, e.g., second), i.e., the inter-arrival times are independent and identically distributed (i.i.d.) exponential random variables with mean $1/\tilde{\lambda}_i$. The arrival task stream is decomposed into two streams, that is, there is a Poisson stream of computation tasks with arrival rate λ_i which are offloaded to the MEC and processed remotely in the MEC, and there is a Poisson stream of computation tasks with arrival rate $\tilde{\lambda}_i - \lambda_i$ which are not offloaded to the MEC and processed locally in UE$_i$. The variable λ_i is actually a *computation offloading strategy* of UE$_i$, for all $1 \le i \le n$.

Each M/G/1 queueing system maintains a queue with infinite capacity for waiting tasks when UE$_i$ is busy in processing other tasks. The first-come-first-served (FCFS) queueing discipline is adopted.

The execution requirements (measured by the number of billion processor cycles or the number of billion instructions (BI) to be executed) of the computation tasks generated on UE$_i$ are i.i.d. random variables r_i with an arbitrary probability distribution. We assume that its mean $\bar{r_i}$ and second moment $\overline{r_i^2}$ are available, for all $1 \le i \le n$. The amount of data (measured by the number of million bits (MB)) to be communicated between UE$_i$ and the MEC for offloaded tasks are i.i.d. random variables d_i with an arbitrary probability distribution. We assume that its mean $\bar{d_i}$ and second moment $\overline{d_i^2}$ are available, for all $1 \le i \le n$.

UE$_i$ has execution speed s_i (measured by GHz or the number of billion instructions that can be executed in one second), where $1 \le i \le n$. The communication speed (measured by the number of million bits that can be transmitted in one second) between UE$_i$ and the MEC is c_i, where $1 \le i \le n$.

B. THE SERVER MODEL: M/G/m

The MEC is treated as an M/G/m queueing system. Thus, the MEC is actually a multiserver system with mixed classes of tasks from different mobile users. There is a Poisson stream of computation tasks with arrival rate λ to the MEC, where $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n$, and λ_i is the arrival rate of the Poisson stream of computation tasks offloaded from UE$_i$, for all $1 \le i \le n$. The MEC has m identical servers, where m is the size of the multiserver system. The M/G/m queueing system maintains a queue with infinite capacity for waiting tasks when all the m servers are busy in processing other tasks. The FCFS queueing discipline is adopted. The execution speed (measured by GHz or the number of billion instructions that can be executed in one second) of the m servers is s.

IV. POWER CONSUMPTION MODELS

In this section, we present power consumption models for both computation and communication in mobile edge computing. All powers are measured by Watts.

TABLE 1. Summary of notations and definitions.

Notation	Definition
Queueing Models and Power Consumption Models	
n	the number of mobile user equipments
$\tilde{\lambda}_i$	the arrival rate of computation tasks to UE$_i$
λ_i	the arrival rate of offloaded computation tasks from UE$_i$ to the MEC
r_i	the execution requirements of the computation tasks generated on UE$_i$
$\overline{r_i}, \overline{r_i^2}$	mean and second moment of r_i
d_i	the amount of data to be communicated between UE$_i$ and the MEC for offloaded tasks
$\overline{d_i}, \overline{d_i^2}$	mean and second moment of d_i
s_i	the execution speed of UE$_i$
c_i	the communication speed between UE$_i$ and the MEC
x_i	the execution times of non-offloaded tasks processed locally in UE$_i$
$\overline{x_i}, \overline{x_i^2}$	mean and second moment of x_i
ρ_i	the utilization of UE$_i$
W_i	the average waiting time of the tasks in UE$_i$
$T_{\text{local},i}$	the average response time of non-offloaded tasks processed locally in UE$_i$
$T_{\text{remote},i}$	the average response time of offloaded tasks from UE$_i$ processed remotely in the MEC
T_i	the average response time of all non-offloaded and offloaded tasks generated on UE$_i$
$\xi_i, \alpha_i, P_{d,i}, P_{s,i}$	parameters of the computation power consumption model of UE$_i$
$w_i, \beta_i, P_{t,i}$	parameters of the communication power consumption model of UE$_i$
$P_{\text{comp},i}$	the average power consumption for computation of UE$_i$
$P_{\text{comm},i}$	the average power consumption for communication of UE$_i$
P_i	the average power consumption of UE$_i$
R_i	$= P_iT_i$, the cost-performance ratio (i.e., power-time product) of UE$_i$
λ	$=\lambda_1 + \lambda_2 + \cdots + \lambda_n$, the arrival rate of computation tasks to the MEC
m	the number of servers in the MEC
s	the execution speed of the MEC
x	the execution times of the tasks in the MEC
$\overline{x}, \overline{x^2}$	mean and second moment of x
ρ	the utilization of the MEC
W	the average waiting time of all tasks in the MEC
$\hat{W}$	the average waiting time of all tasks in an M/M/m with the same utilization as an M/G/m
D	an auxiliary variable introduced for ease of presentation
T	the average response time of all tasks in the MEC
ξ, α, P_d, P_s	parameters of the power consumption model of the MEC
P	the average power consumption of the MEC
R	$= PT$, the cost-performance ratio (i.e., power-time product) of the MEC
Game Theory	
$\mathbb{R}^m$	an Euclidean space
K	a convex set of $\mathbb{R}^m$
$\mathbf{x}, \mathbf{y}$	points in K
$f(\mathbf{x})$	a convex function on K
$\mathbf{H}(f(\mathbf{x}))$	the Hessian matrix of $f(\mathbf{x})$
K_i	the set of strategies of the ith player
$\mathbf{x}_i$	$= (x_{i,1}, x_{i,2}, \ldots, x_{i,m_i}) \in K_i \subseteq \mathbb{R}^{m_i}$, the strategy of the ith player
$\mathcal{K}$	$= K_1 \times K_2 \times \cdots \times K_n$
$\mathbf{x}$	$= (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathcal{K}$, the overall vector of all players' variables, i.e., an action profile
$\mathbf{x}_{-i}$	$= (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n)$, the vector of all players' variables except that of player i
$f_i(\mathbf{x}_i, \mathbf{x}_{-i})$	the payoff function of the ith player
$\mathbf{f}$	$= (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))$
$\mathcal{G}$	$= (\mathcal{K}, \mathbf{f})$, a non-cooperative game with n players
$\mathbf{x}^*$	$= (\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_n^*) \in \mathcal{K}$, a pure strategy Nash equilibrium
Algorithm Theory	
CO(K, f)	a convex optimization problem
I	the maximum length of all initial search intervals
Δ	numerical approximation for derivative
ϵ, δ	the accuracy requirement
N	the number of rounds

A. COMPUTATION

Power dissipation and circuit delay in digital CMOS circuits can be accurately modeled by simple equations, even for complex microprocessor circuits. CMOS circuits have dynamic, static, and short-circuit power dissipation; however, the dominant component in a well designed circuit is dynamic power consumption $P_{d,i}$ (i.e., the switching component of power) of UE$_i$, which is approximately $P_{d,i} = a_i C_i V_i^2 f_i$, where a_i is an activity factor, C_i is the loading capacitance, V_i is the supply voltage, and f_i is the clock frequency [6]. In the ideal case, the supply voltage and the clock frequency are related in such a way that $V_i \propto f_i^{\phi_i}$ for some constant $\phi_i > 0$ [30]. The processor execution speed s_i is usually linearly proportional to the clock frequency, namely, $s_i \propto f_i$. For ease of discussion, we will assume that $V_i = b_i f_i^{\phi_i}$ and $s_i = c_i f_i$, where b_i and c_i are some constants. Hence, we know that the dynamic power consumption is

$$P_{d,i} = a_i C_i V_i^2 f_i = a_i b_i^2 C_i f_i^{2\phi_i+1} = (a_i b_i^2 C_i / c_i^{2\phi_i+1}) s_i^{2\phi_i+1},$$

which can be simplified as

$$P_{d,i} = \xi_i s_i^{\alpha_i},$$

where $\xi_i = a_i b_i^2 C_i / c_i^{2\phi_i+1}$, and $\alpha_i = 2\phi_i + 1$. For instance, by setting $a_i C_i = 7.0$, $b_i = 1.16$, $c_i = 1.0$, $\phi_i = 0.5$, $\alpha_i = 2\phi_i + 1 = 2.0$, and $\xi_i = a_i b_i^2 C_i / c_i^{\alpha_i} = 9.4192$, the value of $P_{d,i}$ calculated by the equation $P_{d,i} = a_i C_i V_i^2 f_i = \xi_i s_i^{\alpha_i}$ is reasonably close to that in [14] for the Intel Pentium M processor.

We will consider two types of server speed and power consumption models. In the *idle-speed model*, a server runs at zero speed when there is no task to perform. Since the power for speed s_i is $\xi_i s_i^{\alpha_i}$, the average amount of energy consumed by UE$_i$ in one second is $\rho_i P_{d,i} = \rho_i \xi_i s_i^{\alpha_i}$, where we notice that the speed of a server is zero when it is idle, and $\rho_i = (\tilde{\lambda}_i - \lambda_i)(\overline{r_i}/s_i)$ is the utilization of UE$_i$ to be derived in Section 5.2.1. The average amount of energy consumed by UE$_i$ in one second, i.e., the power supply to UE$_i$, is $P_{\text{comp},i} = \rho_i \xi_i s_i^{\alpha_i}$. Since a server still consumes some amount of base power $P_{s,i}$ even when it is idle (assume that an idle server consumes certain base power $P_{s,i}$, which includes static power dissipation, short circuit power dissipation, and other leakage and wasted power [19]), we will include $P_{s,i}$ in $P_{\text{comp},i}$, i.e.,

$$P_{\text{comp},i} = \rho_i P_{d,i} + P_{s,i} = \rho_i \xi_i s_i^{\alpha_i} + P_{s,i},$$

In the *constant-speed model*, a server still runs at the speed s_i even if there is no task to perform. Again, we use $P_{\text{comp},i}$ to represent the power allocated to UE$_i$. Since the power for speed s_i is $P_{d,i} = \xi_i s_i^{\alpha_i}$, the power allocated to UE$_i$ is

$$P_{\text{comp},i} = P_{d,i} + P_{s,i} = \xi_i s_i^{\alpha_i} + P_{s,i}.$$

Similarly, we use P_d, P_s, and P to represent the dynamic, static, and average power consumption of the MEC. Then, we have

$$P_d = \xi s^{\alpha},$$

and

$$P = m(\rho P_d + P_s) = m(\rho \xi s^{\alpha} + P_s),$$

for the idle-speed model, and

$$P = m(P_d + P_s) = m(\xi s^{\alpha} + P_s),$$

for the constant-speed model, where s is the execution speed of the MEC, and ξ and α are some constants.

B. COMMUNICATION

In addition to power consumption for computation, a UE also consumes power for communication. Let $P_{t,i}$ be the transmission power of UE$_i$, where $1 \le i \le n$. The data transmission rate c_i from UE$_i$ to the MEC is

$$c_i = w_i \log_2(1 + \beta_i P_{t,i}),$$

where w_i is the channel bandwidth and β_i is a combined quantity which summarizes various factors such as the channel gain between UE$_i$ and the MEC, the interference on the communication channel caused by other devices' data transmission to the same MEC, and the background noise power. Since the average communication time for one offloaded task from UE$_i$ to the MEC is $\overline{d_i}/c_i$, the average energy consumption to complete data transmission for one offloaded task from UE$_i$ to the MEC is $P_{t,i}(\overline{d_i}/c_i)$, where

$$P_{t,i} = \frac{2^{c_i/w_i} - 1}{\beta_i}.$$

Since there are λ_i tasks offloaded from UE$_i$ to the MEC in one second, the average energy consumption of data transmission for offloaded tasks from UE$_i$ to the MEC in one second, i.e., the average power consumption for communication of UE$_i$, is

$$P_{\text{comm},i} = \lambda_i \frac{\overline{d_i}}{c_i} P_{t,i} = \lambda_i \frac{\overline{d_i}}{c_i} \cdot \frac{2^{c_i/w_i} - 1}{\beta_i}.$$

By adding the average power consumption for computation and communication together, we get the average power consumption P_i of UE$_i$ as

$$\begin{aligned} P_i &= P_{\text{comp},i} + P_{\text{comm},i} \\ &= \rho_i \xi_i s_i^{\alpha_i} + P_{s,i} + \lambda_i \frac{\overline{d_i}}{c_i} \cdot \frac{2^{c_i/w_i} - 1}{\beta_i} \\ &= (\tilde{\lambda}_i - \lambda_i)\overline{r_i} \xi_i s_i^{\alpha_i - 1} + P_{s,i} + \lambda_i \frac{\overline{d_i}}{c_i} \cdot \frac{2^{c_i/w_i} - 1}{\beta_i}, \end{aligned}$$

for the idle-speed model, and

$$\begin{aligned} P_i &= P_{\text{comp},i} + P_{\text{comm},i} \\ &= \xi_i s_i^{\alpha_i} + P_{s,i} + \lambda_i \frac{\overline{d_i}}{c_i} \cdot \frac{2^{c_i/w_i} - 1}{\beta_i}, \end{aligned}$$

for the constant-speed model.

V. A GAME FORMULATION

We use non-cooperative games to study computation offloading strategy and computation/communication speeds optimization for non-cooperative mobile users competing for resources from a mobile edge cloud.

A. BACKGROUND INFORMATION

In this section, we describe some background information of non-cooperative game theory. (The material in this section is adapted from Sections 4.1–4.2 of [20] and included here for the sake of completeness.)

A set $K \subseteq \mathbb{R}^m$ is convex if for any two points $\mathbf{x}, \mathbf{y} \in K$, the segment joining them belongs to K, i.e.,

$$\beta\mathbf{x} + (1-\beta)\mathbf{y} \in K, \quad \text{for all } \beta \in [0,1].$$

Given a convex set $K \subseteq \mathbb{R}^m$, a function $f(\mathbf{x}) : K \to \mathbb{R}$ is said to be convex on K if for all $\mathbf{x}, \mathbf{y} \in K$ and $\beta \in [0,1]$, we have

$$f(\beta\mathbf{x} + (1-\beta)\mathbf{y}) \le \beta f(\mathbf{x}) + (1-\beta) f(\mathbf{y}).$$

It is well known [1] that a continuous and twice differentiable function $f(\mathbf{x}) : K \to \mathbb{R}$, where $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, is convex on a convex set K if and only if its Hessian matrix

$$\mathbf{H}(f(\mathbf{x})) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{m\times m}$$

of second partial derivatives is positive semidefinite on the interior of K. Let the kth leading principal minor of the symmetric matrix $\mathbf{H}(f(\mathbf{x}))$ be the determinant of its upper-left $k \times k$ submatrix, where $1 \le k \le m$. By the well known Sylvester's criterion, $\mathbf{H}(f(\mathbf{x}))$ is positive semidefinite if and only if all the leading principal minors (i.e., all these determinants) are non-negative [11].

Given a closed and convex $K \subseteq \mathbb{R}^m$ and an objective function $f(\mathbf{x}) : K \to \mathbb{R}$, which is convex and continuously differentiable on K, the convex optimization (CO) problem, denoted by $\mathrm{CO}(K, f)$, is to

$$\text{minimize } f(\mathbf{x}), \quad \text{subject to } \mathbf{x} \in K,$$

i.e., to find a solution $\mathbf{x}^* \in K$, such that

$$f(\mathbf{x}^*) \le f(\mathbf{x}), \quad \text{for all } \mathbf{x} \in K.$$

Assume that there are n players in a game. The ith player controls a variable (which represents the *strategy* of the player) $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m_i}) \in K_i \subseteq \mathbb{R}^{m_i}$, where K_i (which is the set of strategies of the ith player) is closed and convex, for all $1 \le i \le n$. Let $\mathcal{K} = K_1 \times K_2 \times \cdots \times K_n$ be the set of combinations of all players' strategies. We use the notation $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathcal{K}$ to denote the overall vector of all players' variables, and $\mathbf{x}_{-i} = (\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n)$ to denote the vector of all players' variables except that of player i. Each player has a *payoff function* $f_i(\mathbf{x}_i, \mathbf{x}_{-i}) : \mathcal{K} \to \mathbb{R}$. It is assumed that the payoff function f_i is continuously differentiable in $\mathbf{x}$ and convex as a function of $\mathbf{x}_i$ alone for every fixed $\mathbf{x}_{-i}$.

A *non-cooperative game* with n players is specified by $\mathcal{G} = (\mathcal{K}, \mathbf{f})$, where $\mathcal{K} = K_1 \times K_2 \times \cdots \times K_n$ and $\mathbf{f} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))$. The aim of player i, given other players' strategies $\mathbf{x}_{-i}$, is to choose an *action* $\mathbf{x}_i \in K_i$ that minimizes his payoff function $f_i(\mathbf{x}_i, \mathbf{x}_{-i})$, i.e., to

$$\text{minimize } f_i(\mathbf{x}_i, \mathbf{x}_{-i}), \quad \text{subject to } \mathbf{x}_i \in K_i.$$

Therefore, in an n-player non-cooperative game, we have a set of n coupled convex optimization problems $\mathrm{CO}(K_i, f_i)$, where $f_i : K_i \to \mathbb{R}$ is viewed as a function of $\mathbf{x}_i$, for all $1 \le i \le n$. A point (i.e., an *action profile*) $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathcal{K}$ is feasible if $\mathbf{x}_i \in K_i$ for all $1 \le i \le n$. The purpose of the game is to find a (pure strategy) *Nash equilibrium* (NE), i.e., a feasible point $\mathbf{x}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_n^*) \in \mathcal{K}$, such that

$$f_i(\mathbf{x}_i^*, \mathbf{x}_{-i}^*) \le f_i(\mathbf{x}_i\, \mathbf{x}_{-i}^*), \quad \text{for all } \mathbf{x}_i \in K_i,$$

holds for each player $i = 1, 2, \ldots, n$. In words, a Nash equilibrium is a feasible strategy profile $\mathbf{x}^*$ with the property that no single player i can benefit from a unilateral deviation from $\mathbf{x}_i^*$, if all other players act according to it.

