A Game Theoretic Approach to Computation Offloading Strategy Optimization for Non-Cooperative Users in Mobile Edge Computing

Keqin Li, *Fellow, IEEE*

Abstract—Computation offloading from a *user equipment* (UE, also called mobile user, mobile subscriber, or mobile device) to a *mobile edge cloud* (MEC) provides an effective way to virtualize an ordinary smart mobile device (e.g., smartphone, tablet, handheld computer, wearable device, and personal digital assistant) into a formidable equipment, which is able to provide more and stronger functionalities than that of a laptop or a desktop computer. It is conceivable that there can be several MECs with different processing capabilities in a geographic area, and each MEC may serve many UEs with endless sequences of computation tasks, various application characteristics, and diversified communication requirements and bandwidths. Furthermore, the mobile users are competitive and selfish, which means that computation offloading strategy optimization needs to be carried out for each individual mobile user to optimize the performance of only his applications. In this paper, we conduct a mathematical study of computation offloading strategy optimization for non-cooperative users in mobile edge computing by using a game theoretic approach. The main contributions of this paper can be summarized as follows. We establish an M/G/1 queueing model to characterize multiple heterogeneous UEs and MECs, so that the average response time of all offloadable and non-offloadable tasks generated on a UE can be calculated analytically and the optimal computation offloading strategy of a UE can be defined rigorously. We construct a non-cooperative game framework for a mobile edge computing environment, in which each player (i.e., a UE) can selfishly minimize his payoff by choosing an appropriate strategy in his strategy space. We prove the existence of the Nash equilibrium of the above game. We develop algorithms to find the Nash equilibrium, including an algorithm to find the best response of a mobile user and an iterative algorithm to find the Nash equilibrium. We demonstrate numerical examples and data of our game, including numerical data for the Nash equilibrium and numerical data for the convergence of the Nash equilibrium. To the best of the author's knowledge, this is the first paper that effectively investigates computation offloading strategy optimization for multiple, heterogeneous, and competitive mobile users and multiple heterogeneous mobile edge clouds by using a non-cooperative game approach. Hence, the paper makes noticeable contributions towards the understanding of a competing mobile edge computing environment and its stabilization.

Index Terms—Average response time, computation offloading strategy optimization, mobile edge computing, Nash equilibrium, non-cooperative game, queueing system, user equipment

1 Introduction

1.1 Motivation

The technique of *computation offloading* refers to the transfer of certain computing tasks to an external platform, such as a cluster, a grid, or a cloud. Computation offloading from a *user equipment* (UE, also called mobile user, mobile subscriber, or mobile device) to a *mobile edge cloud* (MEC) provides an effective way to virtualize an ordinary smart mobile device (e.g., smartphone, tablet, handheld computer, wearable device, and personal digital assistant) into a formidable equipment, which is able to provide more and stronger functionalities than that of a laptop or a desktop computer. Furthermore, computation offloading may also be employed to save energy consumption of a mobile device. Due to improved computing capability, increased memory capacity, enhanced database storage, and prolonged battery lifetime, mobile users can run pervasive and powerful applications, such as speech recognition, natural language processing, image processing, face detection and recognition, interactive gaming, reality augmentation, intelligent video acceleration, connected vehicles, and Internet of Things gateway [9]. Therefore, an MEC has the potential to ease the computational burden of mobile devices, to improve the performance of mobile applications, to reduce the energy consumption and to extend the battery lifetime of mobile user equipments.

An MEC provides cloud computing capabilities and service environments at the edge of cellular networks and within the radio access networks in close proximity to mobile subscribers [2]. It is conceivable that there can be several MECs with different processing capabilities in a geographic area, and each MEC many serve many UEs with endless sequences of computation tasks, various

- *The author is with the Department of Computer Science, State University of New York, New Paltz, NY 12561. E-mail: lik@newpaltz.edu.*

Manuscript received 16 May 2018; revised 15 July 2018; accepted 2 Sept. 2018. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Keqin Li.)
Recommended for acceptance by J. Taheri, S. Dustar, and M. Villari.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSUSC.2018.2868655

application characteristics, and diversified communication requirements and bandwidths. Therefore, there are multiple heterogeneous mobile users competing for resources from multiple heterogeneous mobile edge clouds. When a UE makes the decision of computation offloading, which includes the selection of an application to offload and the choice of an MEC to offload the application, the UE should be aware of that fact that the MECs are serving other UEs. To optimize the performance (i.e., the average response time) of a UE's applications, the UE needs to know the current workload of each MEC, so that an optimal computation offloading strategy (i.e., a load distribution method) of the UE can be decided. Furthermore, the mobile users are competitive and selfish, which means that computation offloading strategy optimization needs to be carried out for each individual mobile user to optimize the performance of only his applications. However, computation offloading strategy optimization for non-cooperative users has not been well studied with the above considerations.

1.2 Our Contributions

In this paper, we conduct a mathematical study of computation offloading strategy optimization for non-cooperative users in mobile edge computing by using a game theoretic approach. The main contributions of this paper can be summarized as follows.

- We establish an M/G/1 queueing model to characterize multiple heterogeneous UEs and MECs, so that the average response time of all offloadable and non-offloadable tasks generated on a UE can be calculated analytically and the optimal computation offloading strategy of a UE can be defined rigorously.
- We construct a non-cooperative game framework for a mobile edge computing environment, in which each player (i.e., a UE) can selfishly minimize his payoff by choosing an appropriate strategy in his strategy space. We prove the existence of the Nash equilibrium of the above game.
- We develop algorithms to find the Nash equilibrium, including an algorithm to find the best response of a mobile user and an iterative algorithm to find the Nash equilibrium.
- We demonstrate numerical examples and data of our game, including numerical data for the Nash equilibrium and numerical data for the convergence of the Nash equilibrium.

To the best of the author's knowledge, this is the first paper that effectively investigates computation offloading strategy optimization for multiple, heterogeneous, and competitive mobile users and multiple heterogeneous mobile edge clouds by using a non-cooperative game approach. Hence, the paper makes noticeable contributions towards the understanding of a competing mobile edge computing environment and its stabilization.

The organization of the paper is outlined as follows. In Section 2, we review related research involving multiple mobile users and multiple mobile edge clouds. In Section 3, we provide background information, i.e., queueing models for multiple mobile users and multiple heterogeneous mobile edge computing servers. In Section 4, we formulate a non-cooperative game for mobile users competing for mobile edge computing resources and show the existence of the Nash equilibrium of the game. In Section 5, we develop algorithms to find the Nash equilibrium. In Section 6, we demonstrate numerical examples and data. In Section 7, we conclude the paper.

2 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 [3], [12] for recent comprehensive surveys.

You et al. studied optimal resource allocation for a multiuser 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 [16]. 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 [17]. 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 [13].

