Stackelberg-Game-Based Computation Offloading Method in Cloud–Edge Computing Networks

Huan Zhou, *Member, IEEE*, Zhenning Wang, *Graduate Student Member, IEEE*, Nan Cheng, *Member, IEEE*, Deze Zeng, *Member, IEEE*, and Pingzhi Fan, *Fellow, IEEE*

***Abstract*—Offloading computation tasks through cloud–edge collaboration has been a promising way to improve the Quality of Service (QoS) of applications. Usually, cloud server (CS) and edge server (ES) are selfish and rational and, therefore, it is imperative to develop incentive mechanisms, which can encourage idle ESs or the CS to participate in the task offloading process. In this article, we propose a computation offloading method based on the game theory, which is suitable for cloud–edge computing networks. It is considered that the CS has a lot of computation tasks to conduct, and ESs usually have idle computational resources. The CS can offload computation tasks to ESs with idle computational resources to reduce its own cost and pressure, and ESs can profit by selling their computational resources. The interaction between the CS and ESs is modeled as a Stackelberg game, and the proposed game is analyzed by using the backward induction method. It is proved that the game can achieve a unique Nash equilibrium. Then, a gradient-based iterative search algorithm (GISA) is proposed to obtain the optimal solution in order to maximize the utility of the CS and ESs. Finally, numerical simulation results show that our proposed method greatly outperforms other benchmark schemes under different scenarios, and can encourage ESs to trade their computational resources with the CS effectively.**

***Index Terms*—Cloud–edge, computation offloading, edge computing, game theory, Nash equilibrium.**

I. Introduction

As AN important paradigm of the fifth-generation network (5G), the Internet of Things (IoT) technology enables all kinds of ubiquitous objects to interact and cooperate with each other to achieve common goals [1]–[3]. It is predicted that in the near future, the number of IoT devices in the world will exceed 80 billion. At the same time, with the progress of technology and the popularity of smart mobile devices, the IoT technology has opened up many attractive types of applications with computing-intensive features, such as intelligent transportation, healthcare, online interactive games, and augmented/virtual reality (AR/VR) [4]–[7]. However, current devices have been already limited by computational resources and energy consumption, which may become an unavoidable bottleneck in supporting these computation-intensive applications in the future [8]–[10].

Cloud computing has become a promising example of increasing the computing power of devices, where resource-rich cloud data centers are used to run applications, thereby overcoming the challenges of constrained computing resources and limited storage capacity [11]. However, for latency-sensitive IoT services, cloud computing may not be feasible because of the additional transmission costs and delays associated with long transmission distances in cloud-based processing. Recently, edge computing has emerged as an important component to cope with the computation-intensive tasks in the 5G architecture, which extends cloud computing services from the centralized cloud to the edges of the network [12]. For example, the roadside unit on the road can equip with edge servers (ESs) to relieve the computing and storage pressure of vehicles or the cloud server (CS). To improve the Quality of Service (QoS) of applications with considerably reduced latency and system cost, cloud and edge can cooperate with each other to process computation tasks. Cloud–edge collaboration has become a research hot spot in recent years. CS's resources are abundant but the cost is higher, and ESs' resources are limited but the cost is relatively low. These two kinds of servers can cooperate with each other to give better play to their advantages [13].

Some studies have investigated the typical cloud–edge collaboration scenario, in which the mobile users offload the computation tasks to the edge and request the assistance of the CS when necessary [14]. Most of them aim to improve the QoS of the system, i.e., reduce the latency of tasks, system cost, or energy consumption, etc. [15]–[18]. However, the task offloading process inevitably consumes a lot of computation and communication resources. From the economical perspective, given that the ES and the CS are commonly rational and selfish, they will be reluctant to participate in the task offloading process without receiving any reward. Therefore, it is imperative to develop incentive mechanisms, which can encourage idle ESs or the CS to participate in the task offloading process.

Manuscript received 2 April 2021; revised 5 July 2021 and 29 December 2021; accepted 7 February 2022. Date of publication 23 February 2022; date of current version 24 August 2022. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62172255, Grant 61872221, Grant 62172375, and Grant 61731017; in part by the Open Research Projects of Zhejiang Lab under Grant 2021KE0AB02; and in part by the 111 Project under Grant 111-2-14. *(Corresponding authors: Huan Zhou; Deze Zeng.)*

Huan Zhou and Zhenning Wang are with the College of Computer and Information Technology, China Three Gorges University, Yichang 443002, China (e-mail: zhouhuan117@gmail.com; wzn0115@163.com).

Nan Cheng is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China (e-mail: dr.nan.cheng@ieee.org).

Deze Zeng is with the Hubei Key Laboratory of Intelligent Geo-Information Processing, School of Computer Science, China University of Geosciences, Wuhan 430074, China (e-mail: deze@cug.edu.cn).

Pingzhi Fan is with the Information Coding & Transmission Key Laboratory of Sichuan Province, CSNMT International Cooperation Research Centre (MoST), Southwest Jiaotong University, Chengdu 610031, China (e-mail: pzfan@swjtu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2022.3153089

Some researchers have used economic theories to design incentive mechanisms, such as the auction theory, contract theory, game theory, and so forth [19]–[22]. The game theory can be used to analyze the interactions between independent and selfish players to maximize the rewards for all players involved in the game, especially between the CS and ES [23], [24]. Particularly, the Stackelberg game can be used to study the conflict and cooperation among intelligent rational decision-makers (players), with participants divided into leaders and followers [25]–[28]. However, to our knowledge, existing studies about the design of the Stackelberg game in the cloud–edge computing networks rarely consider the impact of task execution time constraint, and the scenario that the CS has a lot of computation tasks to conduct.

In this article, different from the typical cloud–edge collaboration scenario, we consider the scenario that the CS has a lot of computation tasks to conduct, and ESs have idle computational resources. To reduce its cost and pressure, the CS can offload computation tasks to ESs with idle computational resources. The ESs can sell their computational resources to make a profit. In particular, the Stackelberg game is used to model the interaction between the CS and ESs, in which the CS is the leader, and ESs are followers. Both the CS and ESs want to maximize their utilities; thus, we design a gradient-based algorithm to obtain the optimal solution and prove the existence of a unique Nash equilibrium. In addition, the impact of task execution time and ES's satisfaction on the utility of the CS and ESs are also considered.

The key contributions are summarized as follows.

1) We model the interaction process between the CS and ESs as a Stackelberg game, where the CS is the leader and ESs are the followers. The goal is to maximize the utility of both ESs and the CS.
2) We use the backward induction method to analyze the proposed game, and prove that there is a unique Nash equilibrium between the CS and ESs. Then, we design a gradient-based iterative search algorithm (GISA) to obtain the optimal solution.
3) Numerical simulation results demonstrate the effectiveness of our proposed model and algorithm. It can be observed that our proposed model can well encourage the collaboration between the CS and ESs to achieve a win-win situation.

The remainder of this article is organized as follows. After reviewing the related work in Section II, Section III introduces the system model related to this article, including the network architecture, delay model, and the utility function of the CS and ESs. Section IV introduces the optimization problem and analysis. Section V introduces our proposed algorithm to find the Nash equilibrium. Section VI introduces evaluation the performance of our proposed methods. Finally, Section VII concludes this article.