It is well known that if $f_i(\mathbf{x}_i, \mathbf{x}_{-i})$ is a convex function of $\mathbf{x}_i$ for each fixed $\mathbf{x}_{-i}$, for all $1 \le i \le n$, there is a Nash equilibrium of $\mathcal{G} = (\mathcal{K}, \mathbf{f})$ [26].

B. NON-COOPERATIVE GAMES

In this section, we present seven non-cooperative games for non-cooperative mobile users to play to stabilize a competitive mobile edge computing environment.

1) THE UE PLAYERS

Based on the queueing model for the UEs in Section 3.1, we know that the execution times of non-offloaded tasks processed locally in UE$_i$ are i.i.d. random variables

$$x_i = \frac{r_i}{s_i}$$

with mean $\overline{x_i} = \overline{r_i}/s_i$ and second moment $\overline{x_i^2} = \overline{r_i^2}/s_i^2$. The utilization of UE$_i$ is

$$\rho_i = (\tilde{\lambda}_i - \lambda_i)\overline{x_i} = (\tilde{\lambda}_i - \lambda_i)\frac{\overline{r_i}}{s_i}.$$

The average waiting time of the tasks in UE$_i$ is ([16], p. 190)

$$W_i = \frac{(\tilde{\lambda}_i - \lambda_i)\overline{x_i^2}}{2(1-\rho_i)} = \frac{(\tilde{\lambda}_i - \lambda_i)(\overline{r_i^2}/s_i^2)}{2(1 - (\tilde{\lambda}_i - \lambda_i)(\overline{r_i}/s_i))}.$$

The average response time of non-offloaded tasks processed locally in UE$_i$ is

$$T_{\mathrm{local},i} = \overline{x_i} + W_i = \frac{\overline{r_i}}{s_i} + \frac{(\tilde{\lambda}_i - \lambda_i)(\overline{r_i^2}/s_i^2)}{2(1 - (\tilde{\lambda}_i - \lambda_i)(\overline{r_i}/s_i))}.$$

Let W be the average waiting time of all tasks in the MEC, which will be derived in Section 5.2.2. Then, the average response time of offloaded tasks from UE$_i$ processed remotely in the MEC is

$$T_{\mathrm{remote},i} = \frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i} + W = \frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i} + \frac{m^{m-2}}{2m!}\lambda\overline{x^2}D,$$

where $\overline{x^2}$ and D will be defined shortly in Section 5.2.2. Therefore, the average response time (measured by seconds) of all non-offloaded and offloaded tasks generated on UE$_i$ is

$$T_i = \frac{\tilde{\lambda}_i - \lambda_i}{\tilde{\lambda}_i} T_{\mathrm{local},i} + \frac{\lambda_i}{\tilde{\lambda}_i} T_{\mathrm{remote},i}$$

$$= \frac{\tilde{\lambda}_i - \lambda_i}{\tilde{\lambda}_i}(\overline{x_i} + W_i) + \frac{\lambda_i}{\tilde{\lambda}_i}\left(\frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i} + W\right)$$
$$= \frac{\tilde{\lambda}_i - \lambda_i}{\tilde{\lambda}_i}\left(\frac{\overline{r_i}}{s_i} + \frac{(\tilde{\lambda}_i - \lambda_i)(\overline{r_i^2}/s_i^2)}{2(1-(\tilde{\lambda}_i - \lambda_i)(\overline{r_i}/s_i))}\right)$$
$$+ \frac{\lambda_i}{\tilde{\lambda}_i}\left(\frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i} + \frac{m^{m-2}}{2m!}\lambda\overline{x^2}D\right).$$

Our performance measure is $1/T_i$, which is inversely proportional to the average response time T_i, the higher, the better. Our cost measure is the average power consumption P_i, the lower, the better. The *cost-performance ratio* (CPR) refers to a UE's ability to deliver performance for its cost. Generally speaking, a UE with lower CPR is more desirable, excluding other factors. In this paper, we define CPR as cost/performance

$$R_i = P_i T_i,$$

i.e., power-time product (measured by Watts-seconds).

2) THE MEC PLAYER

Based on the queueing model for the MEC in Section 3.2, we know that the total arrival rate of all offloaded tasks from the n UEs to the MEC is

$$\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n.$$

The execution times of the tasks in the MEC are i.i.d. random variables x, which is $r_i/s + d_i/c_i$ with probability λ_i/λ, with mean

$$\overline{x} = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda}\left(\frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i}\right),$$

and the second moment

$$\overline{x^2} = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda}\left(\frac{\overline{r_i^2}}{s^2} + 2\frac{\overline{r_i}}{s}\cdot\frac{\overline{d_i}}{c_i} + \frac{\overline{d_i^2}}{c_i^2}\right),$$

and the variance

$$\sigma^2 = \overline{x^2} - \overline{x}^2,$$

and the coefficient of variation

$$C = \frac{\sigma}{\overline{x}} = \sqrt{\frac{\overline{x^2}}{\overline{x}^2} - 1}.$$

It is well known that the average waiting time of all tasks in the MEC has very accurate approximation:

$$W = \left(\frac{C^2+1}{2}\right)\hat{W},$$

where $\hat{W}$ is the average waiting time of all tasks in an M/M/m queueing system with the same utilization as the M/G/m queueing system [18]. The utilization of the MEC is

$$\rho = \frac{\lambda\overline{x}}{m} = \sum_{i=1}^{n} \frac{\lambda_i}{m}\left(\frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i}\right).$$

Then, we have ([16, p. 102])

$$\hat{W} = \overline{x}\cdot\frac{p_m}{m(1-\rho)^2},$$

where

$$p_m = p_0\frac{(m\rho)^m}{m!},$$

and

$$p_0 = \left(\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!}\cdot\frac{1}{1-\rho}\right)^{-1}.$$

Since

$$C^2 + 1 = \frac{\overline{x^2}}{\overline{x}^2},$$

and

$$m\rho = \lambda\overline{x},$$

we get

$$W = \frac{\overline{x^2}}{2\overline{x}^2}\cdot\overline{x}\cdot m\rho\cdot p_0\frac{(m\rho)^{m-1}}{m!}\cdot\frac{1}{m(1-\rho)^2}$$
$$= \frac{m^{m-2}}{2m!}\lambda\overline{x^2}D,$$

where

$$D = p_0\frac{\rho^{m-1}}{(1-\rho)^2}.$$

The average response time of all tasks in the MEC is

$$T = \sum_{i=1}^{n}\frac{\lambda_i}{\lambda}T_{\text{remote},i}$$
$$= \sum_{i=1}^{n}\frac{\lambda_i}{\lambda}\left(\frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i} + W\right)$$
$$= \sum_{i=1}^{n}\frac{\lambda_i}{\lambda}\left(\frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i}\right) + W$$
$$= \sum_{i=1}^{n}\frac{\lambda_i}{\lambda}\left(\frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i}\right) + \frac{m^{m-2}}{2m!}\lambda\overline{x^2}D.$$

The cost-performance ratio of the MEC is

$$R = PT.$$

3) THE GAMES

In this paper, we consider seven non-cooperative games with $n+1$ players, i.e., UE$_1$, UE$_2$, ..., UE$_n$, and the MEC, specified by $\mathcal{G} = (\mathcal{K}, \mathbf{f})$, where $\mathcal{K} = K_1 \times K_2 \times \cdots \times K_n \times K_{n+1}$ and $\mathbf{f} = (f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), \ldots, f_n(\mathbf{x}_n), f_{n+1}(\mathbf{x}_{n+1}))$.

UE$_i$ can play with three variables, i.e., λ_i, s_i, and c_i. It is clear that all these three variables can affect both T_i and P_i. Each of these variables must be in some appropriate interval. For instance, we require

$$\lambda_i \in [\lambda_i', \lambda_i''].$$

To ensure $\rho_i < 1$, we need

$$\lambda_i' = \max\left(0, \tilde{\lambda}_i - \frac{s_i}{\overline{r_i}}\right).$$

To ensure $\rho < 1$, we need

$$\lambda_i'' = \min\left(\tilde{\lambda}_i, m\left(1 - \sum_{j\neq i}\frac{\lambda_j}{m}\left(\frac{\overline{r_j}}{s} + \frac{\overline{d_j}}{c_j}\right)\right)\Big/\left(\frac{\overline{r_i}}{s} + \frac{\overline{d_i}}{c_i}\right)\right).$$

We also require

$$s_i \in [s_i', s_i''].$$

To ensure $\rho_i < 1$, we need

$$s_i' = (\tilde{\lambda}_i - \lambda_i)\overline{r_i}.$$

We also require

$$c_i \in [c_i', c_i''],$$

for some appropriate interval $[c_i', c_i'']$.

The strategy sets and payoff functions of the seven games are specified as follows.

- Game(λ_i): $\mathbf{x}_i = \lambda_i$, $K_i = [\lambda_i', \lambda_i'']$, and $f_i(\mathbf{x}_i) = R_i(\lambda_i) = P_i(\lambda_i)T_i(\lambda_i)$.
- Game(s_i): $\mathbf{x}_i = s_i$, $K_i = [s_i', s_i'']$, and $f_i(\mathbf{x}_i) = R_i(s_i) = P_i(s_i)T_i(s_i)$.
- Game(c_i): $\mathbf{x}_i = c_i$, $K_i = [c_i', c_i'']$, and $f_i(\mathbf{x}_i) = R_i(c_i) = P_i(c_i)T_i(c_i)$.
- Game(λ_i, s_i): $\mathbf{x}_i = (\lambda_i, s_i)$, $K_i = [\lambda_i', \lambda_i''] \times [s_i', s_i'']$, and $f_i(\mathbf{x}_i) = R_i(\lambda_i, s_i) = P_i(\lambda_i, s_i)T_i(\lambda_i, s_i)$.
- Game(λ_i, c_i): $\mathbf{x}_i = (\lambda_i, c_i)$, $K_i = [\lambda_i', \lambda_i''] \times [c_i', c_i'']$, and $f_i(\mathbf{x}_i) = R_i(\lambda_i, c_i) = P_i(\lambda_i, c_i)T_i(\lambda_i, c_i)$.
- Game(s_i, c_i): $\mathbf{x}_i = (s_i, c_i)$, $K_i = [s_i', s_i''] \times [c_i', c_i'']$, and $f_i(\mathbf{x}_i) = R_i(s_i, c_i) = P_i(s_i, c_i)T_i(s_i, c_i)$.
- Game(λ_i, s_i, c_i): $\mathbf{x}_i = (\lambda_i, s_i, c_i)$, $K_i = [\lambda_i', \lambda_i''] \times [s_i', s_i''] \times [c_i', c_i'']$, and $f_i(\mathbf{x}_i) = R_i(\lambda_i, s_i, c_i) = P_i(\lambda_i, s_i, c_i)T_i(\lambda_i, s_i, c_i)$.

The MEC can play with one variable, i.e., s, which can affect both T and P. We require

$$s \in [s', s''].$$

To ensure $\rho < 1$, we need

$$s' = \left(\sum_{i=1}^{n}\frac{\lambda_i}{m}\overline{r_i}\right)\Big/\left(1 - \sum_{i=1}^{n}\frac{\lambda_i}{m}\cdot\frac{\overline{d_i}}{c_i}\right).$$

s'' should be reasonably large. In all the seven games, we have $\mathbf{x}_{n+1} = s$, $K_{n+1} = [s', s'']$, and $f_{n+1}(\mathbf{x}_{n+1}) = R(s) = P(s)T(s)$.

C. EXISTENCE OF THE NASH EQUILIBRIUM

In this section, we show the existence of the Nash equilibrium of the games. For clarity of presentation, the derivations of all the first and second order partial derivatives are moved to Appendices 1 and 2.

For the MEC, we can show that

$$M_s = \frac{\partial^2 R}{\partial s^2} > 0,$$

where

$$\frac{\partial^2 R}{\partial s^2} = T\frac{\partial^2 P}{\partial s^2} + 2\frac{\partial P}{\partial s}\frac{\partial T}{\partial s} + P\frac{\partial^2 T}{\partial s^2}.$$

For Game(λ_i), Game(s_i), and Game(c_i), we can show that

$$M_{\lambda_i} = \frac{\partial^2 R_i}{\partial \lambda_i^2} > 0,$$
$$M_{s_i} = \frac{\partial^2 R_i}{\partial s_i^2} > 0,$$

and

$$M_{c_i} = \frac{\partial^2 R_i}{\partial c_i^2} > 0,$$

where

$$n\frac{\partial^2 R_i}{\partial \lambda_i^2} = T_i\frac{\partial^2 P_i}{\partial \lambda_i^2} + 2\frac{\partial P_i}{\partial \lambda_i}\frac{\partial T_i}{\partial \lambda_i} + P_i\frac{\partial^2 T_i}{\partial \lambda_i^2},$$
$$\frac{\partial^2 R_i}{\partial s_i^2} = T_i\frac{\partial^2 P_i}{\partial s_i^2} + 2\frac{\partial P_i}{\partial s_i}\frac{\partial T_i}{\partial s_i} + P_i\frac{\partial^2 T_i}{\partial s_i^2},$$

and

$$\frac{\partial^2 R_i}{\partial c_i^2} = T_i\frac{\partial^2 P_i}{\partial c_i^2} + 2\frac{\partial P_i}{\partial c_i}\frac{\partial T_i}{\partial c_i} + P_i\frac{\partial^2 T_i}{\partial c_i^2}.$$

For Game(λ_i, s_i), Game(λ_i, c_i), and Game(s_i, c_i), we can show that the determinants of the following three matrices, i.e.,

$$\begin{bmatrix} \dfrac{\partial^2 R_i}{\partial \lambda_i^2} & \dfrac{\partial^2 R_i}{\partial \lambda_i \partial s_i} \\ \dfrac{\partial^2 R_i}{\partial s_i \partial \lambda_i} & \dfrac{\partial^2 R_i}{\partial s_i^2} \end{bmatrix},$$

$$\begin{bmatrix} \dfrac{\partial^2 R_i}{\partial \lambda_i^2} & \dfrac{\partial^2 R_i}{\partial \lambda_i \partial c_i} \\ \dfrac{\partial^2 R_i}{\partial c_i \partial \lambda_i} & \dfrac{\partial^2 R_i}{\partial c_i^2} \end{bmatrix},$$

and

$$\begin{bmatrix} \dfrac{\partial^2 R_i}{\partial s_i^2} & \dfrac{\partial^2 R_i}{\partial s_i \partial c_i} \\ \dfrac{\partial^2 R_i}{\partial c_i \partial s_i} & \dfrac{\partial^2 R_i}{\partial c_i^2} \end{bmatrix},$$

where

$$\frac{\partial^2 R_i}{\partial \lambda_i \partial s_i} = T_i\frac{\partial^2 P_i}{\partial \lambda_i \partial s_i} + \frac{\partial P_i}{\partial \lambda_i}\frac{\partial T_i}{\partial s_i} + \frac{\partial T_i}{\partial \lambda_i}\frac{\partial P_i}{\partial s_i} + P_i\frac{\partial^2 T_i}{\partial \lambda_i \partial s_i},$$
$$\frac{\partial^2 R_i}{\partial \lambda_i \partial c_i} = T_i\frac{\partial^2 P_i}{\partial \lambda_i \partial c_i} + \frac{\partial P_i}{\partial \lambda_i}\frac{\partial T_i}{\partial c_i} + \frac{\partial T_i}{\partial \lambda_i}\frac{\partial P_i}{\partial c_i} + P_i\frac{\partial^2 T_i}{\partial \lambda_i \partial c_i},$$
$$\frac{\partial^2 R_i}{\partial s_i \partial \lambda_i} = T_i\frac{\partial^2 P_i}{\partial s_i \partial \lambda_i} + \frac{\partial P_i}{\partial s_i}\frac{\partial T_i}{\partial \lambda_i} + \frac{\partial T_i}{\partial s_i}\frac{\partial P_i}{\partial \lambda_i} + P_i\frac{\partial^2 T_i}{\partial s_i \partial \lambda_i},$$
$$\frac{\partial^2 R_i}{\partial s_i \partial c_i} = T_i\frac{\partial^2 P_i}{\partial s_i \partial c_i} + \frac{\partial P_i}{\partial s_i}\frac{\partial T_i}{\partial c_i} + \frac{\partial T_i}{\partial s_i}\frac{\partial P_i}{\partial c_i} + P_i\frac{\partial^2 T_i}{\partial s_i \partial c_i},$$

$$\frac{\partial^2 R_i}{\partial c_i \partial \lambda_i} = T_i \frac{\partial^2 P_i}{\partial c_i \partial \lambda_i} + \frac{\partial P_i}{\partial c_i}\frac{\partial T_i}{\partial \lambda_i} + \frac{\partial T_i}{\partial c_i}\frac{\partial P_i}{\partial \lambda_i} + P_i \frac{\partial^2 T_i}{\partial c_i \partial \lambda_i},$$
$$\frac{\partial^2 R_i}{\partial c_i \partial s_i} = T_i \frac{\partial^2 P_i}{\partial c_i \partial s_i} + \frac{\partial P_i}{\partial c_i}\frac{\partial T_i}{\partial s_i} + \frac{\partial T_i}{\partial c_i}\frac{\partial P_i}{\partial s_i} + P_i \frac{\partial^2 T_i}{\partial c_i \partial s_i},$$

are all positive, that is,

$$M_{\lambda_i,s_i} = \frac{\partial^2 R_i}{\partial \lambda_i^2}\cdot\frac{\partial^2 R_i}{\partial s_i^2} - \frac{\partial^2 R_i}{\partial \lambda_i \partial s_i}\cdot\frac{\partial^2 R_i}{\partial s_i \partial \lambda_i} > 0,$$
$$M_{\lambda_i,c_i} = \frac{\partial^2 R_i}{\partial \lambda_i^2}\cdot\frac{\partial^2 R_i}{\partial c_i^2} - \frac{\partial^2 R_i}{\partial \lambda_i \partial c_i}\cdot\frac{\partial^2 R_i}{\partial c_i \partial \lambda_i} > 0,$$

and

$$M_{s_i,c_i} = \frac{\partial^2 R_i}{\partial s_i^2}\cdot\frac{\partial^2 R_i}{\partial c_i^2} - \frac{\partial^2 R_i}{\partial s_i \partial c_i}\cdot\frac{\partial^2 R_i}{\partial c_i \partial s_i} > 0.$$