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 [5]. 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 [7]. 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 [8]. 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 [11]. 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, 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 [6]. However, the above study employed the M/M/1 queueing model, which is not able to capture the heterogeneity of mobile devices, since the merge of tasks from different mobile users does not yield an exponential distribution anymore. 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 [15]. However, this study did not use the game theoretic approach to dealing with competitive and selfish mobile users.

Our investigation has the following new and unique features.

- We consider 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.
- Each mobile user has an endless sequence of computational tasks, which are classified into offloadable and non-offloadable tasks. Each UE is specified by its own task arrival rates, task execution requirements, data communication requirement, execution speed, and communication speeds. Each MEC is specified by its execution speed.
- We use the game theoretic approach to finding the optimal computation offloading strategy for each mobile user when a mobile computing environment becomes stabilized.

We would like to mention that the purpose of a player in a non-cooperative game is not to defeat other players, but to minimize his own payoff, when other players are also doing so. The purpose of a game is to find a stable situation in which everyone's payoff is minimized in the sense that and no one wants to change anymore. The significance of our research is to show the existence of such a stable situation and to be able to numerically calculate the stable situation.

3 Background Information

To analytically study computation offloading strategy optimization for non-cooperative mobile users competing for resources from multiple heterogeneous mobile edge clouds, we need to establish mathematical models. In this section, we present queueing models for multiple mobile users and multiple heterogeneous mobile edge computing servers. Throughout the paper, we use $\overline{x}$ to represent the expectation of a random variable x. Table 1 gives a list of the symbols and their definitions in this paper.

We consider a mobile edge computing environment with multiple UEs and multiple MECs (see Fig. 1). Assume that there are n mobile user equipments, i.e., UE_1, $\mathrm{UE}_2, \ldots,$ UE_n. There are also m mobile edge clouds, i.e., MEC_1, $\mathrm{MEC}_2, \ldots, \mathrm{MEC}_m$.

In this paper, a UE is treated as an M/G/1 queueing system. 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). Thus, the UE is actually a server. There is a Poisson stream of computation tasks with arrival rate $\hat{\lambda}_i + \lambda_i$ (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 $1/(\hat{\lambda}_i + \lambda_i)$. The arrival task stream is decomposed into two streams, that is, there is a Poisson stream of non-offloadable computation tasks with arrival rate $\hat{\lambda}_i$, and there is a Poisson stream of offloadable computation tasks with arrival rate λ_i. All non-offloadable tasks are processed locally in UE_i. The stream of offloadable computation tasks is further divided into $m+1$ substreams with arrival rates $\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}$ respectively, where $\lambda_i = \lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m}$, such that the substream with arrival rate $\lambda_{i,0}$ is processed locally in UE_i, while the substream with arrival rate $\lambda_{i,j}$ is offloaded to MEC_j and processed remotely in MEC_j, for all $1 \le j \le m$. The vector $\boldsymbol{\lambda}_i = (\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m})$ is actually a *computation offloading strategy* of UE_i, for all $1 \le i \le n$.

Each MEC is also treated as an M/G/1 queueing system. Thus, an MEC is actually a server. There is a Poisson stream of computation tasks with arrival rate $\tilde{\lambda}_j$ to MEC_j, where $\tilde{\lambda}_j = \lambda_{1,j} + \lambda_{2,j} + \cdots + \lambda_{n,j}$, for all $1 \le j \le m$.

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

The execution requirements (measured by the number of processor cycles or the number of billion instructions (BI) to be executed) of the non-offloadable computation tasks generated on UE_i are i.i.d. random variables $\hat{r}_i$ with an arbitrary probability distribution. We assume that its mean $\overline{\hat{r}_i}$ and second moment $\overline{\hat{r}_i^2}$ are available. The execution requirements of the offloadable computation tasks generated on UE_i are i.i.d. random variables r_i with an arbitrary probability distribution. We assume that its mean $\overline{r_i}$ and second moment $\overline{r_i^2}$ are available.

The amount of data (measured by the number of million bits (MB)) to be communicated between UE_i and the MECs for offloadable tasks are i.i.d. random variables d_i with an arbitrary probability distribution. We assume that its mean $\overline{d_i}$ and second moment $\overline{d_i^2}$ are available.

UE_i has execution speed s_i (measured by GHz or the number of billion instructions that can be executed in one second), where $1 \le i \le n$. MEC_j has execution speed $\tilde{s}_j$,

TABLE 1
Summary of Notations and Definitions

Notation	Definition
	Queueing Theory
n	the number of mobile user equipments
UE_i	the ith user equipment, $1 \le i \le n$
m	the number of mobile edge clouds
MEC_j	the jth mobile edge cloud, $1 \le j \le m$
$\hat{\lambda}_i$	the arrival rate of non-offloadable computation tasks to UE_i
λ_i	the arrival rate of offloadable computation tasks to UE_i, $= \lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m}$
$\lambda_{i,0}$	the arrival rate of the substream of tasks processed locally in UE_i
$\lambda_{i,j}$	the arrival rate of the substream of tasks processed remotely in MEC_j
$\boldsymbol{\lambda}_i$	$= (\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m})$, a computation offloading strategy of UE_i
$\tilde{\lambda}_j$	$= \lambda_{1,j} + \lambda_{2,j} + \cdots + \lambda_{n,j}$
$\hat{r}_i$	the execution requirements of the non-offloadable computation tasks generated on UE_i
$\overline{\hat{r}_i}$, $\overline{\hat{r}_i^2}$	mean and second moment of $\hat{r}_i$
r_i	the execution requirements of the offloadable 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 MECs for offloadable tasks
$\overline{d_i}$, $\overline{d_i^2}$	mean and second moment of d_i
s_i	the execution speed of UE_i
$\tilde{s}_j$	the execution speed of MEC_j
$c_{i,j}$	the communication speed between UE_i and MEC_j
x_i	the execution times of all tasks on UE_i
$\overline{x_i}$, $\overline{x_i^2}$	mean and second moment of x_i
ρ_i	the utilization of the server in UE_i
W_i	the average waiting time of the tasks on UE_i
$T_{i,0}$	the average response time of the tasks on UE_i
$\tilde{x}_j$	the execution times of all tasks on MEC_j
$\overline{\tilde{x}_j}$, $\overline{\tilde{x}_j^2}$	mean and second moment of $\tilde{x}_j$
$\tilde{\rho}_j$	the utilization of the server in MEC_j
$\tilde{W}_j$	the average waiting time of the tasks on MEC_j
$T_{i,j}$	the average response time of the tasks offloaded from UE_i to MEC_j
T_i	the average response time of all offloadable and non-offloadable tasks generated on UE_i
	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
K_i	$= \{(\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}) \mid \lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m} = \lambda_i\}$
$\boldsymbol{\lambda}_i$	$= (\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}) \in K_i \subseteq \mathbb{R}^{m+1}$
$T_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$	the payoff function of UE_i
$\mathbf{T}$	$= (T_1(\boldsymbol{\lambda}), T_2(\boldsymbol{\lambda}), \ldots, T_n(\boldsymbol{\lambda}))$
	Algorithm Theory
$\text{CO}(K, f)$	a convex optimization problem
ϕ	a Lagrange multiplier
I	the maximum length of all initial search intervals
ϵ	the accuracy requirement
K	the number of rounds

where $1 \le j \le m$. The communication speed (measured by the number of million bits that can be communicated in one second) between UE_i and MEC_j is $c_{i,j}$, where $1 \le i \le n$ and $1 \le j \le m$.