II. Related Work

To improve the QoS of applications with considerably reduced latency and system cost, cloud–edge collaboration has attracted extensive attention from many researchers in recent years. Mahn *et al.* [16] investigated a multilevel offloading scheme that combines a low-latency MEC system with a CS, and presented a distributed iterative algorithm to solve the energy minimization problem. Zhang *et al.* [17] proposed a cost-effective architecture for mobile cloud–edge computing networks and designed an efficient wholesale and buy-back scheme to maximize the total profit of mobile ESs. Wu *et al.* [18] developed a hybrid offloading model for MCC and MEC cooperation and proposed a deep-learning-based algorithm to optimize the system utility and bandwidth allocation of mobile users. Li *et al.* [29] proposed a two-timescale Lyapunov optimization algorithm to solve the task offloading and resource purchase problems in edge-cloud collaboration systems with the aim of minimizing the cost of task offloading. Wu *et al.* [30] proposed an edge-cloud collaborative multitask computing offload model, and used a nonlinear exponential inertia weighted particle swarm optimization algorithm to get the solution. Wang *et al.* [31] designed a three-tier MU-edge cloud network architecture, and proposed a Q-learning-based algorithm to reduce the total weighted cost of time and energy consumption of mobile users.

Some economic-theory-based incentive mechanisms have been developed to encourage all parties to participate in the offloading process. Zhou *et al.* [21] proposed a delay-constraint and reverse-auction-based incentive mechanism for data offloading through WiFi access points (APs), which aims to maximize the revenue of mobile network operators. Wang *et al.* [32] designed a profit-maximizing multiround auction mechanism for resource transactions between the edge cloud as the seller and the mobile device as the buyer in a competitive environment. Zeng *et al.* [33] used the contract theory framework to formulate the negotiation between task publishers and fog nodes into an optimization problem, with the goal of maximizing the utility of task publishers and fog nodes. Li and Cai [34] proposed an online incentive mechanism integrating task performer selection, resource allocation, and scheduling for collaborative task offloading in MEC networks. Zhan and Zhang [35] proposed an incentive mechanism based on deep reinforcement learning to achieve effective edge learning.

In addition, the game theory is a powerful tool for analyzing the interaction between multiple independent and selfish entities that need to work together to achieve their goals. Recently, some game-theory-based incentive mechanisms have been proposed to encourage the CS or ESs to participate in the offloading process. Zeng *et al.* [25] analyzed the interaction between the requesting vehicle and the vehicular ESs based on the Stackelberg game and found the best strategy for them. Li *et al.* [27] proposed a computation offloading mechanism based on a two-stage Stackelberg game to analyze the interaction between multiple edge clouds and multiple IIoT devices, and proposed two dynamic iterative algorithms to analyze the Stackelberg equilibrium. Zeng *et al.* [36] proposed a novel reputation-based incentive mechanism via using the Stackelberg game in software-defined vehicle edge computing, and analyzed the optimal strategy for both sides of the game by using the reverse induction method. Chang and Wei [37] proposed a uniform-price access control mechanism

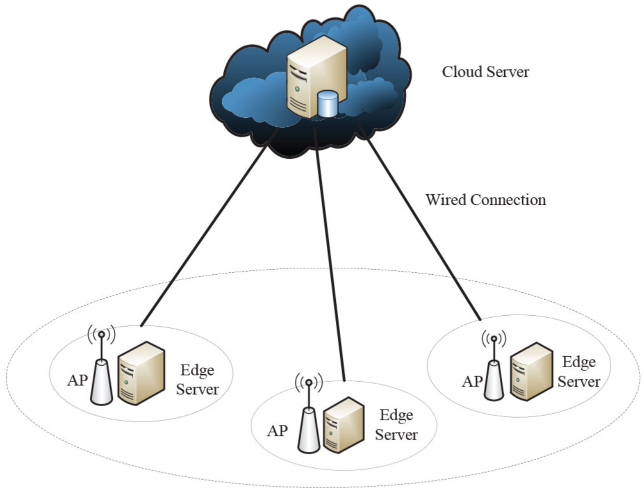


Fig. 1. System architecture of computation offloading.

for ESs, and used the Stackelberg game to reveal the relationship between the CS and ESs. Hong *et al.* [38] studied the multihop computing offloading problem of the IIoT-edge-cloud computing model, and used game theory to minimize the computing time and energy consumption of each task. Cao and Cai [39] formulated the multiuser computation offloading problem for Cloudlet-based mobile cloud computing as a noncooperative game, and then proposed a fully distributed computation offloading algorithm to maximize the number of beneficial cloudlet computing mobile devices. Zhang *et al.* [40] studied a UAV-assisted multiaccess edge computing system, and proposed a game-theory-based method to minimize the weighted cost of time delay and energy consumption.

Compared to the above existing studies, this article also uses the Stackelberg game to model the interaction process between the CS and ESs, aiming to maximize the utility of both the CS and ESs. However, different from typical cloud–edge collaboration scenario, this article considers the scenario that the CS has some computation tasks to offload to ESs, in which the CS is the leader, and ESs are followers. Furthermore, the impact of task execution time and ES's satisfaction on the utility of the CS and ESs are also considered.

III. System Model

In this article, we consider computation offloading in a cloud–edge scenario. As shown in Fig. 1, a CS and N ESs associated with wireless APs exist in the system, where the CS and ESs are connected through wired lines. The CS has some computation tasks to be processed, and it can rent idle ESs within its range to help it process the computation tasks. ESs can improve their utility by processing the offloaded computation tasks, and the CS can reduce its cost and computing pressure through computation offloading.

For ease of reference, we have listed the notations used in this article and provided corresponding explanations in Table I.

TABLE I
Notations and Explanation

Notation	Explanation
$\mathcal{N}$	The set of all ESs
B_i	The maximum cache size of ES i used to store the data of offloaded computation tasks
L_i	The amount of total computation corresponding to B_i
s_i	The amount of computation required by ES i to serve its own users
p_i	The unit reward that ES i serves its own users
c_i	The ES i's cost per unit of computation
h_i	The CPU processing frequency of ES i
H_i	The computation tasks on the CS
l_i	The amount of computation required to accomplish the offloaded task H_i
b_i	The size of the offloaded task H_i
$E[X]$	The empirical mean of Gamma distribution
μ_i	The transmission rate between the CS and ES i
t_i^{max}	The time constraint of the task H_i
t_i^{tran}	The transmission time of task H_i from the CS to ES i
t_i^{com}	The computing time of task H_i on ES i
t_i	The total execution time of task H_i in the computation offloading
W_i	The reward-penalty cost coefficient of ES i
φ, λ	The fixed cost factors of the CS's utility
d_i	The CS's unit payment to ES i
d_i^{max}	The maximum threshold of the CS's unit payment to ES i
d_i^{min}	The minimum threshold of the CS's unit payment to ES i

A. ES's Model

We consider that the CS intends to offload its computation tasks to N ESs. For each ES $i \in \{1, 2, \ldots, N\}$, the computation state consists of several parameters, namely, $\Phi_i = \{B_i, L_i, s_i, p_i, c_i, h_i\}$. Here, B_i is the maximum cache size of ES i used to store the data of offloaded computation tasks [MB]; L_i is the amount of total computation corresponding to B_i [Megacycles]; s_i is the amount of computation required by ES i to serve its own users; p_i is the unit reward that ES i serves its own users; c_i is ES i's cost per unit of computation; and h_i is the CPU processing frequency of ES i [Megacycles/s]. Note that p_i should be larger than c_i.