For Game(λ_i, s_i, c_i), the Hessian matrix is

$$H_i = \begin{bmatrix} \frac{\partial^2 R_i}{\partial \lambda_i^2} & \frac{\partial^2 R_i}{\partial \lambda_i \partial s_i} & \frac{\partial^2 R_i}{\partial \lambda_i \partial c_i} \\ \frac{\partial^2 R_i}{\partial s_i \partial \lambda_i} & \frac{\partial^2 R_i}{\partial s_i^2} & \frac{\partial^2 R_i}{\partial s_i \partial c_i} \\ \frac{\partial^2 R_i}{\partial c_i \partial \lambda_i} & \frac{\partial^2 R_i}{\partial c_i \partial s_i} & \frac{\partial^2 R_i}{\partial c_i^2} \end{bmatrix},$$

which is positive definite, since the three leading principal minors of matrix H_i, i.e.,

$$M_{\lambda_i} = \frac{\partial^2 R_i}{\partial \lambda_i^2},$$
$$M_{\lambda_i,s_i} = \frac{\partial^2 R_i}{\partial \lambda_i^2}\cdot\frac{\partial^2 R_i}{\partial s_i^2} - \frac{\partial^2 R_i}{\partial \lambda_i \partial s_i}\cdot\frac{\partial^2 R_i}{\partial s_i \partial \lambda_i},$$
$$\begin{aligned} M_{\lambda_i,s_i,c_i} = {} & \frac{\partial^2 R_i}{\partial \lambda_i^2}\left(\frac{\partial^2 R_i}{\partial s_i^2}\cdot\frac{\partial^2 R_i}{\partial c_i^2} - \frac{\partial^2 R_i}{\partial s_i \partial c_i}\cdot\frac{\partial^2 R_i}{\partial c_i \partial s_i}\right) \\ & - \frac{\partial^2 R_i}{\partial \lambda_i \partial s_i}\left(\frac{\partial^2 R_i}{\partial s_i \partial \lambda_i}\cdot\frac{\partial^2 R_i}{\partial c_i^2} - \frac{\partial^2 R_i}{\partial s_i \partial c_i}\cdot\frac{\partial^2 R_i}{\partial c_i \partial \lambda_i}\right) \\ & + \frac{\partial^2 R_i}{\partial \lambda_i \partial c_i}\left(\frac{\partial^2 R_i}{\partial s_i \partial \lambda_i}\cdot\frac{\partial^2 R_i}{\partial c_i \partial s_i} - \frac{\partial^2 R_i}{\partial s_i^2}\cdot\frac{\partial^2 R_i}{\partial c_i \partial \lambda_i}\right), \end{aligned}$$

are all positive.

Due to the sophistication of the partial derivatives, analytical proofs of the positiveness of leading principal minors seem infeasible. However, they can be demonstrated numerically (see Appendix 3).

VI. SOLUTIONS TO THE GAMES

In this section, we give the solutions to our games by developing algorithms to find the best responses of all players and an iterative algorithm to find the Nash equilibrium.

A. THE BEST RESPONSE OF A UE

In this section, we develop an algorithm to find the best response of a mobile user in each game.

1) GAME(λ_i)

In this game, UE$_i$ needs to find λ_i such that

$$\frac{\partial R_i}{\partial \lambda_i} = T_i \frac{\partial P_i}{\partial \lambda_i} + P_i \frac{\partial T_i}{\partial \lambda_i} = 0.$$

Our numerical algorithm to find λ_i such that $\partial R_i/\partial \lambda_i = 0$ is given in Algorithm 1. The algorithm uses the classical bisection method (lines 2–10) based on the observation that $\partial R_i/\partial \lambda_i$ is an increasing function of λ_i (lines 5–9), since $\partial^2 R_i/\partial \lambda_i^2 > 0$. (The standard bisection method is described in [3], p. 22). Let I denote the maximum length of all initial search intervals in this paper. Then, the time complexity of Algorithm 1 is $O(\log(I/\epsilon))$. (We set $\epsilon = 10^{-7}$ in this paper.)

Algorithm 1: Find λ_i

Input: $\lambda_j, \overline{r_j}, \overline{r_j^2}, \overline{d_j}, \overline{d_j^2}, c_j$, for all $1 \le j \ne i \le n$, m, s, and $\tilde{\lambda}_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, s_i, c_i, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$.
Output: λ_i, such that $\partial R_i/\partial \lambda_i = 0$.

```
Initialize the search interval of $\lambda_i$;                     (1)
while (the length of the search interval is $\ge \epsilon$) do    (2)
$\lambda_i \leftarrow$ the middle point of the search interval;    (3)
Calculate $\partial R_i/\partial \lambda_i$;                       (4)
if ($\partial R_i/\partial \lambda_i < 0$) then                    (5)
    Change the search interval to the right half;                 (6)
else                                                              (7)
    Change the search interval to the left half;                  (8)
end if                                                            (9)
end do;                                                           (10)
$\lambda_i \leftarrow$ the middle point of the search interval;    (11)
return $\lambda_i$.                                                (12)
```

2) GAME(s_i)

In this game, UE$_i$ needs to find s_i such that

$$\frac{\partial R_i}{\partial s_i} = T_i \frac{\partial P_i}{\partial s_i} + P_i \frac{\partial T_i}{\partial s_i} = 0.$$

Our numerical algorithm to find s_i such that $\partial R_i/\partial s_i = 0$ is given in Algorithm 2 with $s_i'' = 5.0$, which is similar to Algorithm 1. The time complexity of Algorithm 2 is $O(\log(I/\epsilon))$.

3) GAME(c_i)

In this game, UE$_i$ needs to find c_i such that

$$\frac{\partial R_i}{\partial c_i} = T_i \frac{\partial P_i}{\partial c_i} + P_i \frac{\partial T_i}{\partial c_i} = 0.$$

Our numerical algorithm to find c_i such that $\partial R_i/\partial c_i = 0$ is given in Algorithm 3 with $[c_i', c_i''] = [1.0, 15.0]$, which is similar to Algorithm 1. The time complexity of Algorithm 3 is $O(\log(I/\epsilon))$.

Algorithm 2: Find s_i

Input: $\lambda_j, \overline{r_j}, \overline{r_j^2}, \overline{d_j}, \overline{d_j^2}, c_j$, for all $1 \le j \ne i \le n$, m, s, and $\tilde{\lambda}_i, \lambda_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, c_i, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$.
Output: s_i, such that $\partial R_i/\partial s_i = 0$.

```
Initialize the search interval of $s_i$;                              (1)
while (the length of the search interval is $\ge \epsilon$) do        (2)
$s_i \leftarrow$ the middle point of the search interval;             (3)
Calculate $\partial R_i/\partial s_i$;                                (4)
if ($\partial R_i/\partial s_i < 0$) then                             (5)
    Change the search interval to the right half;                     (6)
else                                                                  (7)
    Change the search interval to the left half;                      (8)
end if                                                                (9)
end do;                                                              (10)
$s_i \leftarrow$ the middle point of the search interval;            (11)
return $s_i$.                                                        (12)
```

Algorithm 3: Find c_i

Input: $\lambda_j, \overline{r_j}, \overline{r_j^2}, \overline{d_j}, \overline{d_j^2}, c_j$, for all $1 \le j \ne i \le n$, m, s, and $\tilde{\lambda}_i, \lambda_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, s_i, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$.
Output: c_i, such that $\partial R_i/\partial c_i = 0$.

```
Initialize the search interval of $c_i$;                              (1)
while (the length of the search interval is $\ge \epsilon$) do        (2)
$c_i \leftarrow$ the middle point of the search interval;             (3)
Calculate $\partial R_i/\partial c_i$;                                (4)
if ($\partial R_i/\partial c_i < 0$) then                             (5)
    Change the search interval to the right half;                     (6)
else                                                                  (7)
    Change the search interval to the left half;                      (8)
end if                                                                (9)
end do;                                                              (10)
$c_i \leftarrow$ the middle point of the search interval;            (11)
return $c_i$.                                                        (12)
```

4) GAME(λ_i, s_i)

In this game, UE$_i$ needs to find λ_i and s_i such that

$$\frac{\partial R_i}{\partial \lambda_i} = T_i \frac{\partial P_i}{\partial \lambda_i} + P_i \frac{\partial T_i}{\partial \lambda_i} = 0,$$

and

$$\frac{\partial R_i}{\partial s_i} = T_i \frac{\partial P_i}{\partial s_i} + P_i \frac{\partial T_i}{\partial s_i} = 0.$$

Solving these two sophisticated nonlinear equations simultaneously needs special insight. For a fixed s_i, let

$$\hat{R}_i(s_i) = \min_{\lambda_i}(R_i(\lambda_i, s_i)).$$

Then, it suffices to minimize $\hat{R}_i(s_i)$. It can be shown that $\hat{R}_i(s_i)$ is a convex function of s_i, that is, $\partial \hat{R}_i/\partial s_i$ is an increasing function of s_i. Unfortunately, $\hat{R}_i(s_i)$ is analytically not available. Hence, we use a numerical approximation:

$$\frac{\partial \hat{R}_i}{\partial s_i} = \frac{\hat{R}_i(s_i + \Delta) - \hat{R}_i(s_i)}{\Delta},$$

where Δ is a sufficiently small quantity. (We set $\Delta = 10^{-5}$ in this paper.)

Our numerical algorithm to find λ_i and s_i such that $\partial R_i/\partial \lambda_i = 0$ and $\partial R_i/\partial s_i = 0$ is given in Algorithm 4 with $[s_i', s_i''] = [1.0, 3.4]$. The algorithm uses the classical bisection method (lines 2–12) to find s_i such that $\partial \hat{R}_i/\partial s_i = 0$, and Algorithm 1 to find λ_i (lines 4, 5, 14). In fact, in lines 4–5, we use Algorithm 1 to find λ_i, and then calculate $\hat{R}_i(s_i)$ and $\hat{R}_i(s_i + \Delta)$. The resulting s_i minimizes $\hat{R}_i(s_i)$, and also minimizes $R_i(\lambda_i, s_i)$, thus guaranteeing $\partial R_i/\partial \lambda_i = 0$ and $\partial R_i/\partial s_i = 0$. Due to the use of Algorithm 1 as a sub-algorithm, the time complexity of Algorithm 4 is $O((\log(I/\epsilon))^2)$.

Algorithm 4: Find λ_i and s_i

Input: $\lambda_j, \overline{r_j}, \overline{r_j^2}, \overline{d_j}, \overline{d_j^2}, c_j$, for all $1 \le j \ne i \le n$, m, s, and $\tilde{\lambda}_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, c_i, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$.
Output: λ_i and s_i such that $\partial R_i/\partial \lambda_i = 0$ and $\partial R_i/\partial s_i = 0$.

```
Initialize the search interval of $s_i$;                              (1)
while (the length of the search interval is $\ge \epsilon$) do        (2)
$s_i \leftarrow$ the middle point of the search interval;             (3)
Calculate $\hat{R}_i(s_i)$ using Algorithm 1;                         (4)
Calculate $\hat{R}_i(s_i + \Delta)$ using Algorithm 1;                (5)
Calculate $\partial \hat{R}_i/\partial s_i$;                          (6)
if ($\partial \hat{R}_i/\partial s_i < 0$) then                       (7)
    Change the search interval to the right half;                     (8)
else                                                                  (9)
    Change the search interval to the left half;                     (10)
end if                                                               (11)
end do;                                                              (12)
$s_i \leftarrow$ the middle point of the search interval;            (13)
Find $\lambda_i$ using Algorithm 1;                                  (14)
return $\lambda_i$ and $s_i$.                                        (15)
```

5) GAME(λ_i, c_i)

In this game, UE$_i$ needs to find λ_i and c_i such that

$$\frac{\partial R_i}{\partial \lambda_i} = T_i \frac{\partial P_i}{\partial \lambda_i} + P_i \frac{\partial T_i}{\partial \lambda_i} = 0,$$

and

$$\frac{\partial R_i}{\partial c_i} = T_i \frac{\partial P_i}{\partial c_i} + P_i \frac{\partial T_i}{\partial c_i} = 0.$$

For a fixed c_i, let

$$\hat{R}_i(c_i) = \min_{\lambda_i}(R_i(\lambda_i, c_i)).$$

Then, it can be shown that $\hat{R}_i(c_i)$ is a convex function of c_i, that is, $\partial\hat{R}_i/\partial c_i$ is an increasing function of c_i. Our numerical algorithm to find λ_i and c_i such that $\partial R_i/\partial\lambda_i = 0$ and $\partial R_i/\partial c_i = 0$ is given in Algorithm 5 with $[c'_i, c''_i] = [1.0, 15.0]$, which is similar to Algorithm 4. The time complexity of Algorithm 5 is $O((\log(I/\epsilon))^2)$.

Algorithm 5: Find λ_i and c_i

Input: $\lambda_j, \overline{r_j}, \overline{r_j^2}, \overline{d_j}, \overline{d_j^2}, c_j$, for all $1 \le j \ne i \le n$, m, s, and $\tilde{\lambda}_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, s_i, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$.
Output: λ_i and c_i such that $\partial R_i/\partial\lambda_i = 0$ and $\partial R_i/\partial c_i = 0$.

```
Initialize the search interval of $c_i$;                                (1)
while (the length of the search interval is $\ge \epsilon$) do          (2)
$c_i \leftarrow$ the middle point of the search interval;               (3)
Calculate $\hat{R}_i(c_i)$ using Algorithm 1;                           (4)
Calculate $\hat{R}_i(c_i + \Delta)$ using Algorithm 1;                  (5)
Calculate $\partial\hat{R}_i/\partial c_i$;                             (6)
if ($\partial\hat{R}_i/\partial c_i < 0$) then                          (7)
    Change the search interval to the right half;                       (8)
else                                                                    (9)
    Change the search interval to the left half;                        (10)
end if                                                                  (11)
end do;                                                                 (12)
$c_i \leftarrow$ the middle point of the search interval;               (13)
Find $\lambda_i$ using Algorithm 1;                                     (14)
return $\lambda_i$ and $c_i$.                                           (15)
```

6) GAME(s_i, c_i)

In this game, UE$_i$ needs to find s_i and c_i such that

$$\frac{\partial R_i}{\partial s_i} = T_i\frac{\partial P_i}{\partial s_i} + P_i\frac{\partial T_i}{\partial s_i} = 0,$$

and

$$\frac{\partial R_i}{\partial c_i} = T_i\frac{\partial P_i}{\partial c_i} + P_i\frac{\partial T_i}{\partial c_i} = 0.$$

For a fixed c_i, let

$$\hat{R}_i(c_i) = \min_{s_i}(R_i(s_i, c_i)).$$

Then, it can be shown that $\hat{R}_i(c_i)$ is a convex function of c_i, that is, $\partial\hat{R}_i/\partial c_i$ is an increasing function of c_i. Our numerical algorithm to find s_i and c_i such that $\partial R_i/\partial s_i = 0$ and $\partial R_i/\partial c_i = 0$ is given in Algorithm 6 with $[c'_i, c''_i] = [1.0, 15.0]$, which is similar to Algorithm 4. However, Algorithm 6 calls Algorithm 2 instead of Algorithm 1. The time complexity of Algorithm 6 is $O((\log(I/\epsilon))^2)$.