4 A NON-COOPERATIVE GAME

In this section, we present preliminaries from non-cooperative game theory, describe a game formulation for non-cooperative mobile users competing for mobile edge

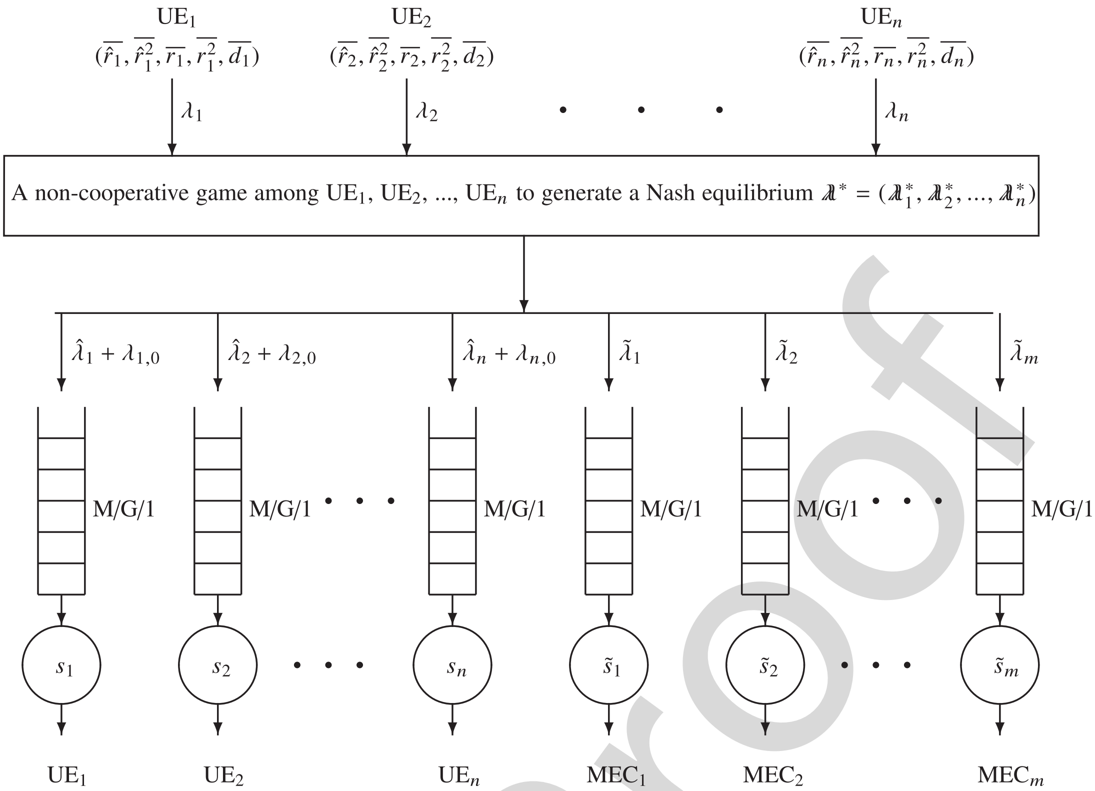

Fig. 1. A mobile edge computing environment with multiple UEs and multiple MECs.

computing resources, and show the existence of the Nash equilibrium of the game.

4.1 Preliminaries

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.,

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

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

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

The following result is well-known [1].

Theorem 1. *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.

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

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

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

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

4.2 Non-Cooperative Games

In this section, we present preliminaries from non-cooperative game theory.

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 $1 \leq i \leq n$. 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

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

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 $1 \leq i \leq n$. 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 $1 \leq i \leq n$. 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

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

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.

The following classic result is from [14].

Theorem 2. *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 $\mathcal{G} = (\mathcal{K}, \mathbf{f})$.*

4.3 A Game Formulation

In this section, we describe a game formulation for non-cooperative mobile users competing for mobile edge computing resources.

Let UE_1, UE_2, . . . , UE_n be the n players in a non-cooperative game. The set of strategies of UE_i is $\boldsymbol{\lambda}_i = (\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}) \in K_i \subseteq \mathbb{R}^{m+1}$, where

$$K_i = \{(\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}) \mid \lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m} = \lambda_i\},$$

which is a convex set, for all $1 \le i \le n$. Let $\mathcal{K} = K_1 \times K_2 \times \cdots \times K_n$ and $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_n) \in \mathcal{K}$. The payoff function of UE_i is the average response time T_i of all tasks generated on UE_i, i.e., $T_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}) : \mathcal{K} \to \mathbb{R}$, which is given in Theorem 3.

Let $\mathbf{T} = (T_1(\boldsymbol{\lambda}), T_2(\boldsymbol{\lambda}), \ldots, T_n(\boldsymbol{\lambda}))$. Then, our non-cooperative game is $\mathcal{G} = (\mathcal{K}, \mathbf{T})$.

The following theorem gives the average response time of all tasks (offloadable and non-offloadable) generated on a UE. This is the main performance metric in mobile edge computing.

Theorem 3. *The average response time of all offloadable and non-offloadable tasks generated on UE_i is*

$$\begin{aligned} T_i = \frac{\hat{\lambda}_i + \lambda_{i,0}}{\hat{\lambda}_i + \lambda_i} \Bigg(& \frac{\hat{\lambda}_i}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{\hat{r}_i}}{s_i} + \frac{\lambda_{i,0}}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{r_i}}{s_i} \\ & + \frac{\hat{\lambda}_i (\overline{\hat{r}_i^2}/s_i^2) + \lambda_{i,0}(\overline{r_i^2}/s_i^2)}{2(1 - (\hat{\lambda}_i(\overline{\hat{r}_i}/s_i) + \lambda_{i,0}(\overline{r_i}/s_i)))} \Bigg) \\ & + \sum_{j=1}^{m} \frac{\lambda_{i,j}}{\hat{\lambda}_i + \lambda_i} \Bigg(\left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}} \right) \\ & + \frac{\sum_{i'=1}^{n} \lambda_{i',j} \left(\overline{r_{i'}^2}/\tilde{s}_j^2 + 2\overline{r_{i'}}\,\overline{d_{i'}}/(\tilde{s}_j c_{i',j}) + \overline{d_{i'}^2}/c_{i',j}^2 \right)}{2\left(1 - \sum_{i'=1}^{n} \lambda_{i',j}(\overline{r_{i'}}/\tilde{s}_j + \overline{d_{i'}}/c_{i',j})\right)} \Bigg), \end{aligned}$$

for all $1 \le i \le n$.