Without loss of generality, we use $H_i \triangleq \{l_i, b_i, t_i^{\max}\}$ to denote the computation task assigned by the CS to ES i, where l_i stands for the amount of computation required to accomplish the offloaded task H_i, b_i denotes the size of the offloaded task H_i, and $t_i^{\max}$ denotes the time constraint of the task H_i. The variables l_i and b_i can be approximated by a Gamma distribution [41]. Hence, based on the empirical mean $E[X]$ of the Gamma distribution, we can obtain $l_i = E[X]b_i$.

B. Time Constraint and Reward–Penalty Function Model

We know that the time constraint of the task H_i is $t_i^{\max}$. If the computation task can be completed within the time constraint, the CS will give corresponding rewards to ESs, and if it is not completed within the time constraint, the CS will charge ESs a fine, which will affect the utility of the CS and ESs.

Since the CS and ESs are connected by wired lines, the transmission rate can be set as a fixed value. Assuming that the transmission rate is μ_i, we can get the time to transmit the task H_i from the CS to ES i as

$$t_i^{\text{tran}} = \frac{b_i}{\mu_i}. \tag{1}$$

Then, the computing time on ES i is the computation amount of the offloaded task divided by the processing frequency of ES i, which is expressed as

$$t_i^{\text{com}} = \frac{l_i}{h_i}. \tag{2}$$

Similar to [42], we ignore the time to return the computation results to the CS, so the total execution time of the task H_i in the computation offloading is easily calculated as

$$t_i = t_i^{\text{com}} + t_i^{\text{tran}} = \frac{b_i}{\mu_i} + \frac{l_i}{h_i}. \tag{3}$$

We define that ES i's reward–penalty function is related to the difference between the task execution time and the time constraint of the task H_i, which is formulated as

$$U_i^R = \left(t_i - t_i^{\max}\right) W_i \tag{4}$$

where W_i is the reward–penalty cost coefficient of ES i. When $t_i > t_i^{\max}$, it is the penalty function and $U_i^R > 0$; when $t_i < t_i^{\max}$, it is the reward function and $U_i^R < 0$.

Then, the reward–penalty function of all ESs can be calculated as

$$U^R = \sum_{i \in N} U_i^R = \sum_{i \in N} \left(t_i - t_i^{\max}\right) W_i. \tag{5}$$

C. Utility Function of the CS

We use an exponential function $\varphi e^{\lambda f(x)}$ to express the relationship between the cost of the CS and the amount of offloaded computation, where φ and λ are the fixed cost factors in this article. Therefore, without considering computation offloading, the cost of processing the computation tasks on the CS is

$$C_0 = \varphi e^{\lambda \sum_{i \in N} l_i}. \tag{6}$$

In the case of computation offloading, the cost of the CS can be expressed as

$$C_1 = \sum_{i \in N} d_i l_i \tag{7}$$

where d_i denotes the CS's unit payment to ES i.

We define the utility of the CS as the cost reduction brought by computation offloading. The utility of the CS can be expressed as the cost without renting ESs' computational resources minus the cost after renting plus the rewards or penalties. Therefore, considering the reward–penalty function of all ESs, the utility function of the CS can be expressed as

$$\begin{aligned} U_C &= C_0 - C_1 + U^R \\ &= \varphi e^{\lambda \sum_{i \in N} l_i} - \sum_{i \in N} d_i l_i + \sum_{i \in N} U_i^R. \end{aligned} \tag{8}$$

D. Utility Function of ESs

For ES i, we assume that the computational resources prioritize for its own use, which is more in line with actual application scenarios. According to [28], we quantitatively simulate ESs' satisfaction by describing the difference between actual consumption and ideal demand. We define the satisfaction function of ESs with a logarithmic function, which is usually used in economics. The satisfaction function of ES i can be expressed as

$$\phi(L_i - l_i) = \ln(1 + L_i - l_i - s_i) \tag{9}$$

where $L_i - l_i$ indicates the remaining amount of computation for its own use after providing computation offloading services. When the remaining computation amount is lower than its own needs s_i, the satisfaction function $\phi(L_i - l_i) < 0$. If the actual remaining computation amount is larger than its own needs s_i, on the other hand, ES i is satisfied, thus $\phi(L_i - l_i) > 0$. When the difference is equal to s_i, $\phi(L_i - l_i) = 0$. Therefore, the income of ES i after selling computational resources can be described as follows:

$$y_i(l_i) = (p_i - c_i)(L_i - l_i + \phi(L_i - l_i)) + (d_i - c_i) l_i. \tag{10}$$

The utility function of ES i can be the profit after trading $y_i(l_i)$ minus the profit before trading $y_i(0)$, that is the income after providing computation offloading services minus the income without providing computation offloading services, where $y_i(0)$ is denoted as follows:

$$y_i(0) = (p_i - c_i)(L_i + \phi(L_i)). \tag{11}$$

Therefore, the utility function of ES i is the increased profits by assisting the CS to perform computation tasks. In addition, there is a reward–penalty function, so the utility of ES i is

$$\begin{aligned} U_i &= y_i(l_i) - y_i(0) - U_i^R \\ &= (p_i - c_i)(L_i - l_i + \ln(1 + L_i - l_i - s_i)) + (d_i - c_i) l_i \\ &\quad - (p_i - c_i)(L_i + \ln(1 + L_i - s_i)) - U_i^R \\ &= (p_i - c_i)\left(\ln \frac{1 + L_i - l_i - s_i}{1 + L_i - s_i} - l_i\right) + (d_i - c_i) l_i \\ &\quad - \left(t_i - t_i^{\max}\right) W_i. \end{aligned} \tag{12}$$

IV. Problem Formation and Analysis

In this section, we first describe the optimization problem and model it as a stackelberg game model. Then, we analyze the proposed problem and use the backward induction method to find the Nash equilibrium.

A. Problem Formulation

We consider the game between the CS and ESs as a single leader multifollower Stackelberg game model. The CS acting as the leader gives its payment strategy d_i, ES i acts as the follower to determine the size of computation task H_i which is received from the CS, and uses l_i to represent ES i's offloading strategy. Therefore, the payment profile and computation task offloading profile are denoted as $\mathbf{d} = (d_1, d_2, \ldots, d_N)$, and $\mathbf{l} = (l_1, l_2, \ldots, l_N)$, respectively. Our goal is to maximize the utility of ESs and the CS.

Formally, the Stackelberg game Γ can be constructed as

$$\Gamma = \{(CS, ESs), (d_i, l_i), (U_C, U_i)\}. \tag{13}$$

Player	CS (leader)	ES i (follower)
Auction	payment strategy d_i	offloading strategy l_i
Utility	U_C	U_i

Our optimization problem is defined as

$$\begin{aligned} \text{Problem 1 : } & \max \ U_C \\ & \text{s.t. } \ d_i^{\min} \le d_i \le d_i^{\max} \end{aligned} \tag{14}$$

where $d_i^{\min}$ and $d_i^{\max}$ are the minimum and maximum bid strategies, which will be introduced as follows:

$$\begin{aligned} \text{Problem 2 : } & \max \ U_i \\ & \text{s.t. } \ 0 \le l_i \le L_i \end{aligned} \tag{15}$$

where the amount of offloaded computation l_i that ES i receives cannot exceed its own capacity.

B. Problem Analysis

The backward induction method is utilized to analyze the proposed Stackelberg game. First, the optimal decision of each ES on the size of the offloaded computation task is analyzed. Then, based on the decisions of all ESs, the optimal strategy of the CS on the payment is investigated.

We first prove that there is a Nash equilibrium in the question raised.

Definition 1: There exists one and only one Nash equilibrium strategy among ESs, where $\mathbf{l}^* = (l_1^*, l_2^*, \ldots, l_N^*)$. At this time, there is a utility function $U_i(l_i^*, l_{-i}^*) \ge U_i(l_i, l_{-i}^*)$, where l_{-i}^* is the best strategies of other ESs.

Theorem 1: In the game, the CS acts as the leader, then, as a follower, ES i has the best strategy about its offloading computation size l_i^*, which is shown as

$$l_i^* = 1 + L_i - s_i + \frac{p_i - c_i}{p_i + A_i - d_i} \tag{16}$$

where

$$A_i = \left(\frac{1}{E(X)\mu_i} + \frac{1}{h_i}\right) W_i. \tag{17}$$

Proof: For ES i, its utility function is U_i, the first-order derivative of (12) is equal to

$$\frac{\partial U_i}{\partial l_i} = (p_i - c_i)\left(\frac{-1}{1 + L_i - l_i - s_i} - 1\right) + d_i - c_i - A_i. \tag{18}$$

Afterward, the second-order derivative of (12) is derived as

$$\frac{\partial^2 U_i}{\partial l_i^2} = \frac{-(p_i - c_i)}{(1 + L_i - l_i - s_i)^2}. \tag{19}$$

Because $(p_i - c_i) > 0$ and $(1 + L_i - l_i - s_i)^2 > 0$, the second-order derivative of the utility function is negative. We can observe that the utility function of ES i is a strictly concave function, which proves the existence of Nash equilibrium. Therefore, according to the expressions of (18) and (19), we can prove the uniqueness of strategy $\mathbf{l} = (l_1, l_2, \ldots, l_N)$. Letting $(\partial U_i / \partial l_i) = 0$, then we get the best response strategy through (18).

We regard ES i's payment as its payment threshold by setting $l_i = 0$ and $l_i = L_i$, denoted by

$$d_i^{\min} = p_i + A_i + \frac{p_i - c_i}{1 + L_i - s_i} \tag{20}$$

$$d_i^{\max} = p_i + A_i + \frac{p_i - c_i}{1 - s_i}. \tag{21}$$

The best strategy should be between $d_i^{\min}$ and $d_i^{\max}$, so ES i's optimal offloaded computation satisfies

$$l_i^* = \begin{cases} 0, & d_i \le d_i^{\min} \\ 1 + L_i - s_i + \frac{p_i - c_i}{p_i + A_i - d_i}, & d_i^{\min} < d_i < d_i^{\max} \\ L_i, & d_i^{\max} \le d_i. \end{cases} \tag{22}$$

■

Lemma 1: Each ES has a unique optimal strategy for a given strategy of the CS.

Proof: If the CS gives a large enough bid strategy (greater than the maximum threshold), ES i will sell all its computational resources; if the bid strategy is too small (less than the minimum threshold), ES i will not sell any computational resources.

When $d_i^{\min} < d_i < d_i^{\max}$, we have

$$\frac{\partial l_i^*}{\partial d_i} = \frac{p_i - c_i}{(p_i + A_i - d_i)^2} \tag{23}$$

$$\frac{\partial^2 l_i^*}{\partial d_i^2} = \frac{-2(p_i - c_i)}{(p_i + A_i - d_i)^3}. \tag{24}$$

The first-order derivative is larger than 0, which means that the higher the payment provided by the CS, the more computational resources the ES will devote to computation offloading.

Moreover, the second-order derivative constant is negative, the above two formulas imply that l_i^* is an increasing concave down function of d_i. Therefore, according to the given range of d_i, it is a strictly concave function within the range, so the derived strategy l_i^* is optimal and unique. ■

Definition 2: There exists a unique Stackelberg equilibrium among the CS and ESs if $U_C(d_i^*, l_i^*) > U_C(d_i, l_i^*)$.

Theorem 2: Considering all the best strategies of ESs, the CS has a unique best strategy.

Proof: The utility function of the CS, denoted by (8), can be detailed as

$$U_C = \varphi e^{\lambda \sum_{i \in N} l_i} - \sum_{i \in N} d_i l_i + \sum_{i \in N} \left(\frac{b_i}{\mu_i} + \frac{l_i}{h_i} - t_i^{\max}\right) W_i. \tag{25}$$

Substituting the obtained best strategy l_i^* into the utility function U_C, we can get $U_C(d_i, l_i^*)$. We use M to represent $\lambda \varphi e^{\lambda \sum_{i \in N} l_i^*}$, then the first-order partial derivative of $U_C(d_i, l_i^*)$ to d_i is

$$\begin{aligned} \frac{\partial U_C}{\partial d_i} &= M \frac{\partial l_i^*}{\partial d_i} - d_i \frac{\partial l_i^*}{\partial d_i} - G_i \frac{\partial l_i^*}{\partial d_i} \\ &= -l_i^* + (p_i - c_i) \frac{M - d_i - G_i}{(p_i + A_i - d_i)^2} \end{aligned} \tag{26}$$

where

$$G_i = \frac{(h_i + E[X]\mu_i)}{h_i \mu_i E[X]}. \tag{27}$$

We take the second-order derivative of $U_C(\mathbf{d}^*, \mathbf{l}^*)$ with respect to d_i in the form of the Hessian matrix of $U_C(\mathbf{d}^*, \mathbf{l}^*)$. Then, we can obtain its Hessian matrix H as

$$H = \begin{pmatrix} \frac{\partial^2 U_C}{\partial d_1 \partial d_1} & \frac{\partial^2 U_C}{\partial d_1 \partial d_2} & \cdots & \frac{\partial^2 U_C}{\partial d_1 \partial d_n} \\ \frac{\partial^2 U_C}{\partial d_2 \partial d_1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 U_C}{\partial d_n \partial d_1} & \cdots & \cdots & \frac{\partial^2 U_C}{\partial d_n \partial d_n} \end{pmatrix}$$

where

$$\frac{\partial^2 U_C}{\partial {d_i}^2} = \frac{-\lambda M(p_i - c_i)(p_i - c_i) + 2(p_i + A_i - d_i)(p_i - c_i)(M - d_i - G_i)}{(p_i + A_i - d_i)^2 (p_i + A_i - d_i)^2} \tag{28}$$

and

$$\frac{\partial^2 U_C}{\partial d_i \partial d_j} = \frac{-\lambda M(p_i - c_i)\big(p_j - c_j\big)}{(p_i + A_i - d_i)^2 \big(p_j + A_j - d_j\big)^2}. \tag{29}$$

Under this situation, we only need to prove that H is a negative-definite matrix. Moreover, according to (28) and (29), we can divide H into H_1 and H_2, where $H = H_1 + H_2$

$$H_1 = \begin{pmatrix} \frac{2(p_i+A_i-d_i)(p_i-c_i)(M-d_i-G_i)}{(p_i+A_i-d_i)^2(p_i+A_i-d_i)^2} & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & \frac{2(p_n+A_n-d_n)(p_n-c_n)(M-d_n-G_n)}{(p_n+A_n-d_n)^2(p_n+A_n-d_n)^2} \end{pmatrix}$$

$$H_2 = \begin{pmatrix} \frac{-\lambda M(p_1-c_1)(p_1-c_1)}{(p_1+A_1-d_1)^2(p_1+A_1-d_1)^2} & & \frac{-\lambda M(p_1-c_1)(p_i-c_i)}{(p_1+A_1-d_1)^2(p_i+A_i-d_i)^2} \\ \frac{-\lambda M(p_2-c_2)(p_1-c_1)}{(p_2+A_2-d_2)^2(p_1+A_1-d_1)^2} & & \vdots \\ \vdots & \ddots & \vdots \\ \frac{-\lambda M(p_i-c_i)(p_1-c_1)}{(p_i+A_i-d_i)^2(p_1+A_1-d_1)^2} & \cdots & \frac{-\lambda M(p_i-c_i)(p_i-c_i)}{(p_i+A_i-d_i)^2(p_i+A_i-d_i)^2} \end{pmatrix}.$$

We first need to prove that H_1 is a negative definite matrix. According to (20) and (21), we know $p_i + A_i - d_i < 0$. Therefore, we only need to prove $M - d_i - G_i = \lambda\varphi e^{\lambda \sum_{i\in N} l_i^*} - d_i - G_i \geq 0$, then we can prove H_1 is a negative definite matrix. According to (26), letting $(\partial U_C/\partial d_i) = 0$, we can obtain

$$\lambda\varphi e^{\lambda \sum_{i\in N} l_i^*} - d_i^* - G_i = \frac{l_i^*\big(p_i + A_i - d_i^*\big)^2}{p_i - c_i}. \tag{30}$$

It is obvious that $[(l_i^*(p_i + A_i - d_i^*)^2)/(p_i - c_i)] > 0$; thus, we can prove that $\lambda\varphi e^{\lambda \sum_{i\in N} l_i^*} - d_i^* - G_i \geq 0$. For H_2, according to the convex optimization theorem, it is a real symmetric matrix, so it is a negative-definite matrix. Therefore, we prove that H is a strict negative-definite matrix, and U_C is strictly concave. Then, we can obtain that (14) is a convex optimization problem. Meanwhile, the optimal strategy $\mathbf{d}^*$ is unique.

In summary, Theorem 2 is proved. ■

Algorithm 1 GISA

Input: Φ_i=$\{B_i, L_i, s_i, p_i, c_i, h_i\}$, and φ, λ;
Output: $\mathbf{l}^*$, $\mathbf{d}^*$, U_i^*, U_C^*;
1: **Initialization:** $k = 0$, precision threshold ω, step size θ;
2: Set up $\mathbf{d}(k) = (d_1(k), d_2(k), \ldots, d_n(k))$, where $d_i^{min} < d_i(k) < d_i^{max}$;
3: ES i decides its computation offloading strategy $l_i(k)$ according to Eq. (22);
4: CS uses the gradient ascending search algorithm to find the best payment set, where $\mathbf{d}(k+1) = \mathbf{d}(k) + \theta\nabla U(\mathbf{d}(k), l^*(\mathbf{d}(k)))$;
5: The giving price $\mathbf{d}(k+1)$ will be sent to all ESs;
6: **while** $\frac{\|l(k+1)-l(k)\|_1}{\|l(k)\|_1} \geq \omega$ **do**
7: Repeat from step (2) to step (5).
8: $k \leftarrow k+1$;
9: **end while**
10: Optimal d^* is obtained;
11: Calculating l^* according to Eq. (22);
12: **return** U_i^*, U_C^*

V. Gradient-Based Iterative Search Algorithm

In this section, for the optimal strategies $\mathbf{l}^* = (l_1^*, l_2^*, \ldots, l_N^*)$ and $\mathbf{d}^* = (d_1^*, d_2^*, \cdots, d_N^*)$, we propose GISA to find the unique solution that can achieve the Nash equilibrium and Stackelberg equilibrium.

As shown in Algorithm 1, at first, we set $k = 0$, the CS sets the initial payments $\mathbf{d}$ between $d_i^{\min}$ and $d_i^{\max}$, which can be calculated by (20) and (21), then sends $\mathbf{d}$ to all the ESs. Next, each ES calculates its initial computation offloading strategy l_i based on (22). Having received l_i, the CS computes its new payment according to $\mathbf{d}(k+1) = \mathbf{d}(k) + \theta\nabla U(\mathbf{d}(k), l^*(\mathbf{d}(k)))$, where $\nabla U(\mathbf{d}(k), l^*(\mathbf{d}(k)))$ is the gradient with $[(\partial U_C(k))/(\partial d_i(k))]$ based on (22) and (26), and θ is the step size. Then, we set $k+1 = k$, the CS and ESs will repeat the above process to continue searching their better strategies. The algorithm stops repetition and the iteration ends until $[(\|l(k+1) - l(k)\|_1)/(\|l(k)\|_1)] \leq \omega$, where ω is the precision threshold and $\|x\|_1$ represents x's first-order paradigm. At this moment, we can find the approximate optimal solution to the CS and ESs' strategies, which are expressed as $\mathbf{d}^*$ and $\mathbf{l}^*$. Accordingly, we can get the best utility of the CS and ESs.

Remark 1: The proposed GISA is computationally efficient. In our proposed GISA, the computational complexity is related to the number of iterations. Obviously, only one layer of the while-loop is included, and the while-loop executes $k+1$ times. In each loop, the algorithm is repeated four steps. Then, the computational complexity of the proposed GISA is $O(4k)$. Therefore, the proposed algorithm has low computational complexity.

VI. Numerical Results

In this section, we conduct simulations to prove the effectiveness of our proposed algorithm. Specifically, we compare the proposed method with other benchmark methods. We also evaluate the influence of some parameters on the utility of

ESs and the CS, and verify the convergence of our proposed algorithm.

A. Parameters Settings

We initially set the number of ESs as 5, and the cache size used to store the data of offloaded computation tasks in the range of [100, 200] (in MB) [2]. The total computation amounts of each ES is set as 100 (in megacycles). The amount of computation required by each ES to serve its own users is in the range of [0, 50] (in megacycles), the reward of ESs is in the range of [0.3, 1.2], and the unit cost of computation of ESs is over [0.1, 0.9] according to [21]. The reward–penalty unit price is in the range of [0.01, 0.05]. According to [41], we set Gamma distribution parameters $\alpha = 0.5$, $\beta = 1.6$, and the initial coefficients $\varphi = 0.05$ and $\lambda = 30$.

For performance comparison, we introduce the following three benchmark schemes.

1) *Random:* Both the payment of the CS and ESs' offloading strategies are generated randomly within a given range.
2) *Game-Theory-Based Offloading Strategy (GTOS):* Similar to [43], only the offloading strategy game among ESs is considered, which is regarded as a noncooperative game among ESs. Under this situation, each ES determines its best offloading strategy.
3) *Game-Theory-Based Payment Strategy (GTPS):* In contrast to GTOS, GTPS only considers the CS's payment strategy game, in which the CS's payment strategy is modeled as a game, and the CS determines the optimal payment for each ES.

B. Convergence Performance

In this part, we analyze the convergence of our proposed algorithm.

To represent the iterative process more clearly, we set $p_i = 1.2$ and $c_i = 0.4$. Fig. 2 shows the iterative process of ESs' and the CS's utility. It can be observed that the utility of ESs and the CS grows gradually and eventually converges and stabilizes. Fig. 2(a) shows that the utility of the CS is relatively small at the beginning and increases rapidly with the increase of the iterations, and starts to stabilize after about 80 iterations and approach the optimal solution. Fig. 2(b) shows that the total utility of ESs starts from a relatively higher point, which is related to the initial value we set. It can be observed that the total utility of ESs does not reach a steady state quickly compared to that of the CS. This is because the relationship among ESs is noncooperative, ESs compete with each other to make their utility as large as possible. In the end, both ESs and the CS achieve a relatively optimal solution.