Algorithm 6: Find s_i and c_i

Input: $\lambda_j, \overline{r_j}, \overline{r_j^2}, \overline{d_j}, \overline{d_j^2}, c_j$, for all $1 \le j \ne i \le n$, m, s, and $\tilde{\lambda}_i, \lambda_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$.
Output: s_i and c_i such that $\partial R_i/\partial s_i = 0$ and $\partial R_i/\partial c_i = 0$.

```
Initialize the search interval of $c_i$;                                (1)
while (the length of the search interval is $\ge \epsilon$) do          (2)
$c_i \leftarrow$ the middle point of the search interval;               (3)
Calculate $\hat{R}_i(c_i)$ using Algorithm 2;                           (4)
Calculate $\hat{R}_i(c_i + \Delta)$ using Algorithm 2;                  (5)
Calculate $\partial\hat{R}_i/\partial c_i$;                             (6)
if ($\partial\hat{R}_i/\partial c_i < 0$) then                          (7)
    Change the search interval to the right half;                       (8)
else                                                                    (9)
    Change the search interval to the left half;                        (10)
end if                                                                  (11)
end do;                                                                 (12)
$c_i \leftarrow$ the middle point of the search interval;               (13)
Find $s_i$ using Algorithm 2;                                           (14)
return $s_i$ and $c_i$.                                                 (15)
```

7) GAME(λ_i, s_i, c_i)

In this game, UE$_i$ needs to find λ_i, s_i, and c_i such that

$$\frac{\partial R_i}{\partial \lambda_i} = T_i\frac{\partial P_i}{\partial \lambda_i} + P_i\frac{\partial T_i}{\partial \lambda_i} = 0,$$
$$\frac{\partial R_i}{\partial s_i} = T_i\frac{\partial P_i}{\partial s_i} + P_i\frac{\partial T_i}{\partial s_i} = 0,$$

and

$$\frac{\partial R_i}{\partial c_i} = T_i\frac{\partial P_i}{\partial c_i} + P_i\frac{\partial T_i}{\partial c_i} = 0.$$

For a fixed c_i, let

$$\hat{R}_i(c_i) = \min_{\lambda_i, s_i}(R_i(\lambda_i, s_i, c_i)).$$

Then, it can be shown that $\hat{R}_i(c_i)$ is a convex function of c_i, that is, $\partial\hat{R}_i/\partial c_i$ is an increasing function of c_i. Our numerical algorithm to find λ_i, s_i, and c_i such that $\partial R_i/\partial\lambda_i = 0$, $\partial R_i/\partial s_i = 0$, and $\partial R_i/\partial c_i = 0$ is given in Algorithm 7 with $[c'_i, c''_i] = [1.0, 15.0]$, which is similar to Algorithm 6. However, Algorithm 7 calls Algorithm 4 instead of Algorithm 2. Due to the use of Algorithm 4 as a sub-algorithm, the time complexity of Algorithm 7 is $O((\log(I/\epsilon))^3)$.

B. THE BEST RESPONSE OF AN MEC

In this section, we develop an algorithm to find the best response of the mobile edge cloud. The MEC needs to find s such that

$$\frac{\partial R}{\partial s} = T\frac{\partial P}{\partial s} + P\frac{\partial T}{\partial s} = 0.$$

Our numerical algorithm to find s such that $\partial R/\partial s = 0$ is given in Algorithm 8 with $s'' = 5.0$, which is similar to Algorithm 1. The time complexity of Algorithm 8 is $O(\log(I/\epsilon))$.

Algorithm 7: Find λ_i, s_i, and c_i

Input: $\lambda_j, \overline{r_j}, \overline{r_j^2}, \overline{d_j}, \overline{d_j^2}, c_j$, for all $1 \le j \ne i \le n$, m, s, and $\tilde{\lambda}_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$.
Output: λ_i, s_i, and c_i such that $\partial R_i/\partial\lambda_i = 0$, $\partial R_i/\partial s_i = 0$, and $\partial R_i/\partial c_i = 0$.

```
Initialize the search interval of $c_i$;                               (1)
while (the length of the search interval is $\ge \epsilon$) do         (2)
$c_i \leftarrow$ the middle point of the search interval;              (3)
Calculate $\hat{R}_i(c_i)$ using Algorithm 4;                          (4)
Calculate $\hat{R}_i(c_i + \Delta)$ using Algorithm 4;                 (5)
Calculate $\partial\hat{R}_i/\partial c_i$;                            (6)
if ($\partial\hat{R}_i/\partial c_i < 0$) then                         (7)
    Change the search interval to the right half;                      (8)
else                                                                   (9)
    Change the search interval to the left half;                      (10)
end if                                                                (11)
end do;                                                               (12)
$c_i \leftarrow$ the middle point of the search interval;             (13)
Find $\lambda_i$ and $s_i$ using Algorithm 4;                         (14)
return $\lambda_i$, $s_i$, and $c_i$.                                 (15)
```

Algorithm 8: Find s

Input: $\lambda_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, c_i$, for all $1 \le i \le n$, and m, ξ, α, P_s.
Output: s, such that $\partial R/\partial s = 0$.

```
Initialize the search interval of $s$;                                 (1)
while (the length of the search interval is $\ge \epsilon$) do         (2)
$s \leftarrow$ the middle point of the search interval;                (3)
Calculate $\partial R/\partial s$;                                     (4)
if ($\partial R/\partial s < 0$) then                                  (5)
    Change the search interval to the right half;                      (6)
else                                                                   (7)
    Change the search interval to the left half;                       (8)
end if                                                                 (9)
end do;                                                               (10)
$s \leftarrow$ the middle point of the search interval;               (11)
return $s$.                                                           (12)
```

C. AN ITERATIVE ALGORITHM FOR NASH EQUILIBRIUM

In this section, we develop an iterative algorithm to find the Nash equilibrium.

Algorithm 9 runs in rounds (lines 2–14). In each round, every mobile user finds his best response to the current situation by using Algorithms 1–7 (lines 3–5). The mobile edge server also finds its best response to the current situation by using Algorithm 8 (line 6). The algorithm terminates when the action profiles of two successive rounds are close enough (lines 8–13). The final converged action profile $\mathbf{x}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_n^*, \mathbf{x}_{n+1}^*)$ is returned as the Nash equilibrium, i.e., a strategy profile with the property that no player can benefit from a unilateral deviation from $\mathbf{x}_i^*$, if all the other players act according to it.

Algorithm 9: Calculate the Nash Equilibrium

Input: $\tilde{\lambda}_i, \lambda_i, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, s_i, c_i, \xi_i, \alpha_i, P_{s,i}, w_i, \beta_i$, for all $1 \le i \le n$, and m, s, ξ, α, P_s.
Output: The Nash equilibrium $\mathbf{x}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_n^*, s^*)$.

```
Initialize $\mathbf{x}$;                                               (1)
repeat                                                                 (2)
for $i \leftarrow 1$ to $n$ do                                         (3)
    Obtain $\mathbf{x}_i'$ by using Algorithms 1–7;                    (4)
end do;                                                                (5)
Obtain $s'$ by using Algorithm 8;                                      (6)
$\mathbf{x}' \leftarrow (\mathbf{x}_1', \mathbf{x}_2', \ldots, \mathbf{x}_n', s')$;   (7)
if ($\|\mathbf{x}' - \mathbf{x}\| \ge \delta$) then                    (8)
    $\mathbf{x} \leftarrow \mathbf{x}'$;                               (9)
else                                                                  (10)
    $\mathbf{x}^* \leftarrow \mathbf{x}'$;                            (11)
    return $\mathbf{x}^*$;                                            (12)
end if                                                                (13)
forever.                                                              (14)
```

The termination detection condition in line 8 is

$$\|\mathbf{x}' - \mathbf{x}\| = \sqrt{\sum_{i=1}^{n}(|\lambda_i' - \lambda_i|^2 + |s_i' - s_i|^2 + |c_i' - c_i|^2) + |s' - s|^2} < \delta.$$

Since Algorithm j ($1 \le j \le 7$) is invoked n times in each round, the time complexity of each round is $O(n(\log(I/\epsilon))^3)$, and the overall time complexity of Algorithm 9 is $O(Nn(\log(I/\epsilon))^3)$, where N is the number of rounds, which is mainly determined by the accuracy requirement δ in line 8. (We set $\delta = 10^{-5}$ in this paper.)

VII. NUMERICAL EXAMPLES AND PERFORMANCE DATA

In this section, we present numerical examples and performance data.

A. NUMERICAL EXAMPLES

In this section, we demonstrate numerical examples of our algorithms.

Throughout this section, we consider a mobile edge computing environment with $n = 10$ UEs and a single MEC. The parameters of UE$_i$ are set as follows: $\tilde{\lambda}_i = 1.5 + 0.05(i-1)$ tasks/second, $\lambda_i = 0.41\tilde{\lambda}_i$ tasks/second, $\overline{r_i} = 1.5 + 0.05(i-1)$

TABLE 2. **Parameters of a mobile edge computing environment (idle-speed model).**

	UE$_1$	UE$_2$	UE$_3$	UE$_4$	UE$_5$	UE$_6$	UE$_7$	UE$_8$	UE$_9$	UE$_{10}$
$\tilde{\lambda}_i$	1.50000	1.55000	1.60000	1.65000	1.70000	1.75000	1.80000	1.85000	1.90000	1.95000
λ_i	0.61500	0.63550	0.65600	0.67650	0.69700	0.71750	0.73800	0.75850	0.77900	0.79950
$\overline{r_i}$	1.50000	1.55000	1.60000	1.65000	1.70000	1.75000	1.80000	1.85000	1.90000	1.95000
$\overline{r_i^2}$	2.92500	3.12325	3.32800	3.53925	3.75700	3.98125	4.21200	4.44925	4.69300	4.94325
$\overline{d_i}$	1.00000	1.10000	1.20000	1.30000	1.40000	1.50000	1.60000	1.70000	1.80000	1.90000
$\overline{d_i^2}$	1.50000	1.81500	2.16000	2.53500	2.94000	3.37500	3.84000	4.33500	4.86000	5.41500
s_i	1.50000	1.60000	1.70000	1.80000	1.90000	2.00000	2.10000	2.20000	2.30000	2.40000
c_i	2.00000	2.25000	2.50000	2.75000	3.00000	3.25000	3.50000	3.75000	4.00000	4.25000
ξ_i	1.00000	0.95000	0.90000	0.85000	0.80000	0.75000	0.70000	0.65000	0.60000	0.55000
α_i	2.00000	2.00000	2.00000	2.00000	2.00000	2.00000	2.00000	2.00000	2.00000	2.00000
$P_{s,i}$	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
w_i	3.00000	3.10000	3.20000	3.30000	3.40000	3.50000	3.60000	3.70000	3.80000	3.90000
β_i	2.00000	1.90000	1.80000	1.70000	1.60000	1.50000	1.40000	1.30000	1.20000	1.10000
ρ_i	0.88500	0.88592	0.88847	0.89237	0.89742	0.90344	0.91029	0.91785	0.92604	0.93478
$\overline{x_i}$	1.00000	0.96875	0.94118	0.91667	0.89474	0.87500	0.85714	0.84091	0.82609	0.81250
W_i	5.00217	4.89010	4.87347	4.94036	5.08800	5.32122	5.65305	6.10716	6.72348	7.56962
$T_{\text{local},i}$	6.00217	5.85885	5.81464	5.85703	5.98273	6.19622	6.51019	6.94807	7.54957	8.38212
$\overline{r_i}/s$	0.42857	0.44286	0.45714	0.47143	0.48571	0.50000	0.51429	0.52857	0.54286	0.55714
$\overline{d_i}/c_i$	0.50000	0.48889	0.48000	0.47273	0.46667	0.46154	0.45714	0.45333	0.45000	0.44706
$T_{\text{remote},i}$	3.69505	3.69823	3.70362	3.71064	3.71886	3.72802	3.73791	3.74839	3.75934	3.77068
T_i	5.05625	4.97300	4.94913	4.97701	5.05454	5.18426	5.37356	5.63620	5.99557	6.49143
$P_{\text{comp},i}$	2.99125	3.15456	3.31091	3.45760	3.59175	3.71031	3.81005	3.88756	3.93926	3.96139
$P_{t,i}$	0.29370	0.34412	0.39923	0.45988	0.52711	0.60226	0.68704	0.78371	0.89526	1.02578
$P_{\text{comm},i}$	0.09031	0.10691	0.12571	0.14707	0.17145	0.19944	0.23179	0.26948	0.31383	0.36664
P_i	3.08156	3.26148	3.43662	3.60467	3.76320	3.90975	4.04184	4.15704	4.25309	4.32803
R_i	15.58116	16.21931	17.00828	17.94048	19.02128	20.26916	21.71905	23.42994	25.49974	28.09509
$s = 3.50000$, $\rho = 0.97263$, $W = 2.76648$, $T = 3.72914$, $P = 52.20133$, $R = 194.66585$										

BI, $\overline{r_i^2} = 1.3\overline{r_i}^2$ BI^2, $\overline{d_i} = 1.0 + 0.1(i-1)$ MB, $\overline{d_i^2} = 1.5\overline{d_i}^2$ MB^2, $s_i = 1.5 + 0.1(i-1)$ BI/second, $c_i = 2.0 + 0.25(i-1)$ MB/second, $\xi_i = 1.0 - 0.05(i-1)$, $\alpha_i = 2.0$, $P_{s,i} = 1.0$ Watts, $w_i = 3.0 + 0.1(i-1)$ MB/second, $\beta_i = 2.0 - 0.1(i-1)$ Watts^{-1}, for all $1 \le i \le n$. The parameters of the MEC are set as follows: $m = 7$, $s = 3.5$ BI/second, $\xi = 0.5$, $\alpha = 2.0$, $P_s = 1.5$ Watts. Table 2 shows the above parameters of a mobile edge computing environment, and the resulting performance data for the idle-speed model. The data for the constant-speed model are similar, with slightly increased $P_{\text{comp},i}$, P_i, and R_i.

In the following, we illustrate the best responses of the players by using Algorithm 1–8. Let us consider UE$_5$.

Using Algorithm 1, UE$_5$ sets $\lambda_5 = 0.75861$, which results in $T_5 = 4.46716$, $P_5 = 3.61915$, and $R_5 = 16.16733$. It is clear that compared with the original parameter setting in Table 2, UE$_5$ increases the amount of computation offloading, which gives rise to reduced average response time, reduced average power consumption, and reduced cost-performance ratio.

Using Algorithm 2, UE$_5$ sets $s_5 = 2.76593$, which results in $T_5 = 2.26622$, $P_5 = 4.94440$, and $R_5 = 11.20509$. It is clear that compared with the original parameter setting in Table 2, UE$_5$ increases the server execution speed, which gives rise to significantly reduced average response time, increased average power consumption, and significantly reduced cost-performance ratio.

Using Algorithm 3, UE$_5$ sets $c_5 = 7.28516$, which results in $T_5 = 4.31685$, $P_5 = 3.87771$, and $R_5 = 16.73950$. It is clear that compared with the original parameter setting in Table 2, UE$_5$ increases the data communication speed, which gives rise to reduced average response time, slightly increased average power consumption, and reduced cost-performance ratio.

Using Algorithm 4, UE$_5$ sets $\lambda_5 = 0.47826$ and $s_5 = 3.40000$, which result in $T_5 = 1.33269$, $P_5 = 6.76697$, and $R_5 = 9.01830$. It is clear that compared with the original parameter setting in Table 2, UE$_5$ reduces the amount of computation offloading and increases the server execution speed, which gives rise to significantly reduced average response time, increased average power consumption, and significantly reduced cost-performance ratio.