Proof. Based on the queueing model for the UEs in Section 3, we know that the execution times of non-offloadable tasks on UE_i are i.i.d. random variables with mean $\overline{\hat{r}_i}/s_i$ and second moment $\overline{\hat{r}_i^2}/s_i^2$, and that the execution times of offloadable tasks on UE_i are i.i.d. random variables with mean $\overline{r_i}/s_i$ and second moment $\overline{r_i^2}/s_i^2$. Therefore, the execution times of all tasks on UE_i are i.i.d. random variables x_i with mean

$$\overline{x_i} = \frac{\hat{\lambda}_i}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{\hat{r}_i}}{s_i} + \frac{\lambda_{i,0}}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{r_i}}{s_i},$$

and second moment

$$\overline{x_i^2} = \frac{\hat{\lambda}_i}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{\hat{r}_i^2}}{s_i^2} + \frac{\lambda_{i,0}}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{r_i^2}}{s_i^2},$$

where we notice that $\hat{\lambda}_i/(\hat{\lambda}_i + \lambda_{i,0})$ is the percentage of non-offloadable tasks on UE_i, while $\lambda_{i,0}/(\hat{\lambda}_i + \lambda_{i,0})$ is the percentage of offloadable tasks on UE_i. The utilization of the server in UE_i is

$$\rho_i = (\hat{\lambda}_i + \lambda_{i,0})\overline{x_i} = \hat{\lambda}_i \frac{\overline{\hat{r}_i}}{s_i} + \lambda_{i,0} \frac{\overline{r_i}}{s_i}.$$

The average waiting time of the tasks on UE_i is ([10], p. 190)

$$W_i = \frac{(\hat{\lambda}_i + \lambda_{i,0})\overline{x_i^2}}{2(1 - \rho_i)},$$

where

$$(\hat{\lambda}_i + \lambda_{i,0})\overline{x_i^2} = \hat{\lambda}_i \frac{\overline{\hat{r}_i^2}}{s_i^2} + \lambda_{i,0} \frac{\overline{r_i^2}}{s_i^2}.$$

The average response time of the tasks on UE_i is

$$\begin{aligned} T_{i,0} &= \overline{x_i} + W_i \\ &= \overline{x_i} + \frac{(\hat{\lambda}_i + \lambda_{i,0})\overline{x_i^2}}{2(1 - \rho_i)} \\ &= \frac{\hat{\lambda}_i}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{\hat{r}_i}}{s_i} + \frac{\lambda_{i,0}}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{r_i}}{s_i} \\ &\quad + \frac{\hat{\lambda}_i(\overline{\hat{r}_i^2}/s_i^2) + \lambda_{i,0}(\overline{r_i^2}/s_i^2)}{2(1 - (\hat{\lambda}_i(\overline{\hat{r}_i}/s_i) + \lambda_{i,0}(\overline{r_i}/s_i)))}. \end{aligned}$$

Furthermore, based on the queueing model for the MECs in Section 3, we know that for MEC_j, where $1 \le j \le m$, the execution times of the tasks offloaded from $\mathrm{UE}_{i'}$ are i.i.d. random variables $r_{i'}/\tilde{s}_j + d_{i'}/c_{i',j}$, where $r_{i'}/\tilde{s}_j$ is the computation time and $d_{i'}/c_{i',j}$ is the communication time. These random variables have mean

$$\overline{r_{i'}}/\tilde{s}_j + \overline{d_{i'}}/c_{i',j},$$

and second moment

$$\overline{r_{i'}^2}/\tilde{s}_j^2 + 2\overline{r_{i'}}\,\overline{d_{i'}}/(\tilde{s}_j c_{i',j}) + \overline{d_{i'}^2}/c_{i',j}^2.$$

Therefore, the execution times of all tasks on MEC_j are i.i.d. random variables $\tilde{x}_j$ with mean

$$\overline{\tilde{x}_j} = \sum_{i'=1}^{n} \frac{\lambda_{i',j}}{\tilde{\lambda}_j} \left(\frac{\overline{r_{i'}}}{\tilde{s}_j} + \frac{\overline{d_{i'}}}{c_{i',j}} \right),$$

and second moment

$$\overline{\tilde{x}_j^2} = \sum_{i'=1}^{n} \frac{\lambda_{i',j}}{\tilde{\lambda}_j} \left(\frac{\overline{r_{i'}^2}}{\tilde{s}_j^2} + 2\frac{\overline{r_{i'}}\,\overline{d_{i'}}}{\tilde{s}_j c_{i',j}} + \frac{\overline{d_{i'}^2}}{c_{i',j}^2} \right),$$

where we notice that $\lambda_{i',j}/\tilde{\lambda}_j$ is the percentage of tasks offloaded from $\mathrm{UE}_{i'}$, for all $1 \le i' \le n$. The utilization of the server in MEC_j is

$$\tilde{\rho}_j = \tilde{\lambda}_j \overline{\tilde{x}_j} = \sum_{i'=1}^{n} \lambda_{i',j} \left(\frac{\overline{r_{i'}}}{\tilde{s}_j} + \frac{\overline{d_{i'}}}{c_{i',j}} \right).$$

The average waiting time of the tasks on MEC_j is

$$\tilde{W}_j = \frac{\tilde{\lambda}_j \overline{\tilde{x}_j^2}}{2(1 - \tilde{\rho}_j)},$$

where

$$\tilde{\lambda}_j \overline{\tilde{x}_j^2} = \sum_{i'=1}^{n} \lambda_{i',j} \left(\frac{\overline{r_{i'}^2}}{\tilde{s}_j^2} + 2\frac{\overline{r_{i'}}\,\overline{d_{i'}}}{\tilde{s}_j c_{i',j}} + \frac{\overline{d_{i'}^2}}{c_{i',j}^2} \right).$$

The average response time of the tasks offloaded from UE_i to MEC_j is

$$
\begin{aligned}
T_{i,j} &= \left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}}\right) + \tilde{W}_j \\
&= \left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}}\right) + \frac{\tilde{\lambda}_j \overline{\tilde{x}_j^2}}{2(1-\tilde{\rho}_j)} \\
&= \left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}}\right) \\
&\quad + \frac{\sum_{i'=1}^{n} \lambda_{i',j}\Big(\overline{r_{i'}^2}/\tilde{s}_j^2 + 2\overline{r_{i'}}\,\overline{d_{i'}}/(\tilde{s}_j c_{i',j}) + \overline{d_{i'}^2}/c_{i',j}^2\Big)}{2\Big(1 - \sum_{i'=1}^{n} \lambda_{i',j}(\overline{r_{i'}}/\tilde{s}_j + \overline{d_{i'}}/c_{i',j})\Big)},
\end{aligned}
$$

for all $1 \leq j \leq m$.