C. Impact of Parameters

In this part, we evaluate the impact of different parameters on the CS's utility and ESs' utility.

Fig. 3 shows the impact of unit cost c_i and unit reward p_i on the CS's utility. As illustrated in Fig. 3, the utility of the CS increases with the increase of c_i. This is because as the ES's cost per unit c_i increases, and its reward per unit p_i stays the same, it will gain less benefits from serving its own users, so the ES will be more willing to help the CS offload computation tasks. The more computation tasks the CS offload, the larger its utility. Furthermore, the utility of the CS decreases with the increase of p_i. This is because as the ES's unit reward p_i increases, and the unit cost c_i remains the same, the ES can gain more benefits from serving its own users, so the ES will be more willing to serve its own users, which leads to the decrease of the utility of the CS.

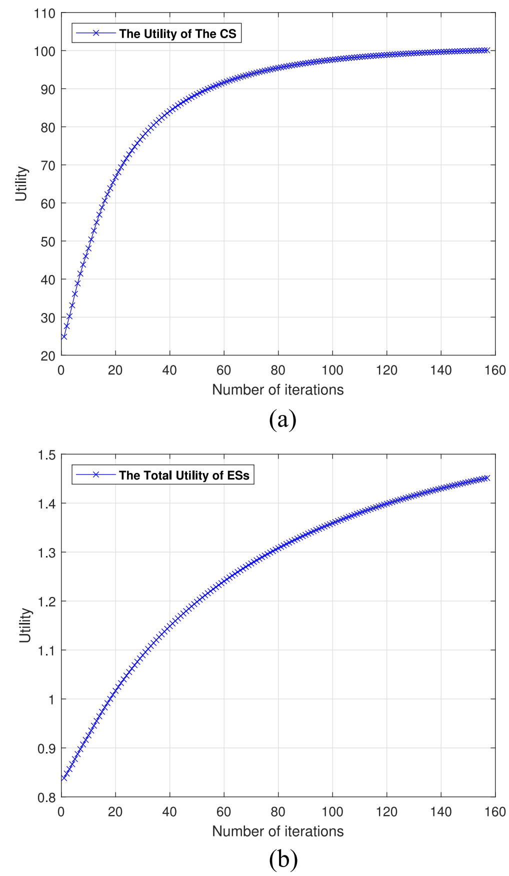


Fig. 2. Iterative process of all ESs' and the CS's utility. (a) Utility of the CS. (b) Total utility of ESs.

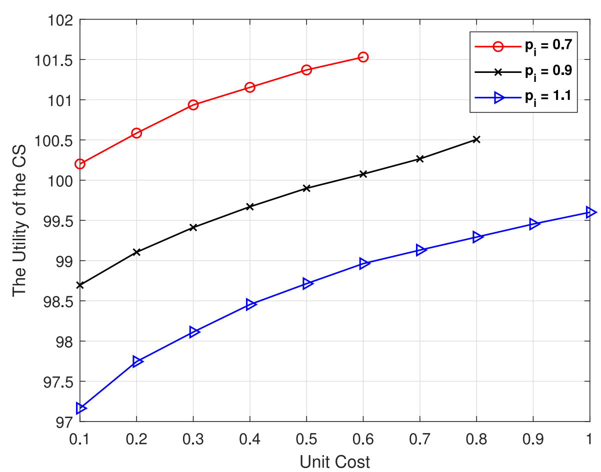


Fig. 3. Utility of the CS versus unit cost c_i.

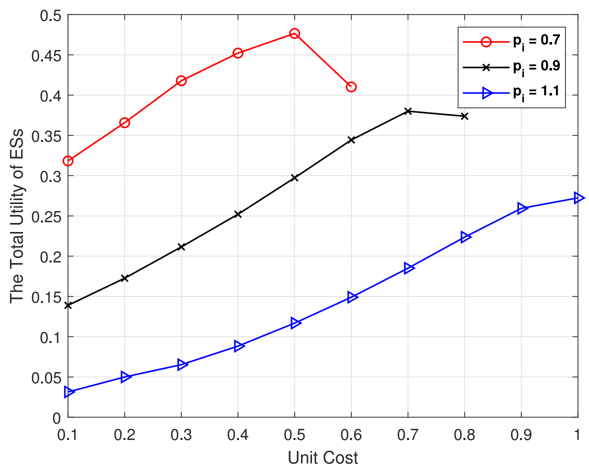


Fig. 4. Total utility of ESs versus unit cost c_i.

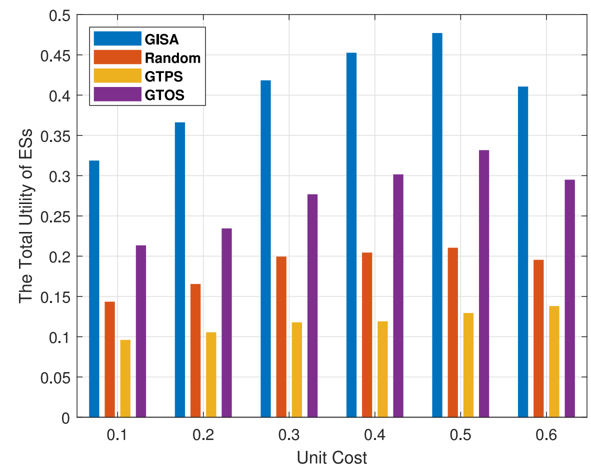


Fig. 5. Performance comparison in terms of the total utility of ESs with different unit cost c_i.

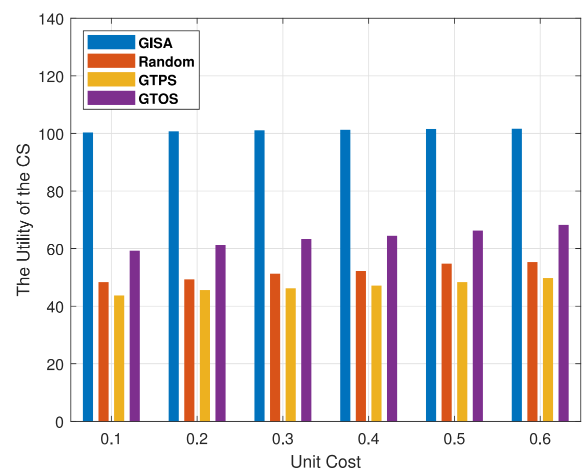


Fig. 6. Performance comparison in terms of the utility of the CS with different unit cost c_i.

Fig. 4 shows the impact of unit cost c_i and unit reward p_i on the total utility of ESs. It can be found that although the unit cost increases, the total utility of ESs increases. This is because as the ES's unit cost c_i increases, its revenue from its own users decreases, then it will be more willing to assist the CS in offloading more computation tasks so as to increase its utility. Of course, when c_i is too high and especially close to p_i, the total utility of ESs does not continue to increase but decrease. This is because ESs cannot gain benefits from the offloading process if the unit cost c_i is too high. Similar to the CS's utility, the total utility of ESs decreases with the increases of unit reward p_i. This is because as p_i increases, the ES will be more willing to serve its own users, which leads to the decrease of the total utility of ESs.

According to Figs. 3 and 4, we observe that the utility of both CS and ESs increases and all participants will benefit from the transaction process. If the CS offloads more computation tasks to the ESs, the utility of the CS and ESs will increase. In other words, our proposed model can encourage the collaboration between the CS and ESs to achieve a win-win situation, which demonstrates the effectiveness of our proposed model.

D. Performance Comparison

In this part, we compare the proposed GISA with three other benchmark schemes under different scenarios.

First of all, we compare the utility of the CS and ESs under different unit cost c_i, and we set the unit reward $p_i = 0.7$. Figs. 5 and 6 show the performance comparison of GISA and other benchmark schemes under different unit cost c_i, respectively. Fig. 5 compares the total utility of ESs, and Fig. 6 compares the utility of the CS. It can be observed that our proposed GISA has the highest utility under different unit cost. In addition, the other three schemes show the same trend as GISA with the increase of c_i. Specifically, we can observe that GTPS has the lowest utility. This is because if the ES's offloading strategy is given, the CS does not have to consider the effect of the ES when playing the payment game. Under this situation, the CS cannot offload more computation tasks, so under GTPS, both the CS and ESs have lower utility. Furthermore, the utility of the CS and ESs in GTOS is relatively high, and close to our proposed GISA. This is because the ES's offloading strategy is more influential than the CS's payment strategy in the game.

Fig. 7 shows the comparison of the average utility of ESs under different number of ESs, and Fig. 8 shows the comparison of the utility of the CS under different number of ESs. As the number of ESs increases, we can observe that our proposed GISA has the maximum utility for both the CS and ESs, which proves the superiority of our proposed scheme. Similarly, in the other three benchmark schemes, GTOS has a higher utility and GTPS has a lower utility. Moreover, as the number of ESs increases, the average utility of the ESs increases very slowly. This is because the ESs are noncooperative games that compete with each other. The utility of the CS increases rapidly with the growth of the number of ESs. This is because as the number of ESs participating in the task offloading process increases, the CS can offload more computation tasks to increase its utility.

To summarize, it can be found that the proposed GISA can not only improve the utility of the CS and ESs, but

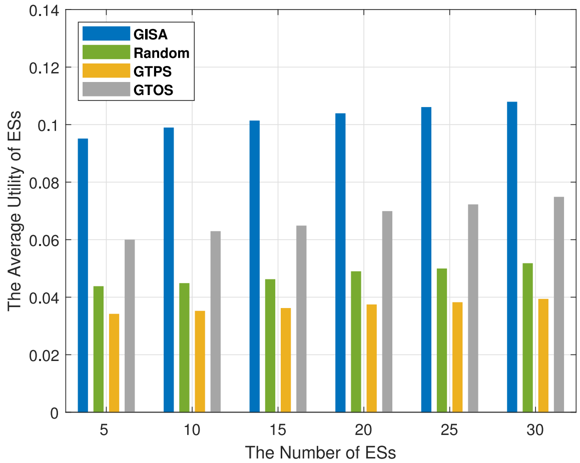


Fig. 7. Performance comparison in terms of the average utility of ESs with different number of ESs.

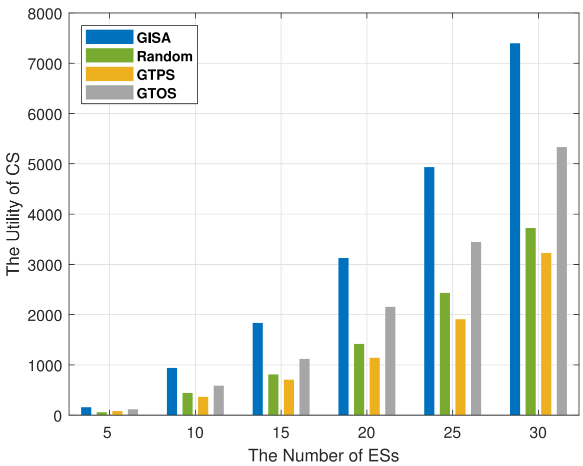


Fig. 8. Performance comparison in terms of the utility of the CS with different number of ESs.

also achieve desirable equilibrium. Therefore, it is demonstrated that the proposed GISA is effective under different scenarios.

VII. Conclusion

In this article, we have investigated the scenario that the CS has a lot of computation tasks to implement, and ESs have idle computational resources to sell. We modeled the interaction between the CS and ESs as a Stackelberg game, and analyzed the proposed game by using the backward induction method. We proved that the proposed game can achieve a unique Nash equilibrium, and then proposed a GISA to obtain the optimal solution. Numerical simulation results illustrated that our proposed GISA can effectively encourage the collaboration between the CS and ESs, and greatly outperforms other benchmark schemes under different scenarios.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[2] Y. Hui, Z. Su, T. H. Luan, C. Li, G. Mao, and W. Wu, "A game theoretic scheme for collaborative vehicular task offloading in 5G HetNets," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16044–16056, Dec. 2020.

[3] Z. Ning *et al.*, "Intelligent edge computing in Internet of Vehicles: A joint computation offloading and caching solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2212–2225, Apr. 2021.

[4] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven deep reinforcement learning for content caching and D2D offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445–2460, Aug. 2021.

[5] H. Zhou, V. C. M. Leung, C. Zhu, S. Xu, and J. Fan, "Predicting temporal social contact patterns for data forwarding in opportunistic mobile networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10372–10383, Nov. 2017.

[6] N. Cheng *et al.*, "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 26–32, Aug. 2018.

[7] Z. Ma, N. Nuermaimaiti, H. Zhang, H. Zhou, and A. Nallanathan, "Deployment model and performance analysis of clustered D2D caching networks under cluster-centric caching strategy," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4933–4945, Aug. 2020.

[8] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4854–4866, Jun. 2019.

[9] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[10] C. Zhou *et al.*, "Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 911–925, Feb. 2021.

[11] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3113–3126, Apr. 2019.

[12] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Netw.*, vol. 32, no. 5, pp. 112–117, Sep./Oct. 2018.

[13] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Jun. 2020.

[14] K. Jiang, C. Sun, H. Zhou, X. Li, M. Dong, and V. C. M. Leung, "Intelligence-empowered mobile edge computing: Framework, issues, implementation, and outlook," *IEEE Netw.*, vol. 35, no. 5, pp. 74–82, Sep./Oct. 2021.

[15] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[16] T. Mahn, H. Al-Shatri, and A. Klein, "Distributed algorithm for energy efficient joint cloud and edge computing with splittable tasks," in *Proc. IEEE WCNC*, Marrakesh, Morocco, 2019, pp. 1–7.

[17] Y. Zhang, X. Lan, Y. Li, L. Cai, and J. Pan, "Efficient computation resource management in mobile edge-cloud computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3455–3466, Apr. 2019.

[18] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020.

[19] H. Zhou, T. Wu, X. Chen, S. He, and J. Wu, "RAIM: A reverse auction-based incentive mechanism for mobile data offloading through opportunistic mobile networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Nov. 15, 2021, doi: 10.1109/TNSE.2021.3126367.

[20] J. Moura and D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 260–288, 1st Quart., 2019.

[21] H. Zhou, X. Chen, S. He, J. Chen, and J. Wu, "DRAIM: A novel delay-constraint and reverse auction-based incentive mechanism for WiFi offloading," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 711–722, Apr. 2020.

[22] Z. Ning *et al.*, "Mobile edge computing enabled 5G health monitoring for Internet of Medical Things: A decentralized game theoretic approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 463–478, Feb. 2021.

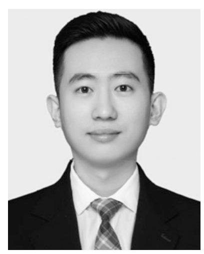
[23] T. Mahn, M. Wirth, and A. Klein, "Game theoretic algorithm for energy efficient mobile edge computing with multiple access points," in *Proc. IEEE MobileCloud*, Oxford, U.K., 2020, pp. 31–38.

[24] J. Zhang, X. Huang, and R. Yu, "Optimal task assignment with delay constraint for parked vehicle assisted edge computing: A Stackelberg game approach," *IEEE Commun. Lett.*, vol. 24, no. 3, pp. 598–602, Mar. 2020.
[25] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3247–3257, Jun. 2021.
[26] R. Wang, F. Zeng, L. Yao, and J. Wu, "Game-theoretic algorithm designs and analysis for interactions among contributors in mobile crowdsourcing with word of mouth," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8271–8286, Sep. 2020.
[27] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5444–5455, Aug. 2020.
[28] Y. Jie, C. Guo, K.-K. R. Choo, C. Z. Liu, and M. Li, "Game-theoretic resource allocation for fog-based industrial Internet of Things environment," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3041–3052, Apr. 2020.
[29] R. Li, Z. Zhou, X. Chen, and Q. Ling, "Resource price-aware offloading for edge-cloud collaboration: A two-timescale online control approach," *IEEE Trans. Cloud Comput.*, early access, Aug. 27, 2019, doi: 10.1109/TCC.2019.2937928.
[30] J. Wu, Z. Cao, Y. Zhang, and X. Zhang, "Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in MEC," in *Proc. IEEE ICPADS*, Tianjin, China, 2019, pp. 959–962.
[31] M. Wang, S. Shi, S. Gu, X. Gu, and X. Qin, "Q-learning based computation offloading for multi-UAV-enabled cloud-edge computing networks," *IET Commun.*, vol. 14, no. 15, pp. 2481–2490, 2020.
[32] Q. Wang, S. Guo, Y. Wang, and Y. Yang, "Incentive mechanism for edge cloud profit maximization in mobile edge computing," in *Proc. IEEE ICC*, Shanghai, China, 2019, pp. 1–6.
[33] M. Zeng, Y. Li, K. Zhang, M. Waqas, and D. Jin, "Incentive mechanism design for computation offloading in heterogeneous fog computing: A contract-based approach," in *Proc. IEEE ICC*, Kansas City, MO, USA, 2018, pp. 1–6.
[34] G. Li and J. Cai, "An online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 624–636, Jan. 2020.
[35] Y. Zhan and J. Zhang, "An incentive mechanism design for efficient edge learning by deep reinforcement learning approach," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, 2020, pp. 2489–2498.
[36] F. Zeng, Y. Chen, L. Yao, and J. Wu, "A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing," *Peer-to-Peer Netw. Appl.*, vol. 14, pp. 467–481, Mar. 2021.
[37] Z.-L. Chang and H.-Y. Wei, "Flat-rate pricing for green edge computing with latency guarantee: A Stackelberg game approach," in *Proc. IEEE GLOBECOM*, Waikoloa, HI, USA, 2019, pp. 1–6.
[38] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2759–2774, Dec. 2019.
[39] 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.
[40] K. Zhang, X. Gui, D. Ren, and D. Li, "Energy–latency tradeoff for computation offloading in UAV-assisted multiaccess edge computing system," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6709–6719, Apr. 2021.
[41] S. Jošilo and G. Dán, "Joint management of wireless and computing resources for computation offloading in mobile edge clouds," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1507–1520, Oct.–Dec. 2021.
[42] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, Jan. 2022.
[43] Y. Wang *et al.*, "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.

Huan Zhou (Member, IEEE) received the Ph.D. degree from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2014.

He was a Visiting Scholar with Temple University, Philadelphia, PA, USA, from November 2012 to May 2013, and a CSC Supported Postdoctoral Fellow with the University of British Columbia, Vancouver, BC, Canada, from November 2016 to November 2017. He is currently a Full Professor with the College of Computer and Information Technology, China Three Gorges University, Yichang, China. He has published more than 50 research papers in some international journals and conferences, including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. His research interests include opportunistic mobile networks, VANETs, mobile data offloading, and mobile-edge computing.

Prof. Zhou received the Best Paper Award of I-SPAN 2014 and I-SPAN 2018. He was a Lead Guest Editor of *Pervasive and Mobile Computing*, the TPC Chair of EAI BDTA 2020, the Local Arrangement Chair of I-SPAN 2018, the Special Session Chair of the 3rd International Conference on Internet of Vehicles (IOV 2016), and the TPC Member of IEEE Globecom, ICC, and ICCCN. He is currently serving as an Associate Editor for IEEE ACCESS and *EURASIP Journal on Wireless Communications and Networking*.



Zhenning Wang (Graduate Student Member, IEEE) received the B.S. degree from Qilu University of Technology, Jinan, China, in 2019. He is currently pursuing the Bachelor of Engineering degree with the College of Computer and Information Technology, China Three Gorges University, Yichang, China.

His main research interests are edge computing, computation offloading, and game theory.

Nan Cheng (Member, IEEE) received the B.E. and M.S. degrees from the Department of Electronics and Information Engineering, Tongji University, Shanghai, China, in 2009 and 2012, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2016.

He worked as a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, from 2017 to 2019. He is currently a Professor with the State Key Laboratory of Integrated Service Networks and the School of Telecommunication Engineering, Xidian University, Xi'an, Shaanxi, China. His current research focuses on B5G/6G, space–air–ground-integrated network, big data in vehicular networks, and self-driving system. His research interests also include performance analysis, MAC, opportunistic communication, and application of AI for vehicular networks.

Deze Zeng (Member, IEEE) received the B.S. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2007, and the M.S. and Ph.D. degrees in computer science from the University of Aizu, Aizu-Wakamatsu, Japan, in 2009 and 2013, respectively.

He is currently a Full Professor with the School of Computer Science, China University of Geosciences, Wuhan, China. His current research interests include edge computing, cloud computing, and future networking technologies. He has authored three books and published over 120 papers in refereed journals and conferences in these areas.

Prof. Zeng serves for the editorial boards of the *Journal of Network and Computer Applications*, *Frontiers of Computer Science*, and *Open Journal of Computer Society*. He is a Senior Member of CCF.

Pingzhi Fan (Fellow, IEEE) received the Ph.D. degree in electronic engineering from the University of Hull, Kingston upon Hull, U.K., in 1994.

He is currently a Distinguished Professor with Southwest Jiaotong University, Chengdu, China, and has been a Visiting Professor with Leeds University, Leeds, U.K., since 1997. His research interests include high-mobility wireless communications, massive random-access techniques, and signal design and coding.

Dr. Fan is a recipient of the U.K. ORS Award in 1992, the NSFC Outstanding Young Scientist Award in 1998, the IEEE VTS Jack Neubauer Memorial Award in 2018, the IEEE SP Soc SPL Best Paper Award in 2018, the IEEE WCSP 10-Year Anniversary Excellent Paper Award from 2009 to 2019, and the IEEE/CIC ICCC Best Paper Award in 2020. He is an IEEE VTS Distinguished Speaker from 2019 to 2022, and a Fellow of IET, CIE, and CIC.