Using Algorithm 5, UE$_5$ sets $\lambda_5 = 0.96903$ and $c_5 = 11.06776$, which result in $T_5 = 2.28860$, $P_5 = 3.54369$, and $R_5 = 8.11010$. It is clear that compared with the original parameter setting in Table 2, UE$_5$ increases the amount of

TABLE 3. Numerical demonstration of positiveness of leading principal minors (idle-speed model).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
M_{λ_i}	2097.67	2081.78	2150.82	2301.26	2543.92	2905.50	3436.32	4228.08	5453.96	7464.57
M_{s_i}	639.57	589.76	572.52	581.56	616.07	679.90	782.51	942.37	1195.05	1612.74
M_{c_i}	8.52	7.23	6.29	5.58	5.03	4.58	4.20	3.88	3.60	3.35
M_{λ_i,s_i}	121488.18	118201.43	121503.89	130900.09	147103.04	172078.86	209599.91	266597.02	356237.69	505378.39
M_{λ_i,c_i}	16383.90	13720.50	12313.85	11704.68	11697.51	12243.30	13407.29	15390.30	18617.24	23969.20
M_{s_i,c_i}	5444.72	4260.36	3600.67	3245.85	3095.95	3110.13	3285.08	3652.83	4295.66	5392.12
M_{λ_i,s_i,c_i}	225004.17	187926.43	170506.14	165599.71	170547.38	185238.69	211762.50	255164.67	325855.07	445902.49
$M_s = 8756.91$										

TABLE 4. Numerical demonstration of positiveness of leading principal minors (constant-speed model).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
M_{λ_i}	2154.57	2141.61	2213.07	2365.35	2609.18	2971.23	3501.76	4292.45	5516.46	7524.41
M_{s_i}	642.60	592.64	575.25	584.14	618.51	682.20	784.65	944.37	1196.90	1614.44
M_{c_i}	9.23	7.84	6.83	6.05	5.43	4.92	4.49	4.12	3.80	3.51
M_{λ_i,s_i}	130209.07	126845.26	130369.87	140252.74	157212.16	183254.33	222232.87	281221.54	373637.42	526793.07
M_{λ_i,c_i}	17920.36	15031.14	13482.97	12785.09	12725.91	13248.33	14414.63	16426.74	19715.38	25174.69
M_{s_i,c_i}	5921.12	4638.15	3917.20	3523.23	3348.49	3348.02	3516.48	3885.20	4537.08	5652.89
M_{λ_i,s_i,c_i}	261279.82	218448.38	197891.65	191447.36	195979.85	211178.41	239108.67	284954.07	359475.84	485472.14
$M_s = 8758.81$										

TABLE 5. Performance data of game(λ_i) (idle-speed model, $N = 48$).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
λ_i	0.74334	0.76221	0.78216	0.80317	0.82521	0.84822	0.87215	0.89696	0.92256	0.94889
ρ_i	0.75666	0.76318	0.76973	0.77626	0.78271	0.78906	0.79530	0.80142	0.80745	0.81340
$\overline{x_i}$	1.00000	0.96875	0.94118	0.91667	0.89474	0.87500	0.85714	0.84091	0.82609	0.81250
W_i	2.02115	2.02920	2.04502	2.06724	2.09493	2.12750	2.16456	2.20596	2.25175	2.30218
$T_{\mathrm{local},i}$	3.02115	2.99795	2.98620	2.98391	2.98967	3.00250	3.02170	3.04687	3.07783	3.11468
$\overline{r_i}/s$	0.30000	0.31000	0.32000	0.33000	0.34000	0.35000	0.36000	0.37000	0.38000	0.39000
$\overline{d_i}/c_i$	0.50000	0.48889	0.48000	0.47273	0.46667	0.46154	0.45714	0.45333	0.45000	0.44706
$T_{\mathrm{remote},i}$	3.67652	3.67541	3.67652	3.67925	3.68319	3.68806	3.69366	3.69985	3.70652	3.71358
T_i	3.34592	3.33109	3.32366	3.32238	3.32631	3.33479	3.34729	3.36346	3.38310	3.40611
$P_{\mathrm{comp},i}$	2.70248	2.85604	3.00208	3.13782	3.26047	3.36718	3.45508	3.52128	3.56286	3.57686
$P_{t,i}$	0.29370	0.34412	0.39923	0.45988	0.52711	0.60226	0.68704	0.78371	0.89526	1.02578
$P_{\mathrm{comm},i}$	0.10916	0.12823	0.14989	0.17461	0.20299	0.23578	0.27392	0.31867	0.37167	0.43515
P_i	2.81164	2.98427	3.15197	3.31243	3.46346	3.60295	3.72900	3.83995	3.93452	4.01201
R_i	9.40754	9.94087	10.47606	11.00514	11.52054	12.01507	12.48204	12.91553	13.31086	13.66533
$s = 5.00000$, $\rho = 0.97703$, $W = 2.87652$, $T = 3.69024$, $P = 95.99045$, $R = 354.22813$										

computation offloading and increases the data communication speed, which gives rise to significantly reduced average response time, reduced average power consumption, and significantly reduced cost-performance ratio.

Using Algorithm 6, UE_5 sets $s_5 = 3.37633$ and $c_5 = 11.51681$, which result in $T_5 = 1.18285$, $P_5 = 6.10672$, and $R_5 = 7.22335$. It is clear that compared with the original parameter setting in Table 2, UE_5 increases the server

TABLE 6. Performance data of game(λ_i) (constant-speed model, $N = 35$).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
λ_i	0.73726	0.75635	0.77673	0.79835	0.82113	0.84501	0.86992	0.89579	0.92254	0.95012
ρ_i	0.76274	0.76885	0.77484	0.78068	0.78635	0.79186	0.79721	0.80240	0.80746	0.81240
$\overline{x_i}$	1.00000	0.96875	0.94118	0.91667	0.89474	0.87500	0.85714	0.84091	0.82609	0.81250
W_i	2.08960	2.09445	2.10525	2.12088	2.14057	2.16381	2.19024	2.21963	2.25189	2.28707
$T_{\text{local},i}$	3.08960	3.06320	3.04642	3.03754	3.03531	3.03881	3.04738	3.06053	3.07798	3.09957
$\overline{r_i}/s$	0.30000	0.31000	0.32000	0.33000	0.34000	0.35000	0.36000	0.37000	0.38000	0.39000
$\overline{d_i}/c_i$	0.50000	0.48889	0.48000	0.47273	0.46667	0.46154	0.45714	0.45333	0.45000	0.44706
$T_{\text{remote},i}$	3.25599	3.25488	3.25599	3.25872	3.26266	3.26753	3.27313	3.27933	3.28599	3.29305
T_i	3.17138	3.15674	3.14816	3.14456	3.14512	3.14925	3.15648	3.16648	3.17898	3.19384
$P_{\text{comp},i}$	3.25000	3.43200	3.60100	3.75400	3.88800	4.00000	4.08700	4.14600	4.17400	4.16800
$P_{t,i}$	0.29370	0.34412	0.39923	0.45988	0.52711	0.60226	0.68704	0.78371	0.89526	1.02578
$P_{\text{comm},i}$	0.10827	0.12724	0.14885	0.17356	0.20199	0.23489	0.27322	0.31826	0.37166	0.43571
P_i	3.35827	3.55924	3.74985	3.92756	4.08999	4.23489	4.36022	4.46426	4.54566	4.60371
R_i	10.65034	11.23559	11.80512	12.35044	12.86352	13.33673	13.76297	14.13596	14.45056	14.70352
$s = 5.00000$, $\rho = 0.97341$, $W = 2.45599$, $T = 3.26976$, $P = 98.00000$, $R = 320.43623$										

TABLE 7. Performance data of game(s_i) (idle-speed model, $N = 2$).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
s_i	3.05143	3.20336	3.35992	3.52161	3.68887	3.86217	4.04210	4.22941	4.42515	4.63077
ρ_i	0.43504	0.44250	0.44953	0.45612	0.46223	0.46784	0.47292	0.47744	0.48132	0.48447
$\overline{x_i}$	0.49157	0.48387	0.47620	0.46854	0.46085	0.45311	0.44531	0.43741	0.42936	0.42110
W_i	0.24605	0.24963	0.25278	0.25541	0.25747	0.25893	0.25971	0.25976	0.25898	0.25722
$T_{\text{local},i}$	0.73762	0.73350	0.72898	0.72394	0.71832	0.71204	0.70503	0.69718	0.68835	0.67832
$\overline{r_i}/s$	0.31676	0.32732	0.33788	0.34844	0.35900	0.36955	0.38011	0.39067	0.40123	0.41179
$\overline{d_i}/c_i$	0.50000	0.48889	0.48000	0.47273	0.46667	0.46154	0.45714	0.45333	0.45000	0.44706
$T_{\text{remote},i}$	1.08111	1.08055	1.08222	1.08551	1.09001	1.09544	1.10160	1.10835	1.11558	1.12319
T_i	0.87845	0.87579	0.87381	0.87218	0.87071	0.86923	0.86762	0.86576	0.86351	0.86072
$P_{\text{comp},i}$	5.05077	5.31364	5.56735	5.80818	6.03191	6.23384	6.40881	6.55123	6.65508	6.71396
$P_{t,i}$	0.29370	0.34412	0.39923	0.45988	0.52711	0.60226	0.68704	0.78371	0.89526	1.02578
$P_{\text{comm},i}$	0.09031	0.10691	0.12571	0.14707	0.17145	0.19944	0.23179	0.26948	0.31383	0.36664
P_i	5.14108	5.42056	5.69306	5.95525	6.20336	6.43328	6.64060	6.82071	6.96891	7.08060
R_i	4.51618	4.74728	4.97464	5.19407	5.40133	5.59202	5.76153	5.90509	6.01772	6.09439
$s = 4.73544$, $\rho = 0.84181$, $W = 0.26435$, $T = 1.09753$, $P = 76.56974$, $R = 84.03735$										

execution speed and increases the data communication speed, which gives rise to significantly reduced average response time, increased average power consumption, and significantly reduced cost-performance ratio.

Using Algorithm 7, UE_5 sets $\lambda_5 = 0.67529$, $s_5 = 3.40000$, and $c_5 = 11.45013$, which result in $T_5 = 1.16043$, $P_5 = 6.21933$, and $R_5 = 7.21709$. It is clear that compared with the original parameter setting in Table 2, UE_5 reduces the amount of computation offloading, increases the server execution speed, and increases the data communication speed, which gives rise to significantly reduced average response time, increased average power consumption, and significantly reduced cost-performance ratio.

TABLE 8. Performance data of game(s_i) (constant-speed model, $N = 2$).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
s_i	2.08718	2.21118	2.33922	2.47144	2.60798	2.74903	2.89479	3.04559	3.20183	3.36412
ρ_i	0.63602	0.64105	0.64569	0.64994	0.65380	0.65728	0.66036	0.66302	0.66521	0.66688
$\overline{x_i}$	0.71867	0.70098	0.68399	0.66763	0.65185	0.63659	0.62181	0.60744	0.59341	0.57965
W_i	0.81629	0.81372	0.81020	0.80569	0.80016	0.79356	0.78583	0.77683	0.76641	0.75428
$T_{\text{local},i}$	1.53497	1.51470	1.49419	1.47332	1.45201	1.43015	1.40763	1.38427	1.35982	1.33392
$\overline{r_i}/s$	0.33754	0.34879	0.36004	0.37129	0.38254	0.39379	0.40504	0.41630	0.42755	0.43880
$\overline{d_i}/c_i$	0.50000	0.48889	0.48000	0.47273	0.46667	0.46154	0.45714	0.45333	0.45000	0.44706
$T_{\text{remote},i}$	1.18953	1.18967	1.19203	1.19601	1.20120	1.20732	1.21418	1.22162	1.22954	1.23785
T_i	1.39334	1.38144	1.37031	1.35962	1.34917	1.33879	1.32832	1.31758	1.30640	1.29453
$P_{\text{comp},i}$	5.35633	5.64486	5.92476	6.19181	6.44125	6.66786	6.86588	7.02915	7.15103	7.22451
$P_{t,i}$	0.29370	0.34412	0.39923	0.45988	0.52711	0.60226	0.68704	0.78371	0.89526	1.02578
$P_{\text{comm},i}$	0.09031	0.10691	0.12571	0.14707	0.17145	0.19944	0.23179	0.26948	0.31383	0.36664
P_i	5.44664	5.75177	6.05047	6.33888	6.61271	6.86730	7.09767	7.29863	7.46487	7.59115
R_i	7.58901	7.94572	8.29099	8.61848	8.92170	9.19387	9.42795	9.61656	9.75213	9.82699
$s = 4.44396$, $\rho = 0.86612$, $W = 0.35199$, $T = 1.20923$, $P = 79.62058$, $R = 96.27967$										

TABLE 9. Performance data of game(c_i) (idle-speed model, $N = 6$).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
c_i	4.61136	4.75860	4.86689	4.93555	4.96311	4.94742	4.88574	4.77479	4.61064	4.38868
ρ_i	0.88500	0.88592	0.88847	0.89237	0.89742	0.90344	0.91029	0.91785	0.92604	0.93478
$\overline{x_i}$	1.00000	0.96875	0.94118	0.91667	0.89474	0.87500	0.85714	0.84091	0.82609	0.81250
W_i	5.00217	4.89010	4.87347	4.94036	5.08800	5.32122	5.65305	6.10716	6.72348	7.56962
$T_{\text{local},i}$	6.00217	5.85885	5.81464	5.85703	5.98273	6.19622	6.51019	6.94807	7.54957	8.38212
$\overline{r_i}/s$	0.38570	0.39856	0.41141	0.42427	0.43713	0.44998	0.46284	0.47570	0.48855	0.50141
$\overline{d_i}/c_i$	0.21686	0.23116	0.24656	0.26340	0.28208	0.30319	0.32748	0.35604	0.39040	0.43293
$T_{\text{remote},i}$	0.71850	0.74566	0.77392	0.80360	0.83515	0.86911	0.90626	0.94767	0.99489	1.05028
T_i	3.83587	3.76244	3.74795	3.78513	3.87222	4.01210	4.21258	4.48791	4.86215	5.37607
$P_{\text{comp},i}$	2.99125	3.15456	3.31091	3.45760	3.59175	3.71031	3.81005	3.88756	3.93926	3.96139
$P_{t,i}$	0.95107	0.99892	1.03873	1.07050	1.09412	1.10928	1.11556	1.11238	1.09893	1.07407
$P_{\text{comm},i}$	0.12684	0.14674	0.16801	0.19075	0.21511	0.24131	0.26961	0.30040	0.33421	0.37177
P_i	3.11809	3.30131	3.47892	3.64835	3.80687	3.95162	4.07966	4.18797	4.27347	4.33315
R_i	11.96058	12.42098	13.03881	13.80947	14.74103	15.85432	17.18591	18.79522	20.77826	23.29532
$s = 3.88904$, $\rho = 0.76502$, $W = 0.11594$, $T = 0.87312$, $P = 50.99714$, $R = 44.52639$										

Now, let us consider the MEC. The MEC uses Algorithm 8 to set $s = 4.73544$, which results in $T = 1.09753$, $P = 76.56974$, and $R = 84.03735$. It is clear that compared with the original parameter setting in Table 2, the MEC increases the server execution speed, which gives rise to significantly reduced average response time, increased average power consumption, and significantly reduced cost-performance ratio.

B. PERFORMANCE DATA

In this section, we demonstrate performance data of our games. For clarity of presentation, all performance data are moved to Appendix 4.

1) GAME(λ_i)

Tables 5 and 6 demonstrate performance data of Game(λ_i) for the idle-speed model and the constant-speed model

TABLE 10. **Performance data of game(c_i) (constant-speed model, $N = 7$).**

	UE$_1$	UE$_2$	UE$_3$	UE$_4$	UE$_5$	UE$_6$	UE$_7$	UE$_8$	UE$_9$	UE$_{10}$
c_i	4.84002	5.00016	5.11778	5.19228	5.22226	5.20570	5.14001	5.02201	4.84792	4.61321
ρ_i	0.88500	0.88592	0.88847	0.89237	0.89742	0.90344	0.91029	0.91785	0.92604	0.93478
$\overline{x_i}$	1.00000	0.96875	0.94118	0.91667	0.89474	0.87500	0.85714	0.84091	0.82609	0.81250
W_i	5.00217	4.89010	4.87347	4.94036	5.08800	5.32122	5.65305	6.10716	6.72348	7.56962
$T_{\text{local},i}$	6.00217	5.85885	5.81464	5.85703	5.98273	6.19622	6.51019	6.94807	7.54957	8.38212
$\overline{r_i}/s$	0.44086	0.45555	0.47025	0.48494	0.49964	0.51433	0.52903	0.54373	0.55842	0.57312
$\overline{d_i}/c_i$	0.20661	0.21999	0.23448	0.25037	0.26808	0.28815	0.31128	0.33851	0.37129	0.41186
$T_{\text{remote},i}$	0.84078	0.86885	0.89803	0.92862	0.96103	0.99579	1.03362	1.07554	1.12302	1.17828
T_i	3.88600	3.81295	3.79883	3.83638	3.92383	4.06404	4.26480	4.54034	4.91468	5.42855
$P_{\text{comp},i}$	3.25000	3.43200	3.60100	3.75400	3.88800	4.00000	4.08700	4.14600	4.17400	4.16800
$P_{t,i}$	1.02980	1.08357	1.12777	1.16240	1.18738	1.20248	1.20738	1.20158	1.18440	1.15481
$P_{\text{comm},i}$	0.13085	0.15149	0.17347	0.19688	0.22187	0.24861	0.27737	0.30852	0.34257	0.38026
P_i	3.38085	3.58349	3.77447	3.95088	4.10987	4.24861	4.36437	4.45452	4.51657	4.54826
R_i	13.13799	13.66368	14.33858	15.15711	16.12644	17.26651	18.61314	20.22501	22.19752	24.69044
$s = 3.40245$, $\rho = 0.81417$, $W = 0.19331$, $T = 0.99913$, $P = 51.01839$, $R = 50.97413$										

TABLE 11. **Performance data of game(λ_i, s_i) (idle-speed model, $N = 53$).**

	UE$_1$	UE$_2$	UE$_3$	UE$_4$	UE$_5$	UE$_6$	UE$_7$	UE$_8$	UE$_9$	UE$_{10}$
λ_i	1.30027	1.34836	0.43407	0.51082	0.58510	0.65675	0.72562	0.79160	0.85460	0.91456
s_i	1.00000	1.00000	3.40000	3.40000	3.40000	3.40000	3.40000	3.40000	3.40000	3.40000
ρ_i	0.29959	0.31254	0.54867	0.55284	0.55745	0.56270	0.56879	0.57589	0.58419	0.59386
$\overline{x_i}$	1.50000	1.55000	0.47059	0.48529	0.50000	0.51471	0.52941	0.54412	0.55882	0.57353
W_i	0.41705	0.45804	0.37186	0.38998	0.40938	0.43050	0.45391	0.48025	0.51033	0.54509
$T_{\text{local},i}$	1.91705	2.00804	0.84245	0.87528	0.90938	0.94521	0.98332	1.02437	1.06915	1.11862
$\overline{r_i}/s$	0.30000	0.31000	0.32000	0.33000	0.34000	0.35000	0.36000	0.37000	0.38000	0.39000
$\overline{d_i}/c_i$	0.50000	0.48889	0.48000	0.47273	0.46667	0.46154	0.45714	0.45333	0.45000	0.44706
$T_{\text{remote},i}$	1.83317	1.83206	1.83317	1.83590	1.83984	1.84471	1.85031	1.85651	1.86317	1.87023
T_i	1.84434	1.85495	1.11122	1.17268	1.22962	1.28278	1.33283	1.38044	1.42630	1.47113
$P_{\text{comp},i}$	1.29959	1.29691	6.70840	6.43217	6.15530	5.87865	5.60264	5.32725	5.05195	4.77574
$P_{t,i}$	0.29370	0.34412	0.39923	0.45988	0.52711	0.60226	0.68704	0.78371	0.89526	1.02578
$P_{\text{comm},i}$	0.19095	0.22684	0.08318	0.11105	0.14393	0.18255	0.22790	0.28124	0.34429	0.41940
P_i	1.49054	1.52376	6.79158	6.54322	6.29922	6.06120	5.83054	5.60849	5.39625	5.19514
R_i	2.74906	2.82650	7.54697	7.67308	7.74566	7.77517	7.77109	7.74216	7.69664	7.64273
$s = 5.00000$, $\rho = 0.94275$, $W = 1.03317$, $T = 1.84571$, $P = 92.99025$, $R = 171.63283$										

respectively. It is clear that in the stable situation, compared with the original parameter setting in Table 2, all UEs increase their amount of computation offloading and get reduced average response time, reduced average power consumption, and reduced cost-performance ratio. The MEC sets its server execution speed to the maximum available, and gets reduced average response time, increased average power consumption, and increased cost-performance ratio.