Finally, the *average response time* of all offloadable and non-offloadable tasks generated on UE_i is

$$
T_i = \frac{\hat{\lambda}_i + \lambda_{i,0}}{\hat{\lambda}_i + \lambda_i} T_{i,0} + \sum_{j=1}^{m} \frac{\lambda_{i,j}}{\hat{\lambda}_i + \lambda_i} T_{i,j},
$$

which leads to the equation in the theorem by substituting all the $T_{i,j}$'s into the last equation, where $0 \leq j \leq m$. This proves the theorem. □

4.4 Existence of the Nash Equilibrium

We now show the existence of the Nash equilibrium of the above game.

Theorem 4. *$T_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ is a convex function of $\boldsymbol{\lambda}_i$ for each fixed $\boldsymbol{\lambda}_{-i}$, for all $1 \leq i \leq n$. Hence, there is a Nash equilibrium for the non-cooperative game $\mathcal{G} = (\mathcal{K}, \mathbf{T})$.*

Proof. From Theorem 1, we have

$$
\begin{aligned}
\frac{\partial T_i}{\partial \lambda_{i,0}} &= \frac{1}{\hat{\lambda}_i + \lambda_i}\Bigg(\frac{\hat{\lambda}_i}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{\hat{r}_i}}{s_i} + \frac{\lambda_{i,0}}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{r_i}}{s_i} \\
&\quad + \frac{\hat{\lambda}_i(\overline{\hat{r}_i^2}/s_i^2) + \lambda_{i,0}(\overline{r_i^2}/s_i^2)}{2(1 - (\hat{\lambda}_i(\overline{\hat{r}_i}/s_i) + \lambda_{i,0}(\overline{r_i}/s_i)))}\Bigg) \\
&\quad + \frac{\hat{\lambda}_i + \lambda_{i,0}}{\hat{\lambda}_i + \lambda_i}\Bigg(-\frac{\hat{\lambda}_i}{(\hat{\lambda}_i + \lambda_{i,0})^2} \cdot \frac{\overline{\hat{r}_i}}{s_i} + \frac{\hat{\lambda}_i}{\hat{\lambda}_i + \lambda_{i,0}} \cdot \frac{\overline{r_i}}{s_i} \\
&\quad + \frac{\overline{r_i^2}/s_i^2}{2(1 - (\hat{\lambda}_i(\overline{\hat{r}_i}/s_i) + \lambda_{i,0}(\overline{r_i}/s_i)))} \\
&\quad + \frac{(\overline{r_i}/s_i)(\hat{\lambda}_i(\overline{\hat{r}_i^2}/s_i^2) + \lambda_{i,0}(\overline{r_i^2}/s_i^2))}{2(1 - (\hat{\lambda}_i(\overline{\hat{r}_i}/s_i) + \lambda_{i,0}(\overline{r_i}/s_i)))^2}\Bigg) \\
&= \frac{1}{\hat{\lambda}_i + \lambda_i}\Bigg(\left(\frac{\lambda_{i,0}}{\hat{\lambda}_i + \lambda_{i,0}} + \hat{\lambda}_i\right) \cdot \frac{\overline{r_i}}{s_i} \\
&\quad + \frac{\hat{\lambda}_i(\overline{\hat{r}_i^2}/s_i^2) + \lambda_{i,0}(\overline{r_i^2}/s_i^2)}{2(1 - (\hat{\lambda}_i(\overline{\hat{r}_i}/s_i) + \lambda_{i,0}(\overline{r_i}/s_i)))}\Bigg) \\
&\quad + \frac{\hat{\lambda}_i + \lambda_{i,0}}{\hat{\lambda}_i + \lambda_i}\Bigg(\frac{\overline{r_i^2}/s_i^2}{2(1 - (\hat{\lambda}_i(\overline{\hat{r}_i}/s_i) + \lambda_{i,0}(\overline{r_i}/s_i)))} \\
&\quad + \frac{(\overline{r_i}/s_i)(\hat{\lambda}_i(\overline{\hat{r}_i^2}/s_i^2) + \lambda_{i,0}(\overline{r_i^2}/s_i^2))}{2(1 - (\hat{\lambda}_i(\overline{\hat{r}_i}/s_i) + \lambda_{i,0}(\overline{r_i}/s_i)))^2}\Bigg),
\end{aligned}
$$

and

$$
\begin{aligned}
\frac{\partial T_i}{\partial \lambda_{i,j}} &= \frac{1}{\hat{\lambda}_i + \lambda_i}\Bigg(\left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}}\right) \\
&\quad + \frac{\sum_{i'=1}^{n} \lambda_{i',j}\Big(\overline{r_{i'}^2}/\tilde{s}_j^2 + 2\overline{r_{i'}}\,\overline{d_{i'}}/(\tilde{s}_j c_{i',j}) + \overline{d_{i'}^2}/c_{i',j}^2\Big)}{2\Big(1 - \sum_{i'=1}^{n} \lambda_{i',j}(\overline{r_{i'}}/\tilde{s}_j + \overline{d_{i'}}/c_{i',j})\Big)}\Bigg) \\
&\quad + \frac{\lambda_{i,j}}{\hat{\lambda}_i + \lambda_i}\Bigg(\frac{\overline{r_i^2}/\tilde{s}_j^2 + 2\overline{r_i}\,\overline{d_i}/(\tilde{s}_j c_{i,j}) + \overline{d_i^2}/c_{i,j}^2}{2\Big(1 - \sum_{i'=1}^{n} \lambda_{i',j}(\overline{r_{i'}}/\tilde{s}_j + \overline{d_{i'}}/c_{i',j})\Big)} \\
&\quad + \frac{(\overline{r_i}/\tilde{s}_j + \overline{d_i}/c_{i,j})}{2\Big(1 - \sum_{i'=1}^{n} \lambda_{i',j}(\overline{r_{i'}}/\tilde{s}_j + \overline{d_{i'}}/c_{i',j})\Big)^2} \\
&\quad \times \sum_{i'=1}^{n} \lambda_{i',j}\Big(\overline{r_{i'}^2}/\tilde{s}_j^2 + 2\overline{r_{i'}}\,\overline{d_{i'}}/(\tilde{s}_j c_{i',j}) + \overline{d_{i'}^2}/c_{i',j}^2\Big)\Bigg),
\end{aligned}
$$

for all $1 \leq j \leq m$. We can easily verify by straightforward algebraic manipulation that

$$
\frac{\partial^2 T_i}{\partial \lambda_{i,j}^2} > 0,
$$

for all $0 \leq j \leq m$, and

$$
\frac{\partial^2 T_i}{\partial \lambda_{i,j} \partial \lambda_{i,k}} = 0,
$$

for all $0 \leq j \neq k \leq m$. In fact, it is easily seen that $\partial T_i / \partial \lambda_{i,j}$ is an increasing function of $\lambda_{i,j}$, and $\partial^2 T_i / \partial \lambda_{i,j}^2 > 0$ for all $0 \leq j \leq m$. Therefore, the Hessian matrix

$$
\mathbf{H}(T_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})) = \left[\frac{\partial^2 T_i}{\partial \lambda_{i,j} \partial \lambda_{i,k}}\right]_{(m+1)\times(m+1)},
$$

of second partial derivatives is a diagonal matrix, in which each element on the main diagonal is positive. That is, the Hessian matrix is positive definite on the interior of K_i. By Theorem 1, $T_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i})$ is a convex function of $\boldsymbol{\lambda}_i$ for each fixed $\boldsymbol{\lambda}_{-i}$, for all $1 \leq i \leq n$. By Theorem 2, there is a Nash equilibrium for the non-cooperative game $\mathcal{G} = (\mathcal{K}, \mathbf{T})$. □

5 The Algorithms

In this section, we develop an algorithm to find the best response of a mobile user and an iterative algorithm to find the Nash equilibrium.

5.1 The Best Response of a Mobile User

In this section, we develop an algorithm to find the best response of a mobile user. The algorithm essentially solves the convex optimization problems $\mathrm{CO}(K_i, T_i)$, which is defined as follows: given n user equipments $\mathrm{UE}_1, \mathrm{UE}_2, \ldots, \mathrm{UE}_n$, where UE_i is specified by the parameters $\hat{\lambda}_i, \lambda_i, \overline{\hat{r}_i}, \overline{\hat{r}_i^2}, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, s_i$, and $c_{i,j}$, for all $1 \leq i \leq n$ and $1 \leq j \leq m$, m MECs specified by the parameters $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_m$, and $\boldsymbol{\lambda}_{i'} = (\lambda_{i',0}, \lambda_{i',1}, \ldots, \lambda_{i',m})$, for all $1 \leq i' \neq i \leq n$, find $\boldsymbol{\lambda}_i = (\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m})$, such that T_i is minimized, subject to the constraint that $\lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m} = \lambda_i$.

The above convex optimization problem to minimize the average response time T_i can be solved by using the method of Lagrange multiplier, namely,

$$\nabla T_i(\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}) = \phi \nabla F(\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}),$$

where

$$F(\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}) = \lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m} = \lambda_i,$$

that is,

$$\frac{\partial T_i}{\partial \lambda_{i,j}} = \phi \frac{\partial F}{\partial \lambda_{i,j}} = \phi,$$

for all $0 \le j \le m$, where ϕ is a Lagrange multiplier. Notice that $\partial T_i / \partial \lambda_{i,0}$ and $\partial T_i / \partial \lambda_{i,j}$ for all $1 \le j \le m$ have already been derived in the proof of Theorem 4.

First, for a given ϕ, our numerical algorithm to find $\lambda_{i,j}$ such that $\partial T_i / \partial \lambda_{i,j} = \phi$ is given in Algorithm 1. The algorithm uses the classical bisection method (lines 2-10) based on the observation that $\partial T_i / \partial \lambda_{i,j}$ is an increasing function of $\lambda_{i,j}$ (lines 5-9). (The standard bisection method is described in [4], p. 22). The initial search interval in line 1 is $[0, ub]$, where ub is obtained as follows. If $j = 0$, we need $\rho_i < 1$, i.e.,

$$ub = \left(1 - \hat{\lambda}_i \frac{\overline{\hat{r}_i}}{s_i}\right) \frac{s_i}{\overline{r_i}}.$$

If $j \ne 0$, we need $\tilde{\rho}_j < 1$, i.e.,

$$ub = \left(1 - \sum_{i' \ne i} \lambda_{i',j} \left(\frac{\overline{r_{i'}}}{\tilde{s}_j} + \frac{\overline{d_{i'}}}{c_{i',j}}\right)\right) \Bigg/ \left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}}\right).$$

The algorithm terminates when the search interval is shorter than ϵ. We set $\epsilon = 10^{-11}$ in this paper. Let I denote the maximum length of all initial search intervals in this paper. Then, the time complexity of Algorithm 1 is $O(\log(I/\epsilon))$.

Algorithm 1. Find $\lambda_{i,j}$

Input: $\hat{\lambda}_i, \lambda_i, \overline{\hat{r}_i}, \overline{\hat{r}_i^2}, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, s_i$, and $c_{i,j}$, for all $1 \le i \le n$, $\tilde{s}_j$, $\lambda_{i',j}$, for all $1 \le i' \ne i \le n$, and ϕ.
Output: $\lambda_{i,j}$ such that $\partial T_i / \partial \lambda_{i,j} = \phi$.

Initialize the search interval of $\lambda_{i,j}$; (1)
while (the length of the search interval is $\ge \epsilon$) **do** (2)
  $\lambda_{i,j} \leftarrow$ the middle point of the search interval; (3)
  Calculate $\partial T_i / \partial \lambda_{i,j}$; (4)
  if ($\partial T_i / \partial \lambda_{i,j} < \phi$) **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)
$\lambda_{i,j} \leftarrow$ the middle point of the search interval; (11)
return $\lambda_{i,j}$. (12)

Second, for a given λ_i, our numerical algorithm to find ϕ and $\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}$, such that $\partial T_i / \partial \lambda_{i,j} = \phi$, for all $0 \le j \le m$, and $\lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m} = \lambda_i$, is given in Algorithm 2. Again, the algorithm uses the classical bisection method (lines 2-12) based on the observation that $\lambda_{i,j}$ is an increasing function ϕ, and thus $\lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m}$ is also an increasing function ϕ (lines 7-11). The initial search interval in line 1 is $[0, ub]$, where ub is sufficiently large. Due the nested loops and the calling of Algorithm 1, the time complexity of Algorithm 2 is $O(m(\log(I/\epsilon))^2)$.

Algorithm 2. Find ϕ and $\boldsymbol{\lambda}_i = (\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m})$

Input: $\hat{\lambda}_i, \lambda_i, \overline{\hat{r}_i}, \overline{\hat{r}_i^2}, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, s_i$, and $c_{i,j}$, for all $1 \le i \le n$ and $1 \le j \le m$, $\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_m$, and $\boldsymbol{\lambda}_{i'} = (\lambda_{i',0}, \lambda_{i',1}, \ldots, \lambda_{i',m})$, for all $1 \le i' \ne i \le n$.
Output: ϕ and $\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}$, such that $\partial T_i / \partial \lambda_{i,j} = \phi$, for all $0 \le j \le m$, and $\lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m} = \lambda_i$.