2) GAME(s_i)

Tables 7 and 8 demonstrate performance data of Game(s_i) for the idle-speed model and the constant-speed model respectively. It is clear that in the stable situation, compared with

TABLE 12. Performance data of game(λ_i, s_i) (constant-speed model, $N = 73$).

	UE$_1$	UE$_2$	UE$_3$	UE$_4$	UE$_5$	UE$_6$	UE$_7$	UE$_8$	UE$_9$	UE$_{10}$
λ_i	1.07438	1.13678	1.19821	1.22207	1.14788	0.36232	0.43357	0.50517	0.57685	0.64835
s_i	1.00000	1.00000	1.00000	1.07796	1.37144	3.40000	3.40000	3.40000	3.40000	3.40000
ρ_i	0.63843	0.64049	0.64286	0.65502	0.68439	0.71425	0.72340	0.73174	0.73941	0.74653
$\overline{x_i}$	1.50000	1.55000	1.60000	1.53067	1.23957	0.51471	0.52941	0.54412	0.55882	0.57353
W_i	1.72159	1.79495	1.87200	1.88911	1.74717	0.83625	0.90000	0.96475	1.03066	1.09798
$T_{\text{local},i}$	3.22159	3.34495	3.47200	3.41978	2.98674	1.35095	1.42941	1.50887	1.58948	1.67151
$\overline{r_i}/s$	0.30000	0.31000	0.32000	0.33000	0.34000	0.35000	0.36000	0.37000	0.38000	0.39000
$\overline{d_i}/c_i$	0.50000	0.48889	0.48000	0.47273	0.46667	0.46154	0.45714	0.45333	0.45000	0.44706
$T_{\text{remote},i}$	2.35314	2.35203	2.35314	2.35587	2.35981	2.36468	2.37029	2.37648	2.38314	2.39020
T_i	2.59956	2.61674	2.63411	2.63180	2.56342	1.56083	1.65604	1.74578	1.83044	1.91047
$P_{\text{comp},i}$	2.00000	1.95000	1.90000	1.98770	2.50468	9.67000	9.09200	8.51400	7.93600	7.35800
$P_{t,i}$	0.29370	0.34412	0.39923	0.45988	0.52711	0.60226	0.68704	0.78371	0.89526	1.02578
$P_{\text{comm},i}$	0.15777	0.19125	0.22962	0.26568	0.28236	0.10071	0.13617	0.17948	0.23239	0.29733
P_i	2.15777	2.14125	2.12962	2.25337	2.78704	9.77071	9.22817	8.69348	8.16839	7.65533
R_i	5.60927	5.60309	5.60964	5.93043	7.14436	15.25045	15.28223	15.17692	14.95174	14.62525
$s = 5.00000$, $\rho = 0.95985$, $W = 1.55314$, $T = 2.36211$, $P = 98.00000$, $R = 231.48671$										

TABLE 13. Performance data of game(λ_i, c_i) (idle-speed model, $N = 64$).

	UE$_1$	UE$_2$	UE$_3$	UE$_4$	UE$_5$	UE$_6$	UE$_7$	UE$_8$	UE$_9$	UE$_{10}$
λ_i	1.50000	1.55000	1.40627	1.36792	1.34840	1.33772	1.33251	1.33111	1.33250	1.33601
c_i	8.93076	9.18771	9.77637	10.10765	10.38171	10.61381	10.80628	10.95797	11.06613	11.12693
ρ_i	0.00000	0.00000	0.18233	0.25858	0.31459	0.36075	0.40070	0.43634	0.46880	0.49887
$\overline{x_i}$	1.00000	0.96875	0.94118	0.91667	0.89474	0.87500	0.85714	0.84091	0.82609	0.81250
W_i	0.00000	0.00000	0.13642	0.20780	0.26693	0.32096	0.37252	0.42313	0.47388	0.52574
$T_{\text{local},i}$	1.00000	0.96875	1.07760	1.12447	1.16167	1.19596	1.22966	1.26403	1.29997	1.33824
$\overline{r_i}/s$	0.30000	0.31000	0.32000	0.33000	0.34000	0.35000	0.36000	0.37000	0.38000	0.39000
$\overline{d_i}/c_i$	0.11197	0.11973	0.12274	0.12862	0.13485	0.14133	0.14806	0.15514	0.16266	0.17076
$T_{\text{remote},i}$	1.19885	1.21660	1.22962	1.24549	1.26173	1.27820	1.29494	1.31201	1.32953	1.34763
T_i	1.19885	1.21660	1.21121	1.22480	1.24103	1.25883	1.27799	1.29856	1.32070	1.34468
$P_{\text{comp},i}$	1.00000	1.00000	1.47425	1.71212	1.90854	2.08224	2.23697	2.37272	2.48797	2.58042
$P_{t,i}$	3.43652	3.57981	4.06197	4.32744	4.56370	4.78824	5.00691	5.22304	5.43945	5.65937
$P_{\text{comm},i}$	0.57719	0.66432	0.70115	0.76135	0.82984	0.90523	0.98784	1.07859	1.17896	1.29108
P_i	1.57719	1.66432	2.17540	2.47347	2.73838	2.98747	3.22481	3.45131	3.66694	3.87151
R_i	1.89082	2.02481	2.63487	3.02951	3.39842	3.76071	4.12126	4.48173	4.84294	5.20593
$s = 5.00000$, $\rho = 0.95426$, $W = 0.78688$, $T = 1.26944$, $P = 93.99786$, $R = 119.32448$										

the original parameter setting in Table 2, all UEs and the MEC choose high server execution speeds, and all get significantly reduced average response time, increased average power consumption, and significantly reduced cost-performance ratio.

3) GAME(c_i)

Tables 9 and 10 demonstrate performance data of Game(c_i) for the idle-speed model and the constant-speed model respectively. It is clear that in the stable situation, compared with the original parameter setting in Table 2, all UEs increase

TABLE 14. Performance data of game(λ_i, c_i) (constant-speed model, $N = 21$).

	UE$_1$	UE$_2$	UE$_3$	UE$_4$	UE$_5$	UE$_6$	UE$_7$	UE$_8$	UE$_9$	UE$_{10}$
λ_i	1.27866	1.28490	1.29344	1.30389	1.31597	1.32939	1.34393	1.35935	1.37545	1.39200
c_i	9.78441	9.99001	10.17733	10.34280	10.48291	10.59412	10.67275	10.71487	10.71625	10.67221
ρ_i	0.22134	0.25681	0.28853	0.31726	0.34361	0.36803	0.39092	0.41259	0.43332	0.45337
$\overline{x_i}$	1.00000	0.96875	0.94118	0.91667	0.89474	0.87500	0.85714	0.84091	0.82609	0.81250
W_i	0.18477	0.21759	0.24809	0.27688	0.30445	0.33122	0.35759	0.38392	0.41059	0.43803
$T_{\text{local},i}$	1.18477	1.18634	1.18927	1.19355	1.19918	1.20622	1.21473	1.22483	1.23668	1.25053
$\overline{r_i}/s$	0.30000	0.31000	0.32000	0.33000	0.34000	0.35000	0.36000	0.37000	0.38000	0.39000
$\overline{d_i}/c_i$	0.10220	0.11011	0.11791	0.12569	0.13355	0.14159	0.14991	0.15866	0.16797	0.17803
$T_{\text{remote},i}$	0.80590	0.82381	0.84161	0.85939	0.87725	0.89529	0.91361	0.93236	0.95167	0.97173
T_i	0.86181	0.88581	0.90822	0.92948	0.94997	0.97002	0.98991	1.00992	1.03035	1.05151
$P_{\text{comp},i}$	3.25000	3.43200	3.60100	3.75400	3.88800	4.00000	4.08700	4.14600	4.17400	4.16800
$P_{t,i}$	4.29480	4.38662	4.48094	4.57633	4.67185	4.76701	4.86170	4.95626	5.05162	5.14941
$P_{\text{comm},i}$	0.56126	0.62062	0.68338	0.75001	0.82107	0.89727	0.97951	1.06893	1.16710	1.27613
P_i	3.81126	4.05262	4.28438	4.50401	4.70907	4.89727	5.06651	5.21493	5.34110	5.44413
R_i	3.28457	3.58986	3.89116	4.18639	4.47349	4.75044	5.01537	5.26667	5.50321	5.72455
$s = 5.00000$, $\rho = 0.91995$, $W = 0.40370$, $T = 0.88872$, $P = 98.00000$, $R = 87.09486$										

TABLE 15. Performance data of game(s_i, c_i) (idle-speed model, $N = 6$).

	UE$_1$	UE$_2$	UE$_3$	UE$_4$	UE$_5$	UE$_6$	UE$_7$	UE$_8$	UE$_9$	UE$_{10}$
s_i	3.98684	4.11855	4.25814	4.40539	4.56026	4.72280	4.89327	5.00000	5.00000	5.00000
c_i	10.61310	10.78743	10.94341	11.07796	11.18775	11.26915	11.31816	11.25390	11.02572	10.74226
ρ_i	0.33297	0.34417	0.35471	0.36462	0.37390	0.38259	0.39066	0.40386	0.42598	0.44870
$\overline{x_i}$	0.37624	0.37635	0.37575	0.37454	0.37279	0.37054	0.36785	0.37000	0.38000	0.39000
W_i	0.12208	0.12837	0.13426	0.13970	0.14471	0.14925	0.15329	0.16293	0.18330	0.20632
$T_{\text{local},i}$	0.49832	0.50472	0.51001	0.51425	0.51749	0.51979	0.52115	0.53293	0.56330	0.59632
$\overline{r_i}/s$	0.39160	0.40465	0.41770	0.43076	0.44381	0.45686	0.46992	0.48297	0.49602	0.50908
$\overline{d_i}/c_i$	0.09422	0.10197	0.10966	0.11735	0.12514	0.13311	0.14137	0.15106	0.16325	0.17687
$T_{\text{remote},i}$	0.50559	0.52639	0.54712	0.56787	0.58871	0.60974	0.63105	0.65379	0.67904	0.70571
T_i	0.50130	0.51360	0.52522	0.53623	0.54669	0.55667	0.56621	0.58248	0.61075	0.64117
$P_{\text{comp},i}$	6.29253	6.54604	6.78834	7.01483	7.22055	7.40013	7.54779	7.56264	7.38970	7.16956
$P_{t,i}$	5.30663	5.34554	5.39003	5.43870	5.49038	5.54420	5.59951	5.56463	5.39338	5.22531
$P_{\text{comm},i}$	0.30750	0.34640	0.38772	0.43176	0.47887	0.52949	0.58418	0.63758	0.68591	0.73890
P_i	6.60003	6.89245	7.17606	7.44660	7.69943	7.92962	8.13197	8.20023	8.07561	7.90846
R_i	3.30857	3.53999	3.76905	3.99311	4.20923	4.41416	4.60437	4.77648	4.93221	5.07067
$s = 3.83046$, $\rho = 0.59305$, $W = 0.01976$, $T = 0.60674$, $P = 40.95532$, $R = 24.84917$										

the data communication speed, and get reduced average response time, slightly increased average power consumption, and reduced cost-performance ratio. The MEC increases its server execution speed and gets significantly reduced average response time, reduced average power consumption, and significantly reduced cost-performance ratio.

4) GAME(λ_i, s_i)

Tables 11 and 12 demonstrate performance data of Game(λ_i, s_i) for the idle-speed model and the constant-speed model respectively. It is clear that for the idle-speed mode, in the stable situation, compared with the original parameter setting in Table 2, UE$_1$ and UE$_2$ increase their amount of

TABLE 16. performance data of game(s_i, c_i) (constant-speed model, $N = 6$).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
s_i	2.22019	2.34349	2.47105	2.60291	2.73912	2.87978	3.02505	3.17516	3.33043	3.49135
c_i	8.61434	8.81448	8.99761	9.16087	9.30120	9.41528	9.49947	9.54975	9.56166	9.53021
ρ_i	0.59792	0.60486	0.61124	0.61711	0.62250	0.62743	0.63192	0.63596	0.63953	0.64258
$\overline{x_i}$	0.67562	0.66141	0.64750	0.63391	0.62064	0.60768	0.59503	0.58265	0.57050	0.55852
W_i	0.65305	0.65808	0.66172	0.66408	0.66523	0.66521	0.66401	0.66161	0.65789	0.65269
$T_{\text{local},i}$	1.32866	1.31948	1.30922	1.29799	1.28587	1.27289	1.25905	1.24426	1.22839	1.21121
$\overline{r_i}/s$	0.51334	0.53045	0.54756	0.56467	0.58178	0.59889	0.61601	0.63312	0.65023	0.66734
$\overline{d_i}/c_i$	0.11609	0.12479	0.13337	0.14191	0.15052	0.15932	0.16843	0.17802	0.18825	0.19937
$T_{\text{remote},i}$	0.74085	0.76667	0.79236	0.81801	0.84373	0.86964	0.89586	0.92256	0.94991	0.97813
T_i	1.08766	1.09283	1.09731	1.10120	1.10459	1.10756	1.11014	1.11236	1.11421	1.11565
$P_{\text{comp},i}$	5.92926	6.21736	6.49550	6.75887	7.00222	7.21986	7.40566	7.55306	7.65505	7.70423
$P_{t,i}$	3.15900	3.25105	3.34520	3.44094	3.53786	3.63567	3.73420	3.83354	3.93403	4.03649
$P_{\text{comm},i}$	0.22553	0.25783	0.29267	0.33033	0.37116	0.41559	0.46417	0.51762	0.57692	0.64339
P_i	6.15479	6.47519	6.78817	7.08920	7.37338	7.63545	7.86983	8.07068	8.23197	8.34762
R_i	6.69433	7.07629	7.44870	7.80660	8.14457	8.45670	8.73662	8.97751	9.17215	9.31301
$s = 2.92205$, $\rho = 0.76041$, $W = 0.11143$, $T = 0.86404$, $P = 40.38431$, $R = 34.89384$										

TABLE 17. Performance data of game(λ_i, s_i, c_i) (idle-speed model, $N = 85$).

	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6	UE_7	UE_8	UE_9	UE_{10}
λ_i	1.49685	1.50197	1.50510	1.50344	1.49411	1.47247	1.42970	1.34159	1.02957	1.06367
s_i	1.00000	1.03635	1.13495	1.25478	1.40416	1.59711	1.86143	2.27475	3.40000	3.40000
c_i	8.82553	9.06294	9.30624	9.55791	9.82376	10.11434	10.45245	10.90808	12.02111	11.91756
ρ_i	0.00473	0.07183	0.13378	0.19272	0.24927	0.30410	0.35808	0.41348	0.48642	0.50834
$\overline{x_i}$	1.50000	1.49564	1.40975	1.31497	1.21069	1.09573	0.96700	0.81328	0.55882	0.57353
W_i	0.00464	0.07524	0.14152	0.20404	0.26130	0.31123	0.35062	0.37266	0.34403	0.38544
$T_{\text{local},i}$	1.50464	1.57088	1.55127	1.51901	1.47200	1.40696	1.31762	1.18594	0.90285	0.95897
$\overline{r_i}/s$	0.30000	0.31000	0.32000	0.33000	0.34000	0.35000	0.36000	0.37000	0.38000	0.39000
$\overline{d_i}/c_i$	0.11331	0.12137	0.12895	0.13601	0.14251	0.14830	0.15307	0.15585	0.14974	0.15943
$T_{\text{remote},i}$	1.16075	1.17881	1.19639	1.21345	1.22995	1.24575	1.26052	1.27329	1.27718	1.29687
T_i	1.16147	1.19096	1.21744	1.24059	1.25927	1.27131	1.27226	1.24928	1.10569	1.14328
$P_{\text{comp},i}$	1.00473	1.07329	1.15509	1.25791	1.39319	1.58176	1.86850	2.39069	4.37380	4.23201
$P_{t,i}$	3.34196	3.46684	3.61490	3.79138	4.00582	4.27449	4.63013	5.16729	6.63308	6.65036
$P_{\text{comm},i}$	0.56681	0.63200	0.70157	0.77529	0.85295	0.93343	1.01330	1.08040	1.02258	1.12776
P_i	1.57154	1.70530	1.85666	2.03320	2.24614	2.51520	2.88180	3.47109	5.39638	5.35977
R_i	1.82530	2.03095	2.26036	2.52238	2.82849	3.19760	3.66641	4.33638	5.96672	6.12774
$s = 5.00000$, $\rho = 0.95234$, $W = 0.74744$, $T = 1.22917$, $P = 93.82982$, $R = 115.33281$										

computation offloading and choose the lowest available server execution speed, and get significantly reduced average response time, reduced average power consumption, and significantly reduced cost-performance ratio. UE_3, $UE_4, \ldots, UE_{10}$ decrease/increase their amount of computation offloading and choose the highest available server execution speed, and get significantly reduced average response time, increased average power consumption,

TABLE 18. Performance data of game(λ_i, s_i, c_i) (constant-speed model, $N = 55$).