Initialize the search interval of ϕ; (1)
while (the length of the search interval is $\ge \epsilon$) **do** (2)
  $\phi \leftarrow$ the middle point of the search interval; (3)
  for $j \leftarrow 0$ **to** m **do** (4)
    Find $\lambda_{i,j}$ s.t. $\partial T_i / \partial \lambda_{i,j} = \phi$ using Algorithm 1; (5)
  end do; (6)
  if ($\lambda_{i,0} + \lambda_{i,1} + \cdots + \lambda_{i,m} < \lambda_i$) **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)
$\phi \leftarrow$ the middle point of the search interval; (13)
for $j \leftarrow 0$ **to** m **do** (14)
  Find $\lambda_{i,j}$ s.t. $\partial T_i / \partial \lambda_{i,j} = \phi$ using Algorithm 1; (15)
end do; (16)
return ϕ and $\lambda_{i,0}, \lambda_{i,1}, \ldots, \lambda_{i,m}$. (17)

5.2 An Iterative Algorithm for Nash Equilibrium

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

Algorithm 3 runs in rounds (lines 2-14). The initial strategy of UE$_i$ is $\boldsymbol{\lambda}_i = (\lambda_i/(m+1), \lambda_i/(m+1), \ldots, \lambda_i/(m+1))$, i.e., an even distribution of offloadable tasks. In each round, every mobile user finds his best response to the current situation by using Algorithm 2 (lines 3-6). The algorithm terminates when the action profiles of two successive rounds are close enough (lines 8-13). The final converged action profile $\boldsymbol{\lambda}^* = (\boldsymbol{\lambda}_1^*, \boldsymbol{\lambda}_2^*, \ldots, \boldsymbol{\lambda}_n^*)$ is returned as the Nash equilibrium, i.e., a strategy profile $\boldsymbol{\lambda}^*$ with the property that no single UE can benefit from a unilateral deviation from $\boldsymbol{\lambda}_i^*$, if all the other UEs act according to it.

The termination detection condition in line 8 is

$$\|\boldsymbol{\lambda}' - \boldsymbol{\lambda}\| = \sqrt{\sum_{i=1}^{n} \sum_{j=0}^{m} |\lambda'_{i,j} - \lambda_{i,j}|^2}.$$

Since Algorithm 2 is invoked n times in each round, the time complexity of each round is $O(mn(\log(I/\epsilon))^2)$, and the overall time complexity of Algorithm 3 is $O(Kmn(\log(I/\epsilon))^2)$, where K is the number of rounds, which is mainly determined by the accuracy requirement ϵ in line 8.

We would like to mention that the essence of a non-cooperative game is not to keep the information and decision of each player confidential and secret to other players, but to emphasize that the mechanism of a game is individual decision making by each player for the benefit of himself, not by group decision making for the benefit of all. Therefore, it really does not matter to assume that all information of all players are

TABLE 2
Numerical Data for the Nash Equilibrium of $n = 10$ UEs and $m = 7$ MECs

i	$\hat{\lambda}_i$	λ_i	$\lambda^*_{i,0}$	$\lambda^*_{i,1}$	$\lambda^*_{i,2}$	$\lambda^*_{i,3}$	$\lambda^*_{i,4}$	$\lambda^*_{i,5}$	$\lambda^*_{i,6}$	$\lambda^*_{i,7}$	ρ_i	T_i
1	1.0000000	1.0000000	0.3320365	0.0842168	0.0879464	0.0916793	0.0954157	0.0991560	0.1029004	0.1066489	0.6653698	2.3215007
2	1.0500000	1.1000000	0.3433942	0.0964103	0.1002965	0.1041862	0.1080793	0.1119761	0.1158766	0.1197808	0.6936007	2.4377323
3	1.1000000	1.2000000	0.3515815	0.1089770	0.1130466	0.1171195	0.1211959	0.1252756	0.1293587	0.1334452	0.7191355	2.5521766
4	1.1500000	1.3000000	0.3565893	0.1219269	0.1262037	0.1304836	0.1347668	0.1390530	0.1433423	0.1476345	0.7421513	2.6651800
5	1.2000000	1.4000000	0.3584739	0.1352592	0.1397646	0.1442729	0.1487840	0.1532978	0.1578143	0.1623333	0.7628451	2.7768688
6	1.2500000	1.5000000	0.3573406	0.1489648	0.1537183	0.1584743	0.1632326	0.1679931	0.1727558	0.1775205	0.7814230	2.8872446
7	1.3000000	1.6000000	0.3533280	0.1630293	0.1680486	0.1730697	0.1780927	0.1831173	0.1881434	0.1931710	0.7980906	2.9962470
8	1.3500000	1.7000000	0.3465952	0.1774344	0.1827357	0.1880381	0.1933416	0.1986461	0.2039514	0.2092574	0.8130460	3.1037932
9	1.4000000	1.8000000	0.3373113	0.1921601	0.1977582	0.2033566	0.2089553	0.2145540	0.2201528	0.2257517	0.8264746	3.2098015
10	1.4500000	1.9000000	0.3256475	0.2071855	0.2130940	0.2190019	0.2249091	0.2308154	0.2367209	0.2426256	0.8385469	3.3142036
$\tilde{\lambda}_j$				1.4355643	1.4826125	1.5296822	1.5767729	1.6238845	1.6710166	1.7181689		
$\tilde{\rho}_j$				0.9063518	0.9079462	0.9094645	0.9109124	0.9122952	0.9136175	0.9148837		

TABLE 3
Numerical Data for the Convergence of the Nash Equilibrium

K	$\lambda_{5,0}$	$\lambda_{5,1}$	$\lambda_{5,2}$	$\lambda_{5,3}$	$\lambda_{5,4}$	$\lambda_{5,5}$	$\lambda_{5,6}$	$\lambda_{5,7}$
5	0.3573805	0.1273049	0.1360352	0.1441961	0.1509310	0.1564997	0.1615311	0.1661216
10	0.3582279	0.1227830	0.1307274	0.1391527	0.1481593	0.1575079	0.1669712	0.1764706
15	0.3584566	0.1342902	0.1382419	0.1425437	0.1475742	0.1532955	0.1594842	0.1661137
20	0.3585235	0.1392043	0.1423044	0.1453763	0.1485364	0.1518463	0.1552971	0.1589118
25	0.3585083	0.1376159	0.1416725	0.1455403	0.1491555	0.1525715	0.1558671	0.1590690
30	0.3584757	0.1350795	0.1398951	0.1446125	0.1491404	0.1534726	0.1576494	0.1616748
35	0.3584606	0.1343052	0.1391196	0.1439734	0.1488337	0.1536665	0.1584559	0.1631851
40	0.3584657	0.1348059	0.1393431	0.1439509	0.1486489	0.1534230	0.1582489	0.1631137
45	0.3584742	0.1353626	0.1397672	0.1441965	0.1486778	0.1532207	0.1578208	0.1624802
50	0.3584771	0.1354827	0.1399269	0.1443561	0.1487788	0.1532076	0.1576516	0.1621192

available to each player. Furthermore, an iterative algorithm to find the Nash equilibrium can be implemented in either centralized or distributed ways. In a distributed implementation (i.e., lines 4-5 of Algorithm 3 are adapted and executed by all mobile users simultaneously and independently), each round should be synchronized (i.e., lines 7-13 should be performed by a coordinator).