	UE$_1$	UE$_2$	UE$_3$	UE$_4$	UE$_5$	UE$_6$	UE$_7$	UE$_8$	UE$_9$	UE$_{10}$
λ_i	1.21355	1.26575	1.31807	1.37050	1.42301	1.47558	1.52820	1.53465	1.44707	1.29035
s_i	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.11746	1.48168	2.04205
c_i	9.68576	9.85932	10.02590	10.18565	10.33881	10.48567	10.62663	10.81547	11.14700	11.63846
ρ_i	0.42967	0.44059	0.45108	0.46117	0.47088	0.48023	0.48923	0.52207	0.58081	0.62992
$\overline{x_i}$	1.50000	1.55000	1.60000	1.65000	1.70000	1.75000	1.80000	1.65555	1.28233	0.95492
W_i	0.73455	0.79351	0.85464	0.91793	0.98337	1.05095	1.12067	1.17551	1.15485	1.05649
$T_{\text{local},i}$	2.23455	2.34351	2.45464	2.56793	2.68337	2.80095	2.92067	2.83105	2.43718	2.01142
$\overline{r_i}/s$	0.30000	0.31000	0.32000	0.33000	0.34000	0.35000	0.36000	0.37000	0.38000	0.39000
$\overline{d_i}/c_i$	0.10324	0.11157	0.11969	0.12763	0.13541	0.14305	0.15057	0.15718	0.16148	0.16325
$T_{\text{remote},i}$	1.32586	1.34418	1.36230	1.38024	1.39802	1.41566	1.43318	1.44979	1.46409	1.47586
T_i	1.49939	1.52745	1.55478	1.58143	1.60745	1.63289	1.65779	1.68524	1.69606	1.65703
$P_{\text{comp},i}$	2.00000	1.95000	1.90000	1.85000	1.80000	1.75000	1.70000	1.81166	2.31723	3.29348
$P_{t,i}$	4.18674	4.24513	4.31841	4.40864	4.51851	4.65155	4.81240	5.06519	5.53267	6.28453
$P_{\text{comm},i}$	0.52457	0.59949	0.68127	0.77115	0.87069	0.98188	1.10731	1.22182	1.29282	1.32385
P_i	2.52457	2.54949	2.58127	2.62115	2.67069	2.73188	2.80731	3.03348	3.61005	4.61732
R_i	3.78530	3.89422	4.01331	4.14516	4.29300	4.46085	4.65391	5.11216	6.12286	7.65105
$s = 5.00000$, $\rho = 0.96007$, $W = 0.92261$, $T = 1.40726$, $P = 98.00000$, $R = 137.91141$										

and significantly reduced cost-performance ratio. For the constant-speed mode, compared with the original parameter setting in Table 2, UE$_1$, UE$_2$,..., UE$_5$ increase their amount of computation offloading and choose low server execution speeds, and get significantly reduced average response time, reduced average power consumption, and significantly reduced cost-performance ratio. UE$_6$, UE$_7$,..., UE$_{10}$ decrease their amount of computation offloading and choose the highest available server execution speed, and get significantly reduced average response time, increased average power consumption, and significantly reduced cost-performance ratio. The MEC sets its server execution speed to the maximum available, and gets reduced average response time, increased average power consumption, and reduced cost-performance ratio.

5) GAME(λ_i, c_i)

Tables 13 and 14 demonstrate performance data of Game(λ_i, c_i) for the idle-speed model and the constant-speed model respectively. It is clear that in the stable situation, compared with the original parameter setting in Table 2, all UEs increase their amount of computation offloading and increase the data communication speed, and get significantly reduced average response time, significantly reduced average power consumption, and significantly reduced cost-performance ratio. The MEC sets its server execution speed to the maximum available, and gets significantly reduced average response time, increased average power consumption, and reduced cost-performance ratio.

6) GAME(s_i, c_i)

Tables 15 and 16 demonstrate performance data of Game(s_i, c_i) for the idle-speed model and the constant-speed model respectively. It is clear that in the stable situation, compared with the original parameter setting in Table 2, all UEs choose high server execution and data communication speeds, and get significantly reduced average response time, increased average power consumption, and significantly reduced cost-performance ratio. The MEC increases its server execution speed and gets significantly reduced average response time, reduced average power consumption, and significantly reduced cost-performance ratio.

7) GAME(λ_i, s_i, c_i)

Tables 17 and 18 demonstrate performance data of Game(λ_i, s_i, c_i) for the idle-speed model and the constant-speed model respectively. It is clear that in the stable situation, compared with the original parameter setting in Table 2, all UEs increase their amount of computation offloading, choose low server execution speeds (except UE$_9$ and UE$_{10}$), and choose high data communication speeds, and get significantly reduced average response time, reduced average power consumption (except UE$_9$ and UE$_{10}$), and significantly reduced cost-performance ratio. The MEC sets its server execution speed to the maximum available, and gets reduced average response time, increased average power consumption, and reduced cost-performance ratio.

VIII. CONCLUDING REMARKS

We have established a non-cooperative game framework to study stabilization of a competitive mobile edge computing environment. Our framework includes a set of non-cooperative games for multiple heterogeneous mobile users and a mobile edge cloud to play. All players can optimize their cost-performance ratios by using the algorithms developed in this paper. Furthermore, we are able to obtain the Nash equilibrium of the games, so that we can examine the stable situation of a competitive mobile edge computing environment. Our investigation in this paper can help each UE and MEC to optimize a combined quantity of performance and cost, and can provide an environment in which everyone's benefit is optimized and no one wants to change.

The research in this paper can be extended to multiple heterogeneous MECs. In this case, there is a load distribution problem for each UE, i.e., how the offloaded tasks are distributed to the multiple MECs. In addition, each UE should also decide the data transmission rate to each MEC. Therefore, each UE has $2k+1$ parameters to determine, where k is the number of MECs. Hence, the convex optimization problem for each UE is significantly more complicated than the situation of a single MEC. The stability problem for such a competitive mobile edge computing environment is much more challenging to solve, and certainly, more interesting. As mentioned earlier, one special case of the above problem has been solved in [20], where only the load distribution (i.e., computation offloading strategy optimization) problem was solved; however, there was no consideration on computation/communication speeds optimization.

APPENDIX 1. DERIVATION OF FIRST ORDER PARTIAL DERIVATIVES

(1) $\partial T_i/\partial\lambda_i$:

$$\frac{\partial T_i}{\partial\lambda_i}=-\frac{1}{\tilde{\lambda}_i}\left(\frac{\overline{r_i}}{s_i}+W_i\right)+\frac{\tilde{\lambda}_i-\lambda_i}{\tilde{\lambda}_i}\cdot\frac{\partial W_i}{\partial\lambda_i}+\frac{1}{\tilde{\lambda}_i}\left(\frac{\overline{r_i}}{s}+\frac{\overline{d_i}}{c_i}+W\right)+\frac{\lambda_i}{\tilde{\lambda}_i}\cdot\frac{\partial W}{\partial\lambda_i},$$

where

$$\frac{\partial W_i}{\partial\lambda_i}=-\frac{\overline{r_i^2}/s_i^2}{2}\times\left(\frac{1}{1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i)}+\frac{(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i)}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^2}\right),$$

$$\frac{\partial W}{\partial\lambda_i}=\frac{m^{m-2}}{2m!}\left(\frac{\partial(\lambda\overline{x^2})}{\partial\lambda_i}D+\lambda\overline{x^2}\cdot\frac{\partial D}{\partial\rho}\cdot\frac{\partial\rho}{\partial\lambda_i}\right),$$

$$\frac{\partial(\lambda\overline{x^2})}{\partial\lambda_i}=\frac{\overline{r_i^2}}{s^2}+2\frac{\overline{r_i}}{s}\cdot\frac{\overline{d_i}}{c_i}+\frac{\overline{d_i^2}}{c_i^2},$$

$$\frac{\partial D}{\partial\rho}=\frac{\partial p_0}{\partial\rho}\cdot\frac{\rho^{m-1}}{(1-\rho)^2}+p_0\frac{\rho^{m-2}((m-1)-(m-3)\rho)}{(1-\rho)^3},$$

$$\frac{\partial p_0}{\partial\rho}=-p_0^2\left(\sum_{k=1}^{m-1}\frac{m^k\rho^{k-1}}{(k-1)!}+\frac{m^m}{m!}\cdot\frac{\rho^{m-1}(m-(m-1)\rho)}{(1-\rho)^2}\right),$$

$$\frac{\partial\rho}{\partial\lambda_i}=\frac{1}{m}\left(\frac{\overline{r_i}}{s}+\frac{\overline{d_i}}{c_i}\right).$$

(2) $\partial T_i/\partial s_i$:

$$\frac{\partial T_i}{\partial s_i}=\frac{\tilde{\lambda}_i-\lambda_i}{\tilde{\lambda}_i}\left(-\frac{\overline{r_i}}{s_i^2}+\frac{\partial W_i}{\partial s_i}\right),$$

where

$$\frac{\partial W_i}{\partial s_i}=-\left(\frac{(\tilde{\lambda}_i-\lambda_i)(\overline{r_i^2}/s_i^3)}{1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i)}+\frac{(\tilde{\lambda}_i-\lambda_i)^2(\overline{r_i^2}/s_i^2)(\overline{r_i}/s_i^2)}{2(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^2}\right).$$

(3) $\partial T_i/\partial c_i$:

$$\frac{\partial T_i}{\partial c_i}=\frac{\lambda_i}{\tilde{\lambda}_i}\left(-\frac{\overline{d_i}}{c_i^2}+\frac{\partial W}{\partial c_i}\right),$$

where

$$\frac{\partial W}{\partial c_i}=\frac{m^{m-2}}{2m!}\left(\frac{\partial(\lambda\overline{x^2})}{\partial c_i}D+\lambda\overline{x^2}\cdot\frac{\partial D}{\partial\rho}\cdot\frac{\partial\rho}{\partial c_i}\right),$$

$$\frac{\partial(\lambda\overline{x^2})}{\partial c_i}=-2\lambda_i\left(\frac{\overline{r_i}}{s}\cdot\frac{\overline{d_i}}{c_i^2}+\frac{\overline{d_i^2}}{c_i^3}\right),$$

$$\frac{\partial\rho}{\partial c_i}=-\frac{\lambda_i}{m}\cdot\frac{\overline{d_i}}{c_i^2}.$$

(4) $\partial P_i/\partial\lambda_i$:

$$\frac{\partial P_i}{\partial\lambda_i}=-\overline{r_i}\xi_i s_i^{\alpha_i-1}+\frac{\overline{d_i}}{c_i}\cdot\frac{2^{c_i/w_i}-1}{\beta_i},$$

for the idle-speed model, and

$$\frac{\partial P_i}{\partial\lambda_i}=\frac{\overline{d_i}}{c_i}\cdot\frac{2^{c_i/w_i}-1}{\beta_i},$$

for the constant-speed model.

(5) $\partial P_i/\partial s_i$:

$$\frac{\partial P_i}{\partial s_i}=(\tilde{\lambda}_i-\lambda_i)\overline{r_i}\xi_i(\alpha_i-1)s_i^{\alpha_i-2},$$

for the idle-speed model, and

$$\frac{\partial P_i}{\partial s_i}=\xi_i\alpha_i s_i^{\alpha_i-1},$$

for the constant-speed model.

(6) $\partial P_i/\partial c_i$:

$$\frac{\partial P_i}{\partial c_i}=\lambda_i\frac{\overline{d_i}}{\beta_i}\left(\frac{2^{c_i/w_i}\ln 2}{c_i w_i}-\frac{2^{c_i/w_i}-1}{c_i^2}\right),$$

for both idle-speed and constant-speed models.

(7) $\partial T/\partial s$:

$$\frac{\partial T}{\partial s}=-\sum_{i=1}^{n}\frac{\lambda_i}{\lambda}\cdot\frac{\overline{r_i}}{s^2}+\frac{\partial W}{\partial s},$$

where

$$\frac{\partial W}{\partial s}=\frac{m^{m-2}}{2m!}\left(\frac{\partial(\lambda\overline{x^2})}{\partial s}D+\lambda\overline{x^2}\cdot\frac{\partial D}{\partial\rho}\cdot\frac{\partial\rho}{\partial s}\right),$$

$$\frac{\partial(\lambda\overline{x^2})}{\partial s} = -2\sum_{i=1}^{n}\lambda_i\left(\frac{\overline{r_i^2}}{s^3} + \frac{\overline{r_i}}{s^2}\cdot\frac{\overline{d_i}}{c_i}\right),$$

$$\frac{\partial\rho}{\partial s} = -\sum_{i=1}^{n}\frac{\lambda_i}{m}\cdot\frac{\overline{r_i}}{s^2}.$$

(8) $\partial P/\partial s$:

$$\frac{\partial P}{\partial s} = m\xi\left(\frac{\partial\rho}{\partial s}s^{\alpha} + \rho\alpha s^{\alpha-1}\right),$$

for the idle-speed model, and

$$\frac{\partial P}{\partial s} = m\xi\alpha s^{\alpha-1},$$

for the constant-speed model.

APPENDIX 2. DERIVATION OF SECOND ORDER PARTIAL DERIVATIVES

(1) $\partial^2 T_i/\partial\lambda_i^2$:

$$\frac{\partial^2 T_i}{\partial\lambda_i^2} = -\frac{2}{\tilde{\lambda}_i}\cdot\frac{\partial W_i}{\partial\lambda_i} + \frac{\tilde{\lambda}_i-\lambda_i}{\tilde{\lambda}_i}\cdot\frac{\partial^2 W_i}{\partial\lambda_i^2} + \frac{2}{\tilde{\lambda}_i}\cdot\frac{\partial W}{\partial\lambda_i} + \frac{\lambda_i}{\tilde{\lambda}_i}\cdot\frac{\partial^2 W}{\partial\lambda_i^2},$$

where

$$\frac{\partial^2 W_i}{\partial\lambda_i^2} = (\overline{r_i^2}/s_i^2)\left(\frac{\overline{r_i}/s_i}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^2} + \frac{(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i)^2}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^3}\right),$$

$$\frac{\partial^2 W}{\partial\lambda_i^2} = \frac{m^{m-2}}{2m!}\left(2\cdot\frac{\partial(\lambda\overline{x^2})}{\partial\lambda_i}\cdot\frac{\partial D}{\partial\rho}\cdot\frac{\partial\rho}{\partial\lambda_i} + \lambda\overline{x^2}\cdot\frac{\partial^2 D}{\partial\rho^2}\left(\frac{\partial\rho}{\partial\lambda_i}\right)^2\right),$$

$$\frac{\partial^2 D}{\partial\rho^2} = \frac{\partial^2 p_0}{\partial\rho^2}\cdot\frac{\rho^{m-1}}{(1-\rho)^2} + 2\cdot\frac{\partial p_0}{\partial\rho}\cdot\frac{\rho^{m-2}((m-1)-(m-3)\rho)}{(1-\rho)^3} + p_0\left(\frac{\rho^{m-3}(m-1)((m-2)-(m-3)\rho)}{(1-\rho)^3} + \frac{3\rho^{m-2}((m-1)-(m-3)\rho)}{(1-\rho)^4}\right),$$

$$\frac{\partial^2 p_0}{\partial\rho^2} = -2p_0\frac{\partial p_0}{\partial\rho}\left(\sum_{k=1}^{m-1}\frac{m^k\rho^{k-1}}{(k-1)!} + \frac{m^m}{m!}\cdot\frac{\rho^{m-1}(m-(m-1)\rho)}{(1-\rho)^2}\right) - p_0^2\left(\sum_{k=2}^{m-1}\frac{m^k\rho^{k-2}}{(k-2)!} + \frac{m^m}{m!}\left(\frac{m(m-1)\rho^{m-2}(1-\rho)}{(1-\rho)^2} + \frac{2\rho^{m-1}(m-(m-1)\rho)}{(1-\rho)^3}\right)\right).$$

(2) $\partial^2 T_i/\partial\lambda_i\partial s_i$:

$$\frac{\partial^2 T_i}{\partial\lambda_i\partial s_i} = -\frac{1}{\tilde{\lambda}_i}\left(-\frac{\overline{r_i}}{s_i^2} + \frac{\partial W_i}{\partial s_i}\right) + \frac{\tilde{\lambda}_i-\lambda_i}{\tilde{\lambda}_i}\cdot\frac{\partial^2 W_i}{\partial\lambda_i\partial s_i},$$

where

$$\frac{\partial^2 W_i}{\partial\lambda_i\partial s_i} = (\overline{r_i^2}/s_i^3)\left(\frac{1}{1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i)} + \frac{(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i)}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^2}\right) + (\overline{r_i^2}/s_i^2)\left(\frac{(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i^2)}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^2} + \frac{(\tilde{\lambda}_i-\lambda_i)^2(\overline{r_i}^2/s_i^3)}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^3}\right).$$