Algorithm 3. Calculate the Nash Equilibrium

Input: $\hat{\lambda}_i, \lambda_i, \overline{\hat{r}_i}, \overline{\hat{r}_i^2}, \overline{r_i}, \overline{r_i^2}, \overline{d_i}, \overline{d_i^2}, s_i$, and $c_{i,j}$, for all $1 \le i \le n$ and $1 \le j \le m$, $\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_m$.
Output: The Nash equilibrium $\boldsymbol{\lambda}^* = (\boldsymbol{\lambda}^*_1, \boldsymbol{\lambda}^*_2, \dots, \boldsymbol{\lambda}^*_n)$.

Initialize $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots, \boldsymbol{\lambda}_n)$; (1)
repeat (2)
  for $i \leftarrow 1$ **to** n **do** (3)
    Obtain $\boldsymbol{\lambda}'_i$ by using Algorithm 2 (4)
    with parameters $\boldsymbol{\lambda}'_1, \dots, \boldsymbol{\lambda}'_{i-1}, \boldsymbol{\lambda}_{i+1}, \dots, \boldsymbol{\lambda}_n$; (5)
  end do; (6)
  $\boldsymbol{\lambda}' \leftarrow (\boldsymbol{\lambda}'_1, \boldsymbol{\lambda}'_2, \dots, \boldsymbol{\lambda}'_n)$; (7)
  if $(\|\boldsymbol{\lambda}' - \boldsymbol{\lambda}\| \ge \epsilon)$ **then** (8)
    $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}'$; (9)
  else (10)
    $\boldsymbol{\lambda}^* \leftarrow \boldsymbol{\lambda}'$; (11)
    return $\boldsymbol{\lambda}^*$; (12)
  end if (13)
forever. (14)

6 Numerical Examples and Data

In this section, we demonstrate numerical examples and data.

Let us consider $n = 10$ UEs and $m = 7$ MECs with the following parameters: $\hat{\lambda}_i = 1.0 + 0.05(i-1)$ tasks/second, $\lambda_i = 1.0 + 0.1(i-1)$ tasks/second, $\overline{\hat{r}_i} = 0.5 + 0.05(i-1)$ BI, $\overline{\hat{r}_i^2} = 1.6\overline{\hat{r}_i}^2$ BI^2, $\overline{r_i} = 1.5 + 0.05(i-1)$ BI, $\overline{r^2} = 1.3\overline{r_i}^2$ BI^2, $\overline{d} = 1.0 + 0.1(i-1)$ MB, $\overline{d^2} = 1.5\overline{d}^2$ MB^2, $s_i = 1.5 + 0.1(i-1)$ GHz, $\tilde{s}_j = 3.2 + 0.1j$ GHz, $c_{i,j} = (10.0 + (i-1)) + 0.5j$ MB/second, for all $1 \le i \le n$ and $1 \le j \le m$.

In Table 2, we show the results of Nash equilibrium, i.e., $\boldsymbol{\lambda}^*_i = (\lambda^*_{i,0}, \lambda^*_{i,1}, \dots, \lambda^*_{i,m})$, for all $1 \le i \le n$. We also show the arrival rate $\tilde{\lambda}_j$ to MEC$_j$, and the utilization $\tilde{\rho}_j$ of the server in MEC$_j$, for all $1 \le j \le m$, and the utilization ρ_i of the server in UE$_i$, and the average response time T_i of all offloadable and non-offloadable tasks generated on UE$_i$, for all $1 \le i \le n$. We set $\epsilon = 10^{-11}$, which requires $K = 250$ rounds of repetition. The main observations of our numerical data are as follows.

- The n mobile user equipments, i.e., UE$_1$, UE$_2$, ..., UE$_n$, have increased server utilizations and average response times, due to their increased service demands (i.e., increased arrival rates and execution requirements).
- The m mobile edge clouds, i.e., MEC$_1$, MEC$_2$, ..., MEC$_m$, receive increased amount of workload and have increased server utilizations, due to their increased execution and communication speeds.

In Table 3, using UE$_5$ as an example, we show the speed of convergence of the Nash equilibrium. It is shown that after 45 rounds, $\boldsymbol{\lambda}_5$ is already very close to $\boldsymbol{\lambda}^*_5$, with the first three digits after the decimal point confirmed for $\lambda_{5,j}$, where $0 \le j \le m$.

TABLE 4
Numerical Data for the Number of Rounds

Accuracy Requirement ϵ	Number of Rounds K
10^{-1}	3
10^{-2}	13
10^{-3}	37
10^{-4}	61
10^{-5}	85
10^{-6}	109
10^{-7}	134
10^{-8}	158
10^{-9}	183
10^{-10}	208

As mentioned earlier, the number of rounds in Algorithm 3 mainly depends on the value of ϵ. In Table 4, we show the number of rounds K for the accuracy requirement $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-10}$. It seems that K increases roughly linearly with $\log(1/\epsilon)$.

7 Concluding Remarks

The paper has adopted a game theoretic approach to computation offloading strategy optimization for non-cooperative mobile users competing for resources from multiple heterogeneous mobile edge clouds. Queueing models are established for multiple mobile users and multiple heterogeneous mobile edge computing servers, so that the strategies as well as the payoff functions of all mobile users can be analytically available. We have proved the existence of the Nash equilibrium, and developed efficient algorithms to find the best action of each mobile user and the Nash equilibrium.

Acknowledgments

The author appreciates the four anonymous reviewers for their comments and suggestions to improve the quality of the manuscript.

References

[1] [Online]. Available: https://en.wikipedia.org/wiki/Convex_function, Accessed on Jul. 5, 2018.

[2] [Online]. Available: https://en.wikipedia.org/wiki/Mobile_edge_computing, Accessed on Jul. 5, 2018.

[3] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. 10th IEEE Int. Conf. Intell. Syst. Control*, Jan. 7–8, 2016, pp. 1–8.

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

[5] H. Cao and J. Cai, "Distributed multi-user 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.

[6] 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, 2016.

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

[8] 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.

[9] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing – A key technology towards 5G," ETSI White Paper No. 11, European Telecommunications Standards Institute, Sep. 2015.

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

[11] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for Cloudlet-based mobile cloud computing," *18th ACM Int. Conf. Model. Anal. Simul. Wireless Mobile Syst.*, Nov. 2–6, 2015, pp. 271–278.

[12] P. Mach and Z. Becvar, "Mobile edge computing: A Survey on architecture and computation offloading," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, Jul.-Sep. 2017.

[13] 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.*, Dec. 4–8, 2016, pp. 1–6.

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

[15] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," arXiv:1705.00704v1, May 1, 2017. [Online]. Available: https://arxiv.org/abs/1705.00704

[16] 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.

[17] 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 a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor of Chinese National Recruitment Program of Global Experts (1000 Plan) at Hunan University, China. He was an Intellectual Ventures endowed visiting chair professor at the National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China, during 2011-2014. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things, and cyber-physical systems. He has published more than 590 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently serving or has served on the editorial boards of *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*. He is a fellow of the IEEE.