(3) $\partial^2 T_i/\partial\lambda_i\partial c_i$:

$$\frac{\partial^2 T_i}{\partial\lambda_i\partial c_i} = \frac{1}{\tilde{\lambda}_i}\left(-\frac{\overline{d_i}}{c_i^2} + \frac{\partial W}{\partial c_i}\right) + \frac{\lambda_i}{\tilde{\lambda}_i}\cdot\frac{\partial^2 W}{\partial\lambda_i\partial c_i},$$

where

$$\frac{\partial^2 W}{\partial\lambda_i\partial c_i} = \frac{m^{m-2}}{2m!}\left(\frac{\partial^2(\lambda\overline{x^2})}{\partial\lambda_i\partial c_i}D + \frac{\partial(\lambda\overline{x^2})}{\partial\lambda_i}\cdot\frac{\partial D}{\partial\rho}\cdot\frac{\partial\rho}{\partial c_i} + \frac{\partial(\lambda\overline{x^2})}{\partial c_i}\cdot\frac{\partial D}{\partial\rho}\cdot\frac{\partial\rho}{\partial\lambda_i} + \lambda\overline{x^2}\cdot\frac{\partial^2 D}{\partial\rho^2}\cdot\frac{\partial\rho}{\partial c_i}\cdot\frac{\partial\rho}{\partial\lambda_i} + \lambda\overline{x^2}\cdot\frac{\partial D}{\partial\rho}\cdot\frac{\partial^2\rho}{\partial\lambda_i\partial c_i}\right),$$

$$\frac{\partial^2(\lambda\overline{x^2})}{\partial\lambda_i\partial c_i} = -2\left(\frac{\overline{r_i}}{s}\cdot\frac{\overline{d_i}}{c_i^2} + \frac{\overline{d_i^2}}{c_i^3}\right),$$

$$\frac{\partial^2\rho}{\partial\lambda_i\partial c_i} = -\frac{1}{m}\cdot\frac{\overline{d_i}}{c_i^2}.$$

(4) $\partial^2 T_i/\partial s_i\partial\lambda_i = \partial^2 T_i/\partial\lambda_i\partial s_i$:

$$\frac{\partial^2 T_i}{\partial s_i\partial\lambda_i} = -\frac{1}{\tilde{\lambda}_i}\left(-\frac{\overline{r_i}}{s_i^2} + \frac{\partial W_i}{\partial s_i}\right) + \frac{\tilde{\lambda}_i-\lambda_i}{\tilde{\lambda}_i}\cdot\frac{\partial^2 W_i}{\partial s_i\partial\lambda_i},$$

where

$$\frac{\partial^2 W_i}{\partial s_i\partial\lambda_i} = \frac{\overline{r_i^2}/s_i^3}{1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i)} + \frac{2(\tilde{\lambda}_i-\lambda_i)(\overline{r_i^2}\,\overline{r_i}/s_i^4)}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^2} + \frac{(\tilde{\lambda}_i-\lambda_i)^2(\overline{r_i^2}\,\overline{r_i}^2/s_i^5)}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^3}.$$

(5) $\partial^2 T_i/\partial s_i^2$:

$$\frac{\partial^2 T_i}{\partial s_i^2} = \frac{\tilde{\lambda}_i-\lambda_i}{\tilde{\lambda}_i}\left(\frac{2\overline{r_i}}{s_i^3} + \frac{\partial^2 W_i}{\partial s_i^2}\right),$$

where

$$\frac{\partial^2 W_i}{\partial s_i^2} = \frac{3(\tilde{\lambda}_i-\lambda_i)(\overline{r_i^2}/s_i^4)}{1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i)} + \frac{3(\tilde{\lambda}_i-\lambda_i)^2(\overline{r_i^2}\,\overline{r_i}/s_i^5)}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^2} + \frac{(\tilde{\lambda}_i-\lambda_i)^3(\overline{r_i^2}\,\overline{r_i}^2/s_i^6)}{(1-(\tilde{\lambda}_i-\lambda_i)(\overline{r_i}/s_i))^3}.$$

(6) $\partial^2 T_i/\partial s_i \partial c_i$:

$$\frac{\partial^2 T_i}{\partial s_i \partial c_i} = 0.$$

(7) $\partial^2 T_i/\partial c_i \partial \lambda_i = \partial^2 T_i/\partial \lambda_i \partial c_i$:

$$\frac{\partial^2 T_i}{\partial c_i \partial \lambda_i} = \frac{1}{\tilde{\lambda}_i}\left(-\frac{\overline{d_i}}{c_i^2} + \frac{\partial W}{\partial c_i}\right) + \frac{\lambda_i}{\tilde{\lambda}_i} \cdot \frac{\partial^2 W}{\partial c_i \partial \lambda_i},$$

where

$$\frac{\partial^2 W}{\partial c_i \partial \lambda_i} = \frac{m^{m-2}}{2m!}\left(\frac{\partial^2(\lambda\overline{x^2})}{\partial c_i \partial \lambda_i} D + \frac{\partial(\lambda\overline{x^2})}{\partial c_i} \cdot \frac{\partial D}{\partial \rho} \cdot \frac{\partial \rho}{\partial \lambda_i} + \frac{\partial(\lambda\overline{x^2})}{\partial \lambda_i} \cdot \frac{\partial D}{\partial \rho} \cdot \frac{\partial \rho}{\partial c_i} + \lambda\overline{x^2} \cdot \frac{\partial^2 D}{\partial \rho^2} \cdot \frac{\partial \rho}{\partial \lambda_i} \cdot \frac{\partial \rho}{\partial c_i} + \lambda\overline{x^2} \cdot \frac{\partial D}{\partial \rho} \cdot \frac{\partial^2 \rho}{\partial c_i \partial \lambda_i}\right),$$

$$\frac{\partial^2(\lambda\overline{x^2})}{\partial c_i \partial \lambda_i} = -2\left(\frac{\overline{r_i}}{s} \cdot \frac{\overline{d_i}}{c_i^2} + \frac{\overline{d_i^2}}{c_i^3}\right),$$

$$\frac{\partial^2 \rho}{\partial c_i \partial \lambda_i} = -\frac{1}{m} \cdot \frac{\overline{d_i}}{c_i^2}.$$

(8) $\partial^2 T_i/\partial c_i \partial s_i = \partial^2 T_i/\partial s_i \partial c_i$:

$$\frac{\partial^2 T_i}{\partial c_i \partial s_i} = 0.$$

(9) $\partial^2 T_i/\partial c_i^2$:

$$\frac{\partial^2 T_i}{\partial c_i^2} = \frac{\lambda_i}{\tilde{\lambda}_i}\left(2\frac{\overline{d_i}}{c_i^3} + \frac{\partial^2 W}{\partial c_i^2}\right),$$

where

$$\frac{\partial^2 W}{\partial c_i^2} = \frac{m^{m-2}}{2m!}\left(\frac{\partial^2(\lambda\overline{x^2})}{\partial c_i^2} D + 2 \cdot \frac{\partial(\lambda\overline{x^2})}{\partial c_i} \cdot \frac{\partial D}{\partial \rho} \cdot \frac{\partial \rho}{\partial c_i} + \lambda\overline{x^2} \cdot \frac{\partial^2 D}{\partial \rho^2}\left(\frac{\partial \rho}{\partial c_i}\right)^2 + \lambda\overline{x^2} \cdot \frac{\partial D}{\partial \rho} \cdot \frac{\partial^2 \rho}{\partial c_i^2}\right),$$

$$\frac{\partial^2(\lambda\overline{x^2})}{\partial c_i^2} = 2\lambda_i\left(\frac{\overline{r_i}}{s} \cdot \frac{2\overline{d_i}}{c_i^3} + \frac{3\overline{d_i^2}}{c_i^4}\right),$$

$$\frac{\partial^2 \rho}{\partial c_i^2} = 2\frac{\lambda_i}{m} \cdot \frac{\overline{d_i}}{c_i^3}.$$

(10) $\partial^2 P_i/\partial \lambda_i^2$:

$$\frac{\partial^2 P_i}{\partial \lambda_i^2} = 0,$$

for both idle-speed and constant-speed models.

(11) $\partial^2 P_i/\partial \lambda_i \partial s_i$:

$$\frac{\partial^2 P_i}{\partial \lambda_i \partial s_i} = -\overline{r_i}\xi_i(\alpha_i - 1)s_i^{\alpha_i - 2},$$

for the idle-speed model, and

$$\frac{\partial^2 P_i}{\partial \lambda_i \partial s_i} = 0,$$

for the constant-speed model.

(12) $\partial^2 P_i/\partial \lambda_i \partial c_i$:

$$\frac{\partial^2 P_i}{\partial \lambda_i \partial c_i} = \frac{\overline{d_i}}{\beta_i}\left(\frac{2^{c_i/w_i}\ln 2}{c_i w_i} - \frac{2^{c_i/w_i} - 1}{c_i^2}\right),$$

for both idle-speed and constant-speed models.

(13) $\partial^2 P_i/\partial s_i \partial \lambda_i = \partial^2 P_i/\partial \lambda_i \partial s_i$:

$$\frac{\partial^2 P_i}{\partial s_i \partial \lambda_i} = -\overline{r_i}\xi_i(\alpha_i - 1)s_i^{\alpha_i - 2},$$

for the idle-speed model, and

$$\frac{\partial^2 P_i}{\partial s_i \partial \lambda_i} = 0,$$

for the constant-speed model.

(14) $\partial^2 P_i/\partial s_i^2$:

$$\frac{\partial^2 P_i}{\partial s_i^2} = (\tilde{\lambda}_i - \lambda_i)\overline{r_i}\xi_i(\alpha_i - 1)(\alpha_i - 2)s_i^{\alpha_i - 3},$$

for the idle-speed model, and

$$\frac{\partial^2 P_i}{\partial s_i^2} = \xi_i\alpha_i(\alpha_i - 1)s_i^{\alpha_i - 2},$$

for the constant-speed model.

(15) $\partial^2 P_i/\partial s_i \partial c_i$:

$$\frac{\partial^2 P_i}{\partial s_i \partial c_i} = 0,$$

for both idle-speed and constant-speed models.

(16) $\partial^2 P_i/\partial c_i \partial \lambda_i = \partial^2 P_i/\partial \lambda_i \partial c_i$:

$$\frac{\partial^2 P_i}{\partial c_i \partial \lambda_i} = \frac{\overline{d_i}}{\beta_i}\left(\frac{2^{c_i/w_i}\ln 2}{c_i w_i} - \frac{2^{c_i/w_i} - 1}{c_i^2}\right),$$

for both idle-speed and constant-speed models.

(17) $\partial^2 P_i/\partial c_i \partial s_i = \partial^2 P_i/\partial s_i \partial c_i$:

$$\frac{\partial^2 P_i}{\partial c_i \partial s_i} = 0,$$

for both idle-speed and constant-speed models.

(18) $\partial^2 P_i/\partial c_i^2$:

$$\frac{\partial^2 P_i}{\partial c_i^2} = \lambda_i\frac{\overline{d_i}}{\beta_i}\left(\frac{2^{c_i/w_i}(\ln 2)^2}{c_i w_i^2} - 2 \cdot \frac{2^{c_i/w_i}\ln 2}{c_i^2 w_i} + \frac{2(2^{c_i/w_i} - 1)}{c_i^3}\right),$$

for both idle-speed and constant-speed models.

(19) $\partial^2 T/\partial s^2$:

$$\frac{\partial^2 T}{\partial s^2} = 2\sum_{i=1}^{n}\frac{\lambda_i}{\lambda} \cdot \frac{\overline{r_i}}{s^3} + \frac{\partial^2 W}{\partial s^2},$$

where

$$\frac{\partial^2 W}{\partial s^2} = \frac{m^{m-2}}{2m!}\left(\frac{\partial^2(\lambda\overline{x^2})}{\partial s^2} D + 2 \cdot \frac{\partial(\lambda\overline{x^2})}{\partial s} \cdot \frac{\partial D}{\partial \rho} \cdot \frac{\partial \rho}{\partial s} + \lambda\overline{x^2} \cdot \frac{\partial^2 D}{\partial \rho^2}\left(\frac{\partial \rho}{\partial s}\right)^2 + \lambda\overline{x^2} \cdot \frac{\partial D}{\partial \rho} \cdot \frac{\partial^2 \rho}{\partial s^2}\right),$$

$$\frac{\partial^2(\lambda\overline{x^2})}{\partial s^2} = 2\sum_{i=1}^{n}\lambda_i\left(\frac{3\overline{r_i^2}}{s^4} + \frac{2\overline{r_i}}{s^3} \cdot \frac{\overline{d_i}}{c_i}\right),$$

$$\frac{\partial^2 \rho}{\partial s^2} = 2\sum_{i=1}^{n} \frac{\lambda_i}{m} \cdot \frac{\overline{r_i}}{s^3}.$$

(20) $\partial^2 P/\partial s^2$:

$$\frac{\partial^2 P}{\partial s^2} = m\xi\left(\frac{\partial^2 \rho}{\partial s^2} s^\alpha + 2\frac{\partial \rho}{\partial s}\alpha s^{\alpha-1} + \rho\alpha(\alpha - 1s^{\alpha-2}\right),$$

for the idle-speed model, and

$$\frac{\partial^2 P}{\partial s^2} = m\xi\alpha(\alpha-1)s^{\alpha-2},$$

for the constant-speed model.

APPENDIX 3. NUMERICAL DEMONSTRATION

For the same parameter setting in Section 7.1, Tables 3 and 4 show that all the leading principal minors mentioned in Section 5.3 are positive.

APPENDIX 4. PERFORMANCE DATA OF THE GAMES

Tables 5–18 demonstrate the performance data of our games. They are explained in Section 7.2.

ACKNOWLEDGMENTS

The author would like to thank the five anonymous reviewers for their constructive comments and suggestions.

REFERENCES

[1] B. Bernstein and R. A. Toupin, "Some properties of the Hessian matrix of a strictly convex function," *J. für die reine und angewandte Mathematik*, vol. 210, pp. 65–72, 1962.

[2] A. Bhattacharya and P. De, "A survey of adaptation techniques in computation offloading," *J. Netw. Comput. Appl.*, vol. 78, pp. 97–115, Jan. 2017.

[3] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis*, 2nd ed. Boston, MA, USA: Prindle, Weber & Schmidt, 1981.

[4] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[5] V. Cardellini, V. De Nitto Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. L. Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Program.*, vol. 157, no. 2, pp. 421–449, Jun. 2016.

[6] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[7] J. Chen, K. Li, Q. Deng, S. Yu, K. Li, and P. S. Yu, "QoE-aware computation offloading game algorithm for 5G mobile edge computing," *IEEE Trans. Mobile Comput.*, 2019.

[8] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Serv. Comput.*, to be published.

[9] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[10] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[11] G. T. Gilbert, "Positive definite matrices and Sylvester's criterion," *Amer. Math. Monthly*, vol. 98, no. 1, pp. 44–46, Jan. 1991.

[12] L. Gupta, R. Jain, and H. A. Chan, "Mobile edge computing—An important ingredient of 5G networks," *IEEE Softwarization*, Mar. 2016.

[13] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5G," Eur. Telecommun. Standards Inst., Sophia Antipolis, France, ETSI White Paper 11, Sep. 2015.

[14] *Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor—White Paper*, Intel, Santa Clara, CA, USA, Mar. 2004.

[15] M. A. Khan, "A survey of computation offloading strategies for performance improvement of applications running on mobile devices," *J. Netw. Comput. Appl.*, vol. 56, pp. 28–40, Oct. 2015.

[16] L. Kleinrock, *Queueing Systems. Volume 1: Theory*. New York, NY, USA: Wiley, 1975.

[17] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[18] A. M. Lee and P. A. Longton, "Queueing processes associated with airline passenger check-in," *J. Oper. Res. Soc.*, vol. 10, no. 1, pp. 56–71, Mar. 1959.

[19] K. Li, "Optimal task execution speed setting and lower bound for delay and energy minimization," *J. Parallel Distrib. Comput.*, vol. 123, pp. 13–25, Jan. 2019.

[20] K. Li, "A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing," *IEEE Trans. Sustain. Comput.*, to be published.

[21] K. Li, "Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing," *IEEE Trans. Sustain. Comput.*, to be published.

[22] C. Liu, K. Li, J. Liang, and K. Li, "COOPER-MATCH: Job offloading with a cooperative game for guaranteeing strict deadlines in MEC," *IEEE Trans. Mobile Comput.*, to be published.

[23] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. 18th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst.*, Cancun, Mexico, Nov. 2015, pp. 271–278.

[24] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[25] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Commun. Conf.*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[26] J. B. Rosen, "Existence and uniqueness of equilibrium points for concave N-person games," *Econometrica*, vol. 33, no. 3, pp. 520–534, Jul. 1965.

[27] M. Shiraz, M. Sookhak, A. Gani, and S. A. A. Shah, "A study on the critical analysis of computational offloading frameworks for mobile cloud computing," *J. Netw. Comput. Appl.*, vol. 47, pp. 47–60, Jan. 2015.

[28] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," May 2017, *arXiv:1705.00704*. [Online]. Available: https://arxiv.org/abs/1705.00704

[29] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[30] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proc. 41st Annu. Design Autom. Conf.*, Jun. 2004, pp. 868–873.

[31] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, 2018.

[32] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

KEQIN LI is currently a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor with Hunan University, China. He has published more than 660 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. His current 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 currently serves or has served on the Editorial Boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing.