QoE-DEER: A QoE-Aware Decentralized Resource Allocation Scheme for Edge Computing

Journal:	*Transactions on Services Computing*
Manuscript ID	TSC-2020-04-0127
Manuscript Types:	Regular Paper
Keywords:	Edge Computing, Quality of Experience (QoE), Scheduling, Resource Allocation, Game Theory

QoE-DEER: A QoE-Aware Decentralized Resource Allocation Scheme for Edge Computing

Songyuan Li, Jiwei Huang, *Member, IEEE,* and Bo Cheng, *Member, IEEE,* and Junliang Chen

Abstract—With the increasing prevalence of online services mounted on IoT devices, edge computing gains significant momentum over conventional cloud-centric architecture. Differing from monolithic architecture of cloud computing, edge servers are geographically deployed in a distributed manner nearby IoT devices, which not only frees online services from the high hardware requirement but also sharply reduces network latency experienced by end users. Recent works have extensively studied the problem of edge resource management and request scheduling to achieve high Quality of Service (QoS) with low latency, but there has been little focus on Quality of Experience (QoE) that an edge resource allocation scheme brings about. In this paper, we study the problem of Edge Resource Allocation (ERA) across multiple service requests with the objective of overall QoE maximization, which has non-polynomial computational complexity. To attack the NP-hardness of solving the ERA problem, we adopt a game-theoretic approach to formulate the ERA problem as a potential game ERAGame which admits a Nash Equilibrium (NE). Then, we novelly present a decentralized algorithm namely QoE-DEER to find an NE solution which equivalently maximizes the overall QoE as the ERA problem. The performance and convergence of our algorithm is analyzed and evaluated both theoretically and experimentally.

Index Terms—Edge Computing, Quality of Experience (QoE), Scheduling, Resource Allocation, Game Theory.

1 Introduction

With the growing popularity of Internet of Things (IoT), a wide variety of IoT devices, including mobile phones, wearable devices, sensors, etc., pervade every aspect of people's life. The rapid growth of IoT devices promotes the sophistication of software services, especially for the online services [1] [2] which need to interact information with remote servers. With remote servers assisted, the online service gets freed from the high hardware requirement and thus caters to the IoT devices with constrained processing capabilities. Nevertheless, network latency caused by interaction with remote servers is a considerable threat and bottleneck affecting the Quality of Service (QoS) throughout a service lifecycle. If the overall service latency including computational and network latency is more than 100 milliseconds, end users will perceive the response delay [3], or negative outcomes may be resulted in for latency-sensitive services [4]. Hence, cloud-centric infrastructure is not sufficiently qualified to host services with a strict low-latency demand, as a sizeable proportion of cloud users would experience the overall latency over 100 milliseconds [5] [6]. It necessitates an emerging computing architecture to support online services in the IoT scenario.

Edge computing is a burgeoning computing paradigm customized to IoT environment [7] [8]. Processing capabilities of numerous IoT devices are augmented by means of the resources of geo-distributed edge servers [7]. Edge servers are geographically placed in close proximity to end devices, and service providers deploy their online services on edge servers to interact with end users in real time. Since edge servers are placed much closer to end devices than the remote cloud, network latency between end devices and servers is sharply reduced compared with cloud-assisted operating model. Thanks to the decline of network latency, the QoS level is substantially upgraded with a much lower latency within the service lifecycle including data transmission and edge processing.

Edge resource management is a critical research topic in edge computing [9], which jointly considers request scheduling and resource allocation on edge servers. On the one hand, since the signal coverage areas of nearby edge servers usually partially overlap to avoid the non-serviceable area [10], thus request scheduling strategy should be delicately designed, especially for the users located in the overlapped area where there exist multiple accessible edge servers for selection. On the other hand, an edge server typically has relatively limited computational resources in comparison with a cloud data-center server, hence driving a requirement to build a more elaborate edge resource allocation strategy. As a joint effort of request scheduling and edge resource allocation strategy, the overall QoS level experienced by end users is upgraded.

Quality of Experience (QoE) is an important metric assessing the user satisfactory. It evaluates the degree of user satisfaction when experiencing a software service [11], which gains great significance especially for end-user services like multiplayer online games [12]. Although there have been previous works on edge computing concerning various topics of cost efficiency [13] [14], service latency minimization [15] and energy efficiency [16], the resource allocation problem for QoE optimization in edge computing still remains largely unexplored. Therefore, it is worthwhile

- *S. Li, B. Cheng, J. Chen are with the the State Key Laboratory of Networking and Switching Technology,Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: {lisy, chengbo, chjl}@bupt.edu.cn).*
- *J. Huang is with Department of Computer Science and Technology, Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China (corresponding author, e-mail: huangjw@cup.edu.cn).*

Manuscript received April 12, 2020.

to further investigate the QoE optimization issue in the context of edge computing.

In this paper, we study the Edge Resource Allocation (ERA) problem with the objective of overall QoE maximization across multiple service requests. Multiple end users concurrently propose edge requests and competes each other for limited edge resources. We formulate the ERA problem as a bin packing problem, which determines each service request's target edge server together with the amount of edge resources for allocation. Due to the NP-hardness [17] of solving the bin packing problem and the inherent nonlinearity of QoE, it is difficult to solve the ERA problem in a centralized way.

To attack these challenges, we adapt ERAGame, a game-theoretical approach to find out the solution of ERA problem in a decentralized manner. In the ERAGame, each end user is modeled as a strategic player who makes independent decision on resource allocation with the purpose of maximizing its own QoE. ERAGame alleviates the burden of centralized optimization through enabling each end user to individually determine their own edge resource allocation, while the collective objective of QoE maximization is sufficiently guaranteed. Furthermore, ERAGame allows each end user to personalize its own unique QoE metric. Benefited from the distributed nature of ERAGame, a decentralized edge resource allocation algorithm namely QoE-DEER is carefully designed to find out a Nash Equilibrium (NE) of ERAGame which is proved to equivalently maximize overall QoE as the ERA problem.

Original contributions of our work are mainly summarized as follows.

- For edge computing environment, we formulate the ERA problem as a bin packing problem, wishing to maximize overall QoE across concurrent service requests. It is NP-hard to conduct centralized optimization to solve the ERA problem.
- To enable the ERA problem solved in a decentralized manner, we formulate it as a game namely ERAGame. Given the non-unique existence of Nash Equilibriums, we propose a decision mechanism named preemption with the purpose of making the ERAGame converge to a high-overall-QoE state.
- Through combining the ERAGame with preemption mechanism, a decentralized algorithm (QoE-DEER) is developed, while a cooperative messaging mechanism is designed to make the QoE-DEER algorithm practical in reality.
- The performance and convergence of our QoE-DEER algorithm is evaluated from both theoretical analysis and experimental validation. Our QoE-DEER algorithm is proved to find out an equivalent solution as the centralized solution for the ERA problem, while outperforming state-of-art approaches in experimental results.

The rest of this paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we introduce our system model. In Section 4, we investigate the correlation between QoS and QoE with the QoE model given, based on which our optimization problem is formulated. Then, Section 5 depicts the design of ERAGame. Then, Section 6 develops a QoE-aware decentralized edge resource allocation algorithm namely QoE-DEER, and then theoretically analyzes the convergence and performance of QoE-DEER algorithm. In Section 7, we conduct experiments based on real-world dataset to evaluate our approach. In Section 8, we conclude our work.

2 Related Work

2.1 Edge Computing

With the emergence of IoT technology and applications, edge computing has gradually been a widely-focused research problem, especially the scheduling and resource allocation problem in edge environment. Regarding the distributed infrastructure of edge computing, Castellano et al. [18] proposed a decentralized edge resource orchestration algorithm where heterogeneous optimization criteria of each application were fulfilled. Ma et al. [19] concentrated on the cost-effective resource allocation problem in the edge-cloud environment with various service requirements and diverse resource instances regarded. Chen et al. [16] studied the multi-user multi-task computation offloading problem for green mobile edge-cloud systems, where the energy harvesting strategy of wireless edge devices was under consideration. Ma et al. [15] designed a cooperative service caching and workload scheduling algorithm in edge environment with the objective of response time minimization, where edge resources were sufficiently utilized to host maximum service requests. Li et al. [20] designed the dynamic task scheduling and resource provision algorithm for the edge computing paradigm which achieved energy efficiency together with the Service Level Agreement guaranteed. Chen et al. [21] jointly considered the task offloading and frequency scaling problem in the mobile edge environment which dynamically balanced the tradeoff between energy efficiency and task queue delay over time periods.

Game theory is a powerful tool for the design of decentralized scheduling and resource allocation algorithms in edge computing environment. He et al. [13] formulated the cost-effective service scheduling solution in edge as a Nash Equilibrium, where the decentralized algorithm to solve out the Nash Equilibrium was given. Ma et al. [22] characterized the joint cooperation and competition of edge resource allocation as a a three-sided cyclic game involving users, edge nodes, and service providers, where the equilibria solution could be calculated in a decentralized manner. Nguyen et al. [23] adopted the non-cooperative game theory to design a price-incentive resource allocation mechanism for edge computing which improved the edge resource utilization and satisfied the budget balance. Josilo et al. [24] developed a Starkelberg game to formulate the interaction between end devices and the operator who jointly managed the wireless and computational resources in the edge computing system, based on which the decentralized algorithm on task offloading decisions is proposed to achieve the cost-minimizing/time-fair resource allocation. Chen et al. [25] formulated the edge resource allocation problem as a Starkelberg game with the objective of balancing the utility between end users and edge providers, and put forward a computation-efficient decentralized algorithm for edge resource allocation.

TABLE 1
Summary of Key Notions

Symbol	Definition
$\mathcal{U}$	Set of M IoT devices/end users, i.e., $\{u_1, ..., u_i, ..., u_N\}$
$\mathcal{E}$	Set of N edge servers, i.e., $\{e_1, ..., e_j, ..., e_M\}$
$\mathcal{E}_i$	Set of accessible edge servers by user u_i
$\mathcal{U}_j^E$	Set of end users covered by edge server e_j
h_i	Number of CPU cycles used for processing user u_i's service request
η_i	Data size of user u_i's service request
B	Channel bandwidth of wireless network
c_j	Number of CPU resources at edge server e_j
$d_{i,j}$	Distance between IoT device/user u_i and edge server e_j
$r_{i,j}$	Data transmission rate from u_j to e_j
f_i^L	CPU frequency of IoT device u_i
f_j^E	CPU frequency per unit CPU resource at edge server e_j
x_i	Index of edge server that user u_i's service request is scheduled to
a_i	Number of edge resources allocated for user u_i's service request
$\mathbf{s}_i$	Strategy of user u_i, i.e., (x_i, a_i)
$\mathbf{S}_i$	Set of feasible strategies for user u_i
$\mathbf{s}$	Strategy profile for $\mathcal{U}$, i.e., $(\mathbf{s}_1, ..., \mathbf{s}_N)$
$\mathbf{S}$	Set of feasible strategy profiles for $\mathcal{U}$
QoE_i^L	QoE gained by user u_i through local computing
QoE_i^E	QoE gained by user u_i through edge computing
π_i	QoE gained by user u_i according to $\mathbf{s}_i$

2.2 Quality of Experience

Since the era of cloud computing, QoE management and optimization has been a significant but challenging issue in the research community. Zhou [26] evaluated the QoE metric as a subjective criterion called Mean Option Score (MOS) which reflects the user's attitude towards the gained response time, based on which a delay announcement mechanism for QoE enhancement is proposed. Chemodanov et al. [27] studied the QoE-aware cloud service orchestration policy for service providers, where the QoE metric indicated the degree of satisfactory scored by the users. With regard to the multi-source video delivery services in cloud, Saleem et al. [28] jointly considered the video quality, bitrate level and quality variations as the factors influencing the QoE, and then designed a cooperative multi-source selection and rate adaption algorithm for QoE optimization. Hong et al. [29] studied the QoE metric in frame rate and processing delay in terms of the cloud gaming service, and then put forward a VM placement algorithm which struck the balance between the QoE and the provider's net profit.

There are also several literatures having started studying on QoE improvement in edge environment. Lin et al. [12] constructed an IoT-Fog-Cloud infrastructure for massively multiplayer online game services, where the supernode in the Fog tier was worked as the edge server responsible for processing the service requests from nearby IoT devices. According to the IoT-Fog-Cloud infrastructure, they designed the request scheduling strategy to enhance the user satisfaction and experience. Mahmud et al. [30] evaluated the QoE metric in the respect of service delivery, based on which a QoE-aware service placement scheme in edge environment is brought up. Hong et al. [31] measured the QoE of IoT devices from multiple aspects of economic efficiency, battery efficiency and response efficiency, and thus proposed an offloading strategy for edge computing aimed to QoE optimization.

In spite of several existent works concerning the QoE issue in edge environment, rare literature in edge computing but [32] studied the quantitative correlation between QoS and QoE to conduct QoE optimization. If the correlation between QoS and QoE is properly formulated, then the quantitative evaluation for the QoE metric can be obtained. Lai et al. [32] studied the correlation between QoS and QoE, and adapted the greedy approach to maximize QoE in edge environment. Therefore, we intend to strengthen the research of QoE optimization in edge environment. We analyze the quantitative correlation between QoS and QoE, and then take advantage of the game theory to solve the QoE maximization problem in a decentralized manner. The game-theoretical solution provides the optimality and convergence guarantee for QoE maximization.

3 System Model

We consider an edge computing system consisting of N end users and M edge servers. The finite set of N end users is denoted by $\mathcal{U} = \{u_1, ..., u_N\}$, while the finite set of M edge servers is represented by $\mathcal{E} = \{e_1, ..., e_M\}$. Multiple end users holding IoT devices propose concurrent service requests to their nearby edge servers for processing. Edge servers are geographically placed based on user distribution, with a limited signal coverage area covering a group of end users. In our scenario, each service request is either processed locally at the IoT device, or delivered to a nearby edge server for further processing.

Request: Without loss of generality, we assume that each end user u_i proposes a single service request, and the user who proposes multiple service requests can be regarded as a group of users. The number of CPU cycles required for processing user u_i's service request is h_i. Meanwhile, if end user u_i chooses to process its service request at edge, then u_i should transmit data to edge with the data size of η_i. Furthermore, given the limited signal coverage area of edge servers, let $\mathcal{E}_i$ represent the set of accessible edge servers by end user u_i, and we also define $\mathcal{U}_j$ as the set of end users covered by edge server e_j.

Resource: We assume that each IoT device has limited resources with the local computational capacity (i.e., CPU frequency) of f_j^L. Besides, we consider the scenario that each edge server e_j is equipped with c_j units of CPU resources. More complicated scenario of multidimensional edge resources (e.g., RAM, storage) will be explored in our future work. Let f_j^E denote the computational capacity (i.e., CPU frequency) per unit CPU resource for edge server e_j.

Network: The IoT device accesses and interacts with edge server through the wireless network whose bandwidth is B, thus we apply the Shannon formula to estimate the data transmission rate between the IoT device and server. The communication distance between the IoT device/user u_i and edge server e_j is $d_{i,j}$, and the reference received signal-to-noise ratio for user u_i per unit communication distance is λ_i^0. Therefore, according to the Shannon formula,

data transmission rate from u_i to e_j can be calculated in (1), as in [33].

$$r_{i,j} = B\log_2(1+\frac{\lambda_i^0}{d_{i,j}^2}) \quad (1)$$

Note that $\lambda_i^0 = \gamma_0 \cdot P_i/\sigma^2$, where γ_0 is the channel power per unit distance, P_i is the transmitting power for each IoT device/user u_i, and σ^2 is the noise power of the wireless environment.

4 Problem Statement

4.1 Correlation between QoS and QoE

With the increasing number of end users, each user is highly likely to have differentiated QoS requirement, ranging from low to high. On the one hand, service requests from the user with strong response susceptibility (e.g., online video streaming delivery) usually have a high QoS demand, thus requiring to consume more computational resources to satisfy the user. On the other hand, service requests from the user with moderate response susceptibility have a medium QoS demand, in needless of many computational resources to make end user satisfied. Specifically take the video analytic service [4] for example, most of service requests merely needs service latency below one second, while other service requests strictly expects of service latency within 100 milliseconds. Therefore, Quality of Experience (QoE), which reflects the user satisfactory, varies across different users with differentiated QoS requirements.

Pertinent literatures on QoE [34] [35] [36] have discovered that a quantitative correlation exists between QoS and QoE, as depicted in Figure 1. The QoE level is by no means proportional to the QoS level (i.e., service latency). Meaning, having a lower service latency improves the QoE level, but such an improvement tends to converge from a critical point (e.g., P2 of Figure 1) where the QoE keeps virtually unchanged in close proximity to the highest QoE level, with no regard to a noticeable increase in the QoS level. Generally, the QoE initially increases at slow growth with the reduction of service latency until one point (e.g., P1 of Figure 1). Then, the QoE keeps steady increase until P2 in Figure 1; after that, the QoE improvement is marginally diminishing.

4.2 QoS Model

Given the correlation between QoS and QoE narrated above, we firstly analyze the QoS gained by users. Service latency, which is one of the most important QoS attributes, is adopted as the QoS metric in this paper. Let $\boldsymbol{s}_i = (x_i, a_i)$ denote the strategy on edge resource allocation for user u_i, where x_i represents the edge server e_{x_i} selected by user u_i for edge processing, and a_i represents the amount of edge resources allocated for user u_i. If user u_i decides to process its own service request locally, (x_i, a_i) is defined as $(0, 0)$.

When user u_i conducts local computing with $x_i = 0$ and $a_i = 0$, then u_i will process its service request with its own resources at IoT device. Therefore, service latency for user u_i's service request by local computing is formulated by (2).

$$t_i^L = \frac{h_i}{f_i^L} \quad (2)$$

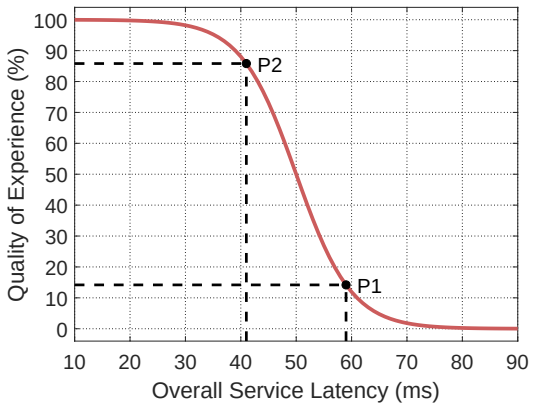


Fig. 1. Correlation between Quality of Experience and Quality of Service.

When user u_i conducts edge computing based on $\boldsymbol{s}_i = (x_i, a_i)$, the computational latency at edge server e_{x_i} with a_i units of CPU resources allocated is formulated by (3), where the computational capacity for a_i units of CPU resources at edge server e_{x_i} is indicated by $(a_i \cdot f_{x_i}^E)$. The computational capacity for a_i units of CPU resources conforms to the additivity of unit computational capacity $f_{x_i}^E$, which can be implemented by Round Robin CPU scheduling.

$$t_{i,x_i}^{E,cmp}(a_i) = \frac{h_i}{a_i \cdot f_{x_i}^E} \quad (3)$$

Besides, the network latency by data transmission from IoT device/user u_i to edge server e_{x_i} is formulated by (4).

$$t_{i,x_i}^{E,off} = \frac{\eta_i}{r_{i,x_i}} \quad (4)$$

Given (3) and (4), overall service latency for user u_i's service request by edge computing based on $\boldsymbol{s}_i = (x_i, a_i)$ is formulated by (5).

$$t_{i,x_i}^{E}(a_i) = t_{i,x_i}^{E,off} + t_{i,x_i}^{E,cmp}(a_i) \quad (5)$$

4.3 QoE Model

As stated previously, the QoE is non-linearly correlated with the QoS. Related literatures [37] [38] [39] apply the sigmoid function to formulate the correlation between QoS and QoE, thereby we use the logistic function, which is a generalization of sigmoid function, to quantify the correlation between QoS and QoE. Logistic function strengthens the quantitative ability for QoE, including QoE growth rate and basic QoE requirement.

The QoE metric is usually evaluated according to various levels, thus we apply the percentum to evaluate the QoE metric in this paper. Each user can gain the maximum QoE of 100%. When user u_i conducts local computing with $x_i = 0$ and $a_i = 0$, then u_i will gain the QoE, which is formulated based on the logistic function as (6).

$$QoE_i^L = \frac{1}{1+e^{\alpha_i(t_i^L-\beta_i)}} \quad (6)$$

where α_i represents the QoE growth rate for user u_i, and β_i represents the mid-point of QoE function for user u_i.

Practically, β_i implies how much QoS (i.e., service latency) should be obtained to acquire the 50% of QoE.

In similar, when user u_i conducts edge computing based on $\boldsymbol{s}_i = (x_i, a_i)$, the QoE gained by user u_i is given as (7).

$$QoE_i^E(\mathbf{s}_i) = \frac{1}{1 + e^{\alpha_i\left(t^E_{i,x_i}(a_i) - \beta_i\right)}} \tag{7}$$

4.4 Optimization Problem

In our optimization problem, we target the overall QoE maximization across multiple end users. We firstly define the user utility, based on the QoE formulation (6)-(7).

When user u_i selects to conduct local computing with $x_i = 0$ and $a_i = 0$, then u_i has the user utility as (8), indicating the QoE gained by local computing.

$$\pi_i(\boldsymbol{s}_i) = QoE_i^L \tag{8}$$

When user u_i selects to conduct edge computing based on $\boldsymbol{s}_i = (x_i, a_i)$, then its user utility is formulated as (9).

$$\pi_i(\boldsymbol{s}_i) = \begin{cases} QoE_i^E & \text{if } QoE_i^E > QoE_i^L \\ QoE_i^L & \text{otherwise} \end{cases} \tag{9}$$

The formulation on user utility (9) embodies the selection rule between local computing and edge computing, which is specifically considered into two cases. Compared with local computing, if the user u_i gains more QoE through its decision on edge resource allocation $\boldsymbol{s}_i = (x_i, a_i)$, then u_i will determine to conduct edge computing according to $\boldsymbol{s}_i$, thereby having the user utility of QoE_i^E which indicates the gained QoE by edge computing. Otherwise, the user u_i will determine to process its own service request locally without any occupancy of edge resources (i.e., $x_i = 0$, $a_i = 0$), thus holding the user utility of QoE_i^L which is the gained QoE by local computing.

With user utility $\pi_i(\boldsymbol{s}_i)$ carefully defined, our optimization problem called Edge Resource Allocation (ERA) problem is given as follows, with the objective of overall QoE maximization (10), subject to (11)-(12). Note that $I_{\{condition\}}$ is an indicator function returning 1 when the *condition* is true, otherwise 0.

$$\max_{x_i, a_i} \sum_{i \in \mathcal{U}} \pi_i(\boldsymbol{s}_i) \tag{10}$$

$$\sum_{i \in \mathcal{U}_j} a_i \cdot I_{\{x_i = j\}} \leq c_j \quad \forall j \in \mathcal{E} \tag{11}$$

$$x_i \in \{0\} \cup \mathcal{E}_i,\ a_i \geq 0 \quad \forall i \in \mathcal{U} \tag{12}$$

The resource constraints (11) ensure that each edge server e_j cannot allocate users with edge resources exceeding its resource capacity of c_j. And the user constraints (12) imply that each user u_i can either be scheduled to one of its accessible edge servers, or process its own request locally.

It can be found that our ERA problem affiliates to the family of bin packing problem which is NP-hard to solve in a centralized manner [17]. Each edge server is conceived as a bin with finite available resources, and our objective is to determine the edge resource allocation scheme of each service request with the overall QoE maximized. Besides, the optimization objective (10) formulated by QoE functions (see Eq. (6), (7)) is in the nonlinear form, making the complexity of problem solving increased as well [40]. Hence, it highly requires an efficient and effective solving approach.

5 Edge Resource Allocation Game

5.1 Game Formulation

To attack the intractability of our ERA problem, game theory is introduced to reduce the centralization of optimization and enable each user a certain degree of autonomy. Specifically, each user is allowed to make individual decision on edge resource allocation based on its own QoE interest, thereby facilitating our ERA problem solved in a decentralized and efficient manner. The definition of *QoE-Incentive ERAGame* is given in Definition 1. Note that, in our game formulation, all possible strategies $\mathbf{s}_i$ for user u_i form up the set of feasible strategies $\mathbf{S}_i$.

Definition 1 (QoE-Incentive ERAGame). *A strategic game $\mathcal{G}$ formulating the edge resource competition amongst N users is defined by a triple $\langle\, \mathcal{U}, (\boldsymbol{S}_i)_{u_i \in \mathcal{U}}, (\pi_i)_{u_i \in \mathcal{U}} \,\rangle$ such that*

- *$\mathcal{U}$ is the set of players which specifically refer to N users here. Players compete against each other to gain a higher QoE via obtaining more edge resource allocation.*
- *$\boldsymbol{S}_i$ is the set of feasible strategies for user u_i. Its strategy $\boldsymbol{s}_i \in \boldsymbol{S}_i$ specifies the edge server where the user u_i's service request is scheduled, together with the amount of edge resources allocated for request processing.*
- *π_i is the utility function of user u_i, formulated by (8) or (9), to evaluate the QoE obtained by user u_i via adopting a strategy $\boldsymbol{s}_i \in \boldsymbol{S}_i$.*

In our ERAGame, users would like to be allocated more edge resources with the purpose of gaining higher QoE. With the limited capacity of edge resources, however, each user has to compete for the limited edge resources with other users and selects a strategy $\mathbf{s}_i \in \mathbf{S}_i$ aimed to maximize its own utility (i.e., QoE). Strategy $\mathbf{s}_i$ selected by each user u_i makes up the *strategy profile* $\mathbf{s} = (\mathbf{s}_1, ..., \mathbf{s}_N)$. During the game phase, suppose that a user u_i originally selects a strategy $\mathbf{s}_i$, but finds another feasible strategy $\mathbf{s}_i'$ which gains higher utility (i.e., QoE). Incentivised by the benefit of this discovery, user u_i would naturally expect to update its strategy decision as $\mathbf{s}_i'$.

However, resulted from the finite resource supply of edge servers, there might be conflicts amongst users each of which wishes to monopolize much more edge resources gaining higher utility (i.e., QoE), hence giving rise to resource competition. To mitigate the conflicts amongst users, the concept of Nash equilibrium (NE) [41] is applied to manage the competitive behavior amongst users. The definition of NE is given in Definition 2.

Definition 2 (Nash Equlibrium). *A Nash Equilibrium for the ERAGame $\mathcal{G} = \langle\, \mathcal{U}, (\boldsymbol{S}_i)_{u_i \in \mathcal{U}}, (\pi_i)_{u_i \in \mathcal{U}} \,\rangle$ is a strategy profile $\boldsymbol{s}^*$ satisfying that for each player $i \in \mathcal{U}$,*

$$\pi_i(\boldsymbol{s}_i^*, \boldsymbol{s}_{-i}^*) \geq \pi_i(\boldsymbol{s}_i, \boldsymbol{s}_{-i}^*),\ \forall \boldsymbol{s}_i \in \boldsymbol{S}_i. \tag{13}$$

Note that, $\mathbf{s}_{-i}$ denote the strategy profile except for user u_i, and thus π_i is extended from $\pi_i(\mathbf{s}_i)$ to $\pi_i(\mathbf{s}_i, \mathbf{s}_{-i})$ in the game formulation, representing the resource competition amongst multiple users. Edge resources supplied for users are in the finite number, hence any user cannot arbitrarily increase its edge resource allocation, not caring the edge resource usage of other competitive users. In addition, we also define $\pi_{-i}(\mathbf{s}_i, \mathbf{s}_{-i})$ as the overall QoE gained by all users except u_i under the strategy profile $\mathbf{s} = (\mathbf{s}_i, \mathbf{s}_{-i})$.

5.2 Preemption Mechanism

Given the nonunique existence of NE [41], we put forward a preemption mechanism on strategy decision to make the ERAGame admit at an NE with high-overall-QoE state.

If there are idle edge resources which are free to be allocated, the user will naturally decide to occupy idle edge resources, thereby gaining higher QoE. Nevertheless, suppose that all edge resources that a user u_i can access to have been occupied by other users, then no more edge resource can be allocated to the user u_i. Hence, a *preemption* mechanism is required to decide whether to allocate a certain amount of edge resources to user u_i at the cost of reducing the corresponding amount of resources from other users.

The user u_i is the one who attempts to preempt resources from other users. Specifically, let $\triangle a_i$ denote the number of edge resource units that user u_i wants to preempt, and $p(u_i)$ represent the set of users that could be preempted by user u_i. Then, the user u_i gains the QoE improvement of $\triangle\pi_i$ through preempting $\triangle a_i$ units of edge resources from users $u_k \in p(u_i)$, while the preempted users $u_k \in p(u_i)$ are totally at the minimum loss of $\triangle\pi_{-i}$ on QoE. Given above, the preemption mechanism is formally defined in Definition 3. Note that, after preemption, strategies of users but $\{u_i\} \cap p(u_i)$ remain unchanged.

Definition 3 (Preemption Mechanism). *Any user u_i will partially preempt the edge resources held by users $u_k \in p(u_i)$, if the overall QoE is upgraded after preemption, i.e.,*

$$\triangle\pi_i > \triangle\pi_{-i} \tag{14}$$

The condition (14) triggering the preemption ensures that, the QoE improvement of user u_i (denoted by $\triangle\pi_i$) overcomes the simultaneous QoE decrease of users $u_k \in p(u_i)$ (denoted by $\triangle\pi_{-i}$). Besides, since strategies of the users except $\{u_i\} \cap p(u_i)$ are kept unchanged during the preemption, the QoE of users except $\{u_i\} \cap p(u_i)$ would not be affected by preemption. Thus, only QoE of users $u_k \in p(u_i)$ is reduced because of preemption. Therefore, the overall QoE is increased after preemption, exactly coinciding with the intention of making the ERAGame converge to a high-overall-QoE state.

5.3 Preemption-Based QoE Improve Algorithm (PRIM)

Given our preemption mechanism, we propose a PReemption-based QoE IMprovement algorithm (PRIM) to implement the above preemption mechanism, as shown in Algorithm 1. In our PRIM algorithm design, on the one hand, if there are enough edge resources available for user u_i to update its strategy with $\mathbf{s}'_i$ gaining higher QoE, then the strategy $\mathbf{s}'_i$ will be adopted. On the other hand, if no enough edge resources are available for user u_i to adopt the new strategy $\mathbf{s}'_i$, then the minimum QoE decrease $\triangle\pi_{-i}$ of users $p(u_i) = \{u_k \in \mathcal{U} : x_k = x'_i \ and \ k \neq i\}$ should be calculated and compared with $\triangle\pi_i$ to determine whether to conduct preemption.

It is worth noting that, the minimum QoE decrease $\triangle\pi_{-i}$ for users $p(u_i)$ can be calculated within finite iterations in an efficient computational complexity. Specifically speaking, in the first iteration, we sort up the users $u_k \in p(u_i)$, who make edge resources insufficient for user u_i to adopt the new strategy $\mathbf{s}'_i$, by $\hat{\pi}_k$ in rank P. Here, $\hat{\pi}_k$ represents the minimum QoE decrease of user u_k when its one unit of edge resources is preempted, formulated as (15). The above sorting process can be implemented by quick-sorting with the complexity of $O(m \log m)$, where $m = |p(u_i)|$ is generally less than N. Then, we pick out the user u_v who has the minimum QoE decrease in P and temporarily release one unit of edge resources from user u_v.

$$\hat{\pi}_k = \begin{cases} \pi_k(\mathbf{s}_k) - \pi_i(x_k, a_k - 1) & \text{if } a_i \geq 2 \\ \pi_k(\mathbf{s}_k) - \pi_i(0, 0) & \text{otherwise} \end{cases} \tag{15}$$

Next, the second iteration starts up. Again, users $u_k \in p(u_i)$ are reordered by $\hat{\pi}_k$, where $\hat{\pi}_v$ for the user u_v has been reevaluated. The reordering operation can be performed on the previously ordered rank P, only needing to locate the insert position of user u_v in P. Such reordering process can be implemented with the complexity of $O(m)$. Having users reordered, we take the similar previous approach to pick out one of users with one unit of edge resources temporarily released. Such iterative process repeats until $\triangle a_i$ units of edge resources released from from users $p(u_i)$. While the iterative process is terminated, the minimum QoE decrease $\triangle\pi_{-i}$ for users $p(u_i)$ is obtained to testify whether the preemption condition (14) is satisfied. If satisfied, then $\triangle a_i$ units of edge resources temporarily released before will be confirmed to be preempted by user u_i. Since the sort operations contribute the majority of computational

Algorithm 1: Preemption-Based QoE Improve Algorithm (PRIM)

Input: a strategy $\mathbf{s}'_i$ that user u_i attempts to update, current strategy profile $\mathbf{s}$.
Output: updated strategy profile $\mathbf{s}'$.

```
if π_i(s'_i) > π_i(s_i) then
    if a'_i + Σ_{u_k∈U:x_k=x'_i,k≠i} a_k ≤ c_{x'_i} then
        return s' ← {s'_i, s_{-i}};
    else
        △a_i ← a'_i − c_{x'_i} + Σ_{u_k∈U:x_k=x'_i,k≠i} a_k;
        Order users u_k ∈ U : x_k = x'_i and k ≠ i by π̂_k
        in an ascending rank P;
        Initialized s'_{-i} ← s_{-i};
        while △a_i > 0 do
            Select the top-ranked user u_v in P;
            Try to update s'_v ← (x_v, a_v − 1);
            Reorder u_k ∈ U : x_k = x'_i and k ≠ i by π̂_k,
            with the update value of π̂_v;
            △a_i ← △a_i − 1;
        △π_i ← π_i(s'_i, s'_{-i}) − π_i(s_i, s_{-i});
        △π_{-i} ← π_{-i}(s_i, s_{-i}) − π_{-i}(s'_i, s'_{-i});
        if △π_i > △π_{-i} then
            return s' ← {s'_i, s'_{-i}};
        else
            return s' ← s;
else
    return s' ← s;
```

complexity, thus the PRIM algorithm has the computational complexity of $O(m(\log m + \triangle a_i - 1))$.

6 Decentralized Algorithm

6.1 Algorithm Design

Based on the PRIM algorithm, a QoE-aware DEcentralized Edge Resource allocation algorithm (QoE-DEER) is put forward to find out an NE solution for ERAGame with high-overall-QoE state. The NE solution with high-overall-QoE state specifies the edge resource allocation scheme for each user. To make the process of edge resource allocation conducted in a decentralized manner, a *cooperative messaging mechanism* is firstly designed, on which the QoE-DEER algorithm is given. The message types involved in the *cooperative messaging mechanism* are listed as follows. They are applied to maintain communications between users and edge servers.

- *Begin Message (BM):* the ERAGame does not reach an NE, as long as there is at least one strategy requested for update. Here, each edge server e_j will broadcast the BM to its affiliated users $u_i \in \mathcal{U}_j$ to continue the ERAGame.
- *Information Message (IM):* while each edge server e_j broadcasts the BM, the edge server e_j informs its affiliated users $u_i \in \mathcal{U}_j$ of edge resource allocation state of other users $u \in \cup_{j \in \mathcal{E}_i} U_j$, via sending the IM.
- *Strategy Message (SM):* with the IM received, each user u_i decides whether to request for strategy change. If requesting, then u_i will send SM to the relevant edge server, aimed to acquire permission for strategy change.
- *Allow Message (AM):* in our game design, only one user is allowed for strategy change in each decision iteration. Thus, all edge servers will negotiate and send only one AM to one user who is granted permission for strategy change. The negotiation is conducted based on all-come-then-improve policy.
- *Update Message (UM):* once after determining which one of users is permitted for strategy change, all edge servers should also notify the other users whose allocated resources are partially preempted via sending the UM massage.

Given these messages, the QoE-DEER algorithm is given, as shown in Algorithm 2. Note that, $\triangle t$ is predefined to denote the required time to transmit a message between edge servers and end users. To clarify the message protocol running in the QoE-DEER algorithm, a flow graph for the message passing in each time-slot is illustrated in Fig. 2. The QoE-DEER algorithm runs in each time slot until the game process terminates and consists of the following three phases.

Phase 1 (Lines 1-3): if the ERAGame has not achieved at the NE, all edge servers will respectively broadcast the BM to their own affiliated users, to continue the game. Meanwhile, each edge server updates and packs the current status of edge resource allocation as an IM message. And the IM is sent to the affiliated users of each edge server, providing users reference information to conduct strategy change.

Phase 2 (Lines 4-10): with sufficient IMs received, each user u_i will invoke the PRIM algorithm to find whether there is a strategy profile $\mathbf{s}'$ gaining higher QoE than current $\pi_i(\mathbf{s}_i)$ and $\sum_{u_k \in \mathcal{U}} \pi_k(\mathbf{s}_k)$. If such a strategy profile $\mathbf{s}'$ exists, then u_i will send SM to the edge server specified by $\mathbf{s}'_i$, requesting permission for updating the strategy profile with $\mathbf{s}'$.

Phase 3 (Lines 11-18): if any user requests for strategy change, edge servers won't receive SMs later than $\triangle t$. After $\triangle t$, edge servers should receive all SMs and negotiate each other to decide which of requests for strategy change is adopted. The negotiation process is based on the all-come-then-improve policy; in specific, the strategy-change request $\mathbf{s}^*$ which gains the greatest overall-QoE improvement is elected out amongst all requests. Then, $\mathbf{s}^*$ is adopted. Here, an AM is sent back to the user requesting $\mathbf{s}^*$ as the SM response, while UMs are sent to the users whose part of allocated resources are preempted. On the contrast, if no SM received within $\triangle t$, the ERAGame has reached an NE,

Algorithm 2: QoE-Aware Decentralized Edge Resource Allocation Algorithm (QoE-DEER)

Input: strategy profile $\mathbf{s}(t)$ in the time slot t.
Output: strategy profile $\mathbf{s}(t+1)$ in the time slot $t+1$.

1 Initially set $\mathbf{s}(t+1) \leftarrow \mathbf{s}(t)$;
2 **Phase 1: Continue the game**
3 Each edge server broadcasts BM and IM to its affiliated users;
4 **Phase 2: Acquire the next strategy**
5 **for** *each user* $i \in \mathcal{U}$ **do**
6 Set a variable $\mathbf{s}' \leftarrow \mathbf{s}(t)$;
7 **for** *each feasible strategy* $\hat{\mathbf{s}}_i \in \boldsymbol{S}_i$ **do**
8 $\mathbf{s}' \leftarrow PRIM(\hat{\mathbf{s}}_i, \mathbf{s}')$;
9 **if** $\boldsymbol{s}' \neq \boldsymbol{s}(t)$ **then**
10 Send SM to the edge server specified by $\mathbf{s}'_i$;
11 **Phase 3: Select one requested strategy to improve**
12 **if** *any edge server receives SM in* $\triangle t$ **then**
13 Pick out the requested strategy $\mathbf{s}^*$ which achieves the greatest QoE improvement;
14 Adopt $\mathbf{s}(t+1) \leftarrow \mathbf{s}^*$;
15 Send AM to the user who requested to update $\mathbf{s}^*$;
16 Send UMs to the user whose allocated resources are partially preempted;
17 **else**
18 **return** $\boldsymbol{s}(t+1)$; ▷ Reach Nash Equilibrium.

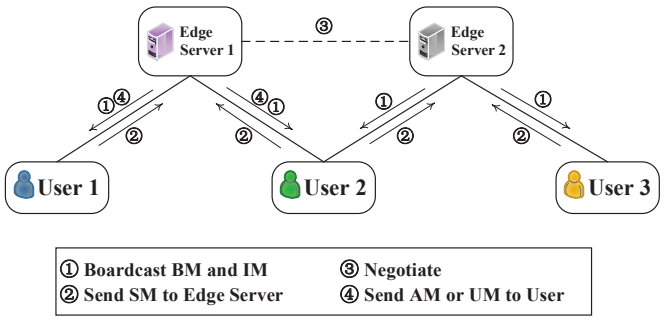


Fig. 2. Flow graph for the message passing in each time-slot decision.

thereby the QoE-DEER algorithm terminates.

In real-world scenarios, our QoE-DEER algorithm for each time-slot decision can be approximately accomplished within the maximum time of $3\triangle t$. In *Phase 1*, each edge server can boardcast BMs and send IMs to its affiliated users in parallel, which takes at most $\triangle t$. Then, in *Phase 2*, if requesting for strategy change, the user will send the SM to edge server; the SM won't received by edge server later than $\triangle t$. In *Phase 3*, with sufficient SMs received, the edge server adopts one of strategy-change requests with an AM sent back, and sends the UMs to inform the users that their allocated resources are partially preempted; the SM and UMs should be delivered at the respective target user within $\triangle t$. To sum up the above three phases, our QoE-DEER algorithm is estimated to be accomplished at the time cost of $3\triangle t$.

6.2 Algorithm Analysis

6.2.1 Convergence Analysis

We investigate whether our QoE-PEER algorithm can converge at an NE within the finite number of decision iterations. Since our QoE-DEER algorithm is the decentralized implementation of ERAGame, thus we just need to justify whether our ERAGame can admit at an NE within the finite number of iterations. The *Finite Improvement Property* is an important property for potential games, indicating that an NE of potential games can be reached via a process going through a finite number of iterations [42]. Therefore, the convergence of QoE-PEER algorithm is ensured if we prove the ERAGame as a potential game. The definition of potential game is firstly given as follows.

Definition 4 (Potential Game). *A game is a potential game, if there exists a potential function $\Phi : \boldsymbol{S} \to \mathbb{R}$ such that for each player $u_i \in \mathcal{U}$, $\Phi(\boldsymbol{s}') > \Phi(\boldsymbol{s})$ holds for any strategy improvement from $\boldsymbol{s}$ to $\boldsymbol{s}'$ satisfying $\pi_i(\boldsymbol{s}'_i) > \pi_i(\boldsymbol{s}_i)$, where $\boldsymbol{s} = (\boldsymbol{s}_i, \boldsymbol{s}_{-i})$ and $\boldsymbol{s}' = (\boldsymbol{s}'_i, \boldsymbol{s}'_{-i})$.*

Note that, since preemption is enabled to reach an NE with high-overall-QoE state in our game formulation, thus the strategy $\mathbf{s}_i$ of more than one users could be simultaneously changed within each strategy improvement from $\mathbf{s}$ to $\mathbf{s}'$. According to Definition 4, the NE $\mathbf{s}^*$ in the ERAGame can be interpreted in such a way that for any user $u_i \in \mathcal{U}$, $\pi_i(\mathbf{s}^*_i, \mathbf{s}^*_{-i}) = max_{\mathbf{s}_i \in \mathcal{S}_i} \pi_i(\mathbf{s}_i, \mathbf{s}^*_{-i})$, which guarantees the existence of NE by seeking out the optima of the potential function [43]. The potential function monotonically increases with each strategy improvement until reaching the optima of the potential function which represents the NE, where the *Finite Improvement Property* of potential games is empowered. Such a useful property of potential games has been leveraged by [13] [22] [44]. We constructively prove the ERAGame as a potential game in Theorem 1, thus the potential function $\Phi(\mathbf{s})$ is pre-defined in (16).

$$\Phi(\mathbf{s}) = \sum_{u_i \in \mathcal{U}} \pi_i(\mathbf{s}_i) \tag{16}$$

Theorem 1 (ERA Potential Game). *The ERAGame is a potential game with the potential function of $\Phi(\boldsymbol{s})$.*

Proof. Suppose that a user $i \in \mathcal{U}$ changes its strategy from $\mathbf{s}_i$ to $\mathbf{s}'_i$, fulfilling the statement of $\pi_i(\mathbf{s}'_i, \mathbf{s}'_{-i}) > \pi_i(\mathbf{s}_i, \mathbf{s}_{-i})$. To prove Theorem 1 true, we need to testify that $\Phi(\mathbf{s}') > \Phi(\mathbf{s})$. Note that $\mathbf{s}_i = (x_i, a_i)$, $\mathbf{s}'_i = (x'_i, a'_i)$.

According to the preemption mechanism, user u_i may need to conduct resource preemption to change its strategy from $\mathbf{s}_i$ to $\mathbf{s}'_i$. In specific, edge resources occupied by other users $\{u_k \in \mathcal{U} : x_k = x'_i \ and \ k \neq i\}$ may be partially preempted. Here, we apply $p(u_i)$ to denote the set of users preempted by u_i. Assume that any user $u_v \in p(u_i)$ originally adopting the strategy $\mathbf{s}_v$ would have to take the strategy as $\mathbf{s}'_v$ after preemption, resulting in a QoE decrease represented by $\pi_i(\mathbf{s}_v) - \pi_i(\mathbf{s}'_v)$. The total QoE decrease for all users $u_v \in p(u_i)$ is $\sum_{u_v \in p(u_i)} (\pi_i(\mathbf{s}_v) - \pi_i(\mathbf{s}'_v))$.

Given the triggering condition (14), resource preemption is allowed only when

$$\pi_i(\mathbf{s}'_i) - \pi_i(\mathbf{s}_i) > \sum_{u_v \in p(u_i)} (\pi_i(\mathbf{s}_v) - \pi_i(\mathbf{s}'_v))$$

which follows that

$$\begin{aligned}
&\Phi(\mathbf{s}') - \Phi(\mathbf{s}) \\
&= (\pi_i(\mathbf{s}'_i) + \sum_{u_v \in p(u_i)} \pi_i(\mathbf{s}'_v)) - (\pi_i(\mathbf{s}_i) + \sum_{u_v \in p(u_i)} \pi_i(\mathbf{s}'_v)) \\
&= (\pi_i(\mathbf{s}'_i) - \pi_i(\mathbf{s}_i)) + \sum_{u_v \in p(u_i)} (\pi_i(\mathbf{s}'_v) - \pi_i(\mathbf{s}'_v)) > 0
\end{aligned}$$

To summarize, $\Phi(\mathbf{s}') - \Phi(\mathbf{s}) > 0$ always holds if $\pi_i(\mathbf{s}') - \pi_i(\mathbf{s}) > 0$, confirming our ERAGame as a potential game with the potential function of $\Phi(\mathbf{s})$. □

With our ERAGame proved as a potential game, it is ensured that our QoE-PEER algorithm can coverage at an NE within the finite number of decision iterations.

6.2.2 Performance Analysis

We analyze the performance of QoE-DEER algorithm from the aspect of optimality, judging whether the QoE-DEER algorithm equivalently solves the ERA problem.

As discussed in Section 4.4, our original ERA problem targets the overall QoE maximization across multiple users. Because of the intractability of our ERA problem, we design the ERAGame which tries to solve the ERA problem in a decentralized manner. Nevertheless, there might be multiple NEs in the ERAGame [41]. Thus, it is worthy to investigate whether the ERAGame eventually achieves the NE solution equivalently maximizing the overall QoE as the ERA problem. Theorem 2 has proved the equivalent optimality as follows.

Theorem 2. *The QoE-DEER algorithm can reach the optimal state of edge allocation where the total QoE is maximized.*

Proof. Let $\mathbf{s} = \{\mathbf{s}_1, ..., \mathbf{s}_n\}$ denote the strategy profile $\mathbf{s}$ solved by the decentralized QoE-DEER algorithm, and $\mathbf{s}^* = \{\mathbf{s}^*_1, ..., \mathbf{s}^*_n\}$ to denote the optimal strategy profile which maximizes the total QoE as $\sum_{u_i \in \mathcal{U}} \pi_i(\mathbf{s}^*_i)$.

We prove Theorem 2 by reduction to absurdity. Assume that the strategy profile $\mathbf{s}$ cannot achieve the maximum total QoE, implying $\sum_{u_i \in \mathcal{U}} \pi_i(\mathbf{s}_i) < \sum_{u_i \in \mathcal{U}} \pi_i(\mathbf{s}^*_i)$. Then, there must exist some users who gain a higher QoE in $\mathbf{s}^*_i$ than in $\mathbf{s}_i$. Thus, all users are divided into two groups $\mathcal{G}_1$ and $\mathcal{G}_2$.

The first group $\mathcal{G}_1$ comprises the users who gain a higher QoE in $\mathbf{s}^*_i$ than in $\mathbf{s}_i$, implying $\sum_{u_i \in \mathcal{G}_1} \pi_i(\mathbf{s}^*_i) >$

$\sum_{u_i \in \mathcal{G}_1} \pi_i(\mathbf{s}_i)$. The QoE increment for these users in $\mathcal{G}_1$ is represented by $\Delta I = \sum_{u_i \in \mathcal{G}_1} (\pi_i(\mathbf{s}_i^*) - \pi_i(\mathbf{s}_i))$.

The second group $\mathcal{G}_2$ comprises the remaining users. For these users, in the worst case, their QoE may be degraded from $\mathbf{s}$ to $\mathbf{s}^*$, i.e., $\sum_{u_i \in \mathcal{G}_2} \pi_i(\mathbf{s}_i^*) \leq \sum_{u_i \in \mathcal{G}_2} \pi_i(\mathbf{s}_i)$. The QoE decrement is correspondingly represented by $\Delta D = \sum_{u_i \in \mathcal{G}_2} (\pi_i(\mathbf{s}_i) - \pi_i(\mathbf{s}_i^*))$.

Given the assumption that $\sum_{u_i \in \mathcal{U}} \pi_i(\mathbf{s}_i) < \sum_{u_i \in \mathcal{U}} \pi_i(\mathbf{s}_i^*)$, the QoE increment brought by users in $\mathcal{G}_1$ should be greater than the QoE decrement brought by users in $\mathcal{G}_2$, i.e., $\Delta I > \Delta D$, triggering the preemption condition (14) further upgrading the overall QoE. Note that, the decentralized QoE-DEER algorithm would not quit the loop until no user can update its strategy to increase the total QoE, even with preemption. In other words, $\mathbf{s}$ is not the final strategy profile solved by QoE-DEER algorithm, which implies a contradiction with the initial assumption. Therefore, the decentralized QoE-DEER algorithm can always maximize the total QoE as the ERA problem. □

7 Evaluation

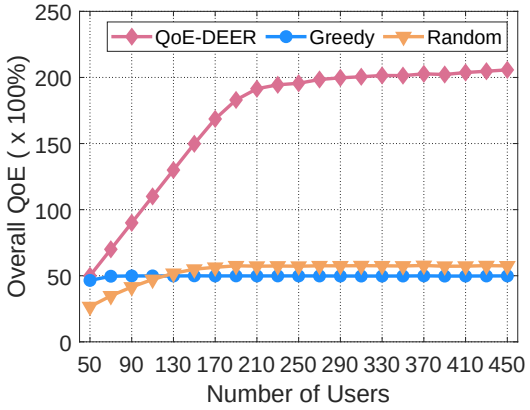
7.1 Experimental Setup

We simulate an edge environment with multiple end users and edge servers in MATLAB, where the EUA dataset [45] is adopted. The EUA dataset records the geographical information of cellular base stations and end users within the Melbourne central business district area in Australia, at the format of longitude and latitude. The geographical information of all cellular base stations in the EUA dataset is drawn from the radio-comms license dataset published by the Australian Communications and Media Authority (ACMA). The IP address blocks allocated to Australia is provided by the Asia Pacific Network Information Centre (APNIC). The geographical location of users is derived through the IP lookup service http://ip-api.com/, which converts the IP address of users into the corresponding geographical location. Since edge servers are normally deployed near cellular base stations [46], thus the geographical information of cellular base stations is used as the location of edge servers in our experimental setting, while the latitude-longitude information of end users in the EUA dataset is straightforwardly adopted here.

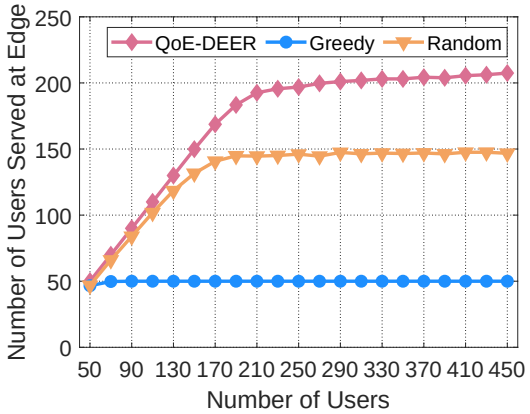
For the edge server, the signal coverage radius of each edge server e_j is randomly set between 450 and 750 meters, and the CPU frequency per unit of edge resources f_j^E is set as 10 GHz. The number of available edge resources c_j is diversely set in various experimental scenarios, ranging from 6 to 26.

For the end user, each user u_i can conduct local computing at the CPU frequency f_i^L, which is drawn from a uniform distribution across $[1.5, 2]$ GHz. Meanwhile, each user u_i generates its QoE-related parameters α_i and β_i respectively following the normal distribution $\mathcal{N}(1.5, 0.25^2)$ and $\mathcal{N}(75, 10^2)$. Besides, the data size η_i of user u_i's service request are drawn from the uniform distribution at the range of [150,200] KB, which approximately matches the size of a photo. The number of CPU cycles per data size for the service request is 5.

For the wireless communication between users and edge servers, we set the transmitting power P_i for each IoT device/user u_i as 0.5 W, and the channel bandwidth B as 10 MHz [47] [48]. The noise power of wireless environment τ^2 is also set as -87 dBm [49], while the channel power per unit communication distance γ_0 is set as -50 dB [33] [50].



(a) Overall Quality of Experience.



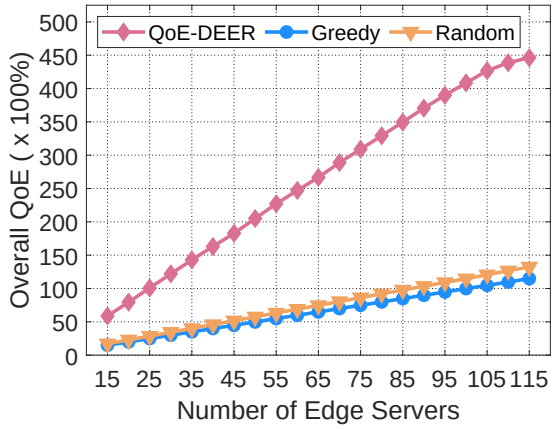
(b) Number of users served at edge.

Fig. 3. NE performance vs. Number of users N.

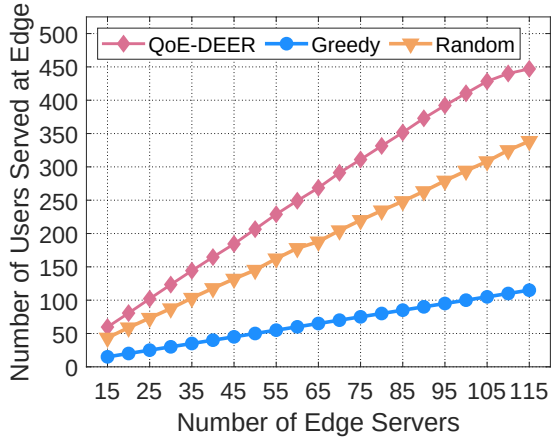
7.2 NE Performance

To evaluate the effectiveness of our QoE-DEER algorithm, we investigate the performance of NE at which the QoE-DEER algorithm finally converges. Given our edge environment, the NE performance is studied from two aspects, which are *overall QoE* and *number of users served at edge*. Besides, we compare our QoE-DEER algorithm with two baseline solutions, which are *Greedy* and *Random*.

- *Greedy:* service requests of users are conducted scheduling and resource allocation on edge according to the non-increasing order by α_i. Each scheduled user is allocated edge resources which brings its maximum QoE, while unscheduled users (because of finite edge resources) select to conduct local computing.
- *Random:* service requests of users are conducted scheduling and resource allocation on edge following a random rank. According to the random rank, each scheduled user should randomly select the target edge server x_i and edge resource allocation x_i. If all accessible resources have been occupied by other users, then the user will

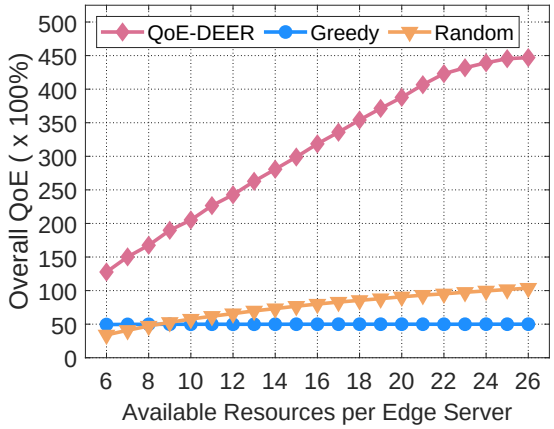


(a) Overall Quality of Experience.

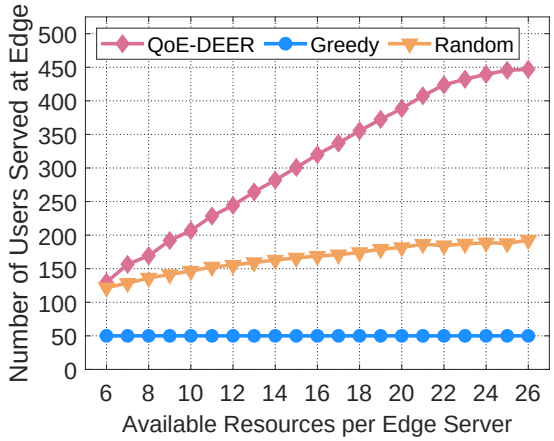


(b) Number of users served at edge.

Fig. 4. NE performance vs. Number of edge servers M.



(a) Overall Quality of Experience.



(b) Number of users served at edge.

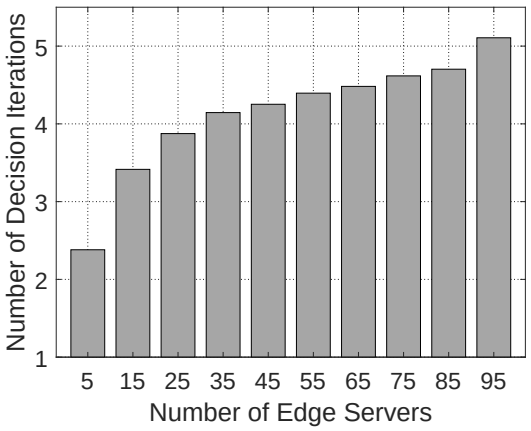
Fig. 5. NE performance vs. Available resources per edge server c_j.

have to conduct local computing with no edge resources allocated.

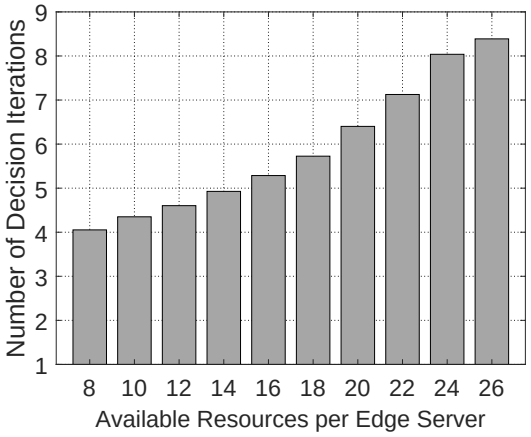
Note that, the following experimental results correlated to the *Greedy* algorithm are averaged over 60 runs, and the following experimental results correlated to the *Random* algorithm are averaged over 7,200 runs. In each run, we randomly a specified number of edge servers and users with different geographical locations from the EUA dataset. Because it is NP-hard to solve the ERA problem as mentioned in Section 4.4, solving out the optimal solution in a centralized way cannot be finished within a rational time especially when the problem scale is large, hence unable to be applied as a baseline used for evaluating our QoE-DEER algorithm.

First, we evaluate the NE performance through adjusting the number of users N from 50 to 450, while the number of edge servers M and the number of available edge resources per edge server c_j are respectively fixed as 50 and 10. The experimental results on NE performance are illustrated in Figure 3. Under various settings on user scale, our QoE-DEER algorithm always gains the highest overall QoE and the greatest number of users served at edge. Moreover, our QoE-DEER algorithm continuously expands the advantage on NE performance, with the increase of user scale. Constrained by limited edge resource supply, our QoE-DEER algorithm cannot enable much more users served at edge after the user scale develops to a certain degree, i.e., 210 users in Figure 3. Hence, the overall QoE will not increase significantly thereupon. Note that, under the *Greedy* algorithm, overall QoE and number of users served at edge remain almost unchanged although the total number of users scales up. This is because, each user greedily takes up edge resources; once a user is scheduled to an edge server, then the user will monopolize all available edge resources. At the initial user scale in Figure 3 (i.e., 50 users), all available edge resources have been utilized with no idle resources left for more users.

Second, we tune the number of edge servers M from 15 to 115 as well, with N and c_j respectively set as 450 and 10. The experimental results on NE performance are shown in Figure 4. Compared with the two baseline algorithms, our QoE-DEER algorithm gains the highest overall QoE and the greatest number of users served at edge, although they all present an increase trend on NE performance with the expanding scale of edge servers. More edge servers engaging into the edge environment implies a better capacity to serve user requests at edge, thus the number of users served at edge grows up as shown in Figure 4(b).



(a) Convergence vs. M.



(b) Convergence vs. c_j.

Fig. 6. Convergence of QoE-DEER algorithm.

TABLE 2
QoE-DEER Convergence vs. Number of Users N

Number of Users	50	130	210	290	380	450
Decision Iterations	4.629	6.419	5.646	4.643	4.455	4.337

With more users served at edge which enables higher QoE than local computing, the overall QoE gets improved, as shown in Figure 4(a). Remarkably, the rate of growth on NE performance has a slight decrease when the number of edge servers is more than 105, because nearly all 450 users have been served at edge with their QoE optimized at the almost highest. Continuing to increase edge servers cannot further improve their QoE.

Third, we also explore the effect of edge server's computational capacity on NE performance, as depicted in Figure 5. In specific, the number of available edge resources per edge server c_j is adjusted from 6 to 26, while M and N is set as 450 and 500. Under various settings on c_j, our QoE-DEER algorithm always outperforms the two other baseline algorithms on both overall QoE and the number of users served at edge, and such the superiority held by QoE-DEER algorithm is gradually strengthened with the increasing setting on c_j, especially compared with the *Greedy* algorithm. In the *Greedy* algorithm, each scheduled user would still monopolize all of available resources on its scheduled server with no idle resources reserved for other users, no matter how many available resources c_j is set at each edge server. Hence, as for the *Greedy* algorithm, it is of little help to improve the overall QoE through enabling a greater c_j per edge server. Note that, when $c_j > 22$, the increase rate on NE performance for our QoE-DEER algorithm slows down. This is because, nearly all 450 users have been scheduled to be served at edge servers, with their QoE maximized at almost the highest level. Even though supplying more edge resources, their QoE cannot be virtually upgraded furthermore, but resulting in the waste of idle edge resources.

7.3 Algorithmic Convergence

We then experimentally analyze the convergence of our QoE-DEER algorithm. The convergence of our QoE-DEER algorithm has been proved in Theorem 1. To measure the convergence of our QoE-DEER algorithm, we adopt the number of decision iteration required for finding the final NE solution. Similar to Section 7.2, we evaluate the algorithmic convergence under diverse experimental settings, varied by different number of users, edge servers, or available resources per edge server, as shown in Fig. 6 and Table 2. Each experimental result representing the number of decision iterations is averaged over 1,000 runs. In each run, a specified number of edge servers and users are selected from the EUA dataset, which have various geographical locations.

Figure 6(a) describes the QoE-DEER convergence with the number of edge servers. Since more edge servers participating in the edge resource allocation implies the enlargement of strategy space $\mathbf{S}_i$ for each user, hence lowering down the algorithmic convergence rate. Thus, the number of decision iterations increases slowly with the number of edge servers.

Figure 6(b) shows the QoE-DEER convergence with the number of available resources per edge server (i.e., c_j). As c_j increases, the number of decision iterations increases linearly as well. When $c_j = 8$, it has the number of decision iterations as 4.053. While c_j is set as 26, there is the number of decision iterations as 8.388. Under different settings on c_j, the number of decision iterations is always acceptable to make our QoE-DEER algorithm into practical use.

Table 2 lists the QoE-DEER convergence with the number of users. No obvious trend can be obtained with the number of users, but the number of decision iterations keeps low with the average iterations of 5.0215. It follows that, our QoE-DEER algorithms is efficient, acceptable in reality.

8 Conclusion

In this paper, we propose a novel game-theoretic approach which solves the Edge Resource Allocation (ERA) problem maximizing the overall QoE across multiple edge users. To attack the intractability of our ERA problem, we formulate the ERA problem as a potential game named ERAGame, where each user chooses its decision on edge resource allocation based on its QoE interests. Given the ERAGame,

the ERA problem can be solved in a decentralized manner, where the preemption mechanism is leveraged to make the ERAGame reach the Nash Equilibrium which equivalently maximizes the overall QoE as the ERA problem. A cooperative message mechanism is designed to make our decentralized approach QoE-DEER appliable. The optimality and convergence of our decentralized approach are analyzed theoretically and verified experimentally.

There are several avenues for our future work. On the one hand, dynamic edge resource management scheme might be designed based on the basic idea proposed in this paper. It can more effectively handle various service requests arriving at different times, and a more generalized QoE optimization objective which considers the temporal dynamics might be put forward. On the other hand, another revenue of our future work is to deploy our approach in real-life IoT environments like unmanned industrial robotic systems. Experimental results derived from real-world IoT scenario should provide us with important reference on user/system behaviors, thereby prompting further algorithmic optimization.

Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 61972414), Beijing Natural Science Foundation (No. 4202066), National Key Research and Development Program of China (No. 2018YFB1003800), and Fundamental Research Funds for Central Universities (No. 2462018YJRC040).

References

[1] M. Raja, V. Ghaderi, and S. Sigg, "WiBot! In-vehicle behaviour and gesture recognition using wireless network edge," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 376–387.

[2] T. Braud, F. H. Bijarbooneh, D. Chatzopoulos, and P. Hui, "Future networking challenges: The case of mobile augmented reality," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 1796–1807.

[3] M. Jarschel, D. Schlosser, S. Scheuring, and T. HoBfeld, "An evaluation of QoE in cloud gaming based on subjective tests," in *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2011, pp. 330–335.

[4] G. Ananthanarayanan, P. Bahl, P. Bodłk, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[5] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Annual Workshop on Network and Systems Support for Games (NetGames)*, 2012, pp. 1–6.

[6] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing public cloud providers," in *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2010, pp. 1–14.

[7] W. Shi, G. Pallis, and Z. Xu, "Edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1474–1481, 2019.

[8] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Transactions on Cloud Computing*, 2019, DOI: 10.1109/TCC.2019.2898657.

[9] K. M. Sim, "Intelligent resource management in intercloud, fog, and edge: Tutorial and new directions," *IEEE Transactions on Services Computing*, 2020, DOI: 10.1109/TSC.2020.2975168.

[10] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[11] K. Brunnström, S. A. Beker, and K. DeMoor et al., "Qualinet white paper on definitions of Quality of Experience," 2013.

[12] Y. Lin and H. Shen, "CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 431–445, 2017.

[13] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, to appear, 2019, DOI: 10.1109/TPDS.2019.2938944.

[14] L. Kuang, T. Gong, S. OuYang, H. Gao, and S. Deng, "Offloading decision methods for multiple users with structured tasks in edge computing for smart cities," *Future Generation Computer Systems*, vol. 105, pp. 717–729, April 2020.

[15] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *IEEE International Conference on Computer Communications (INFOCOM)*, to appear, 2020.

[16] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 726–738, 2019.

[17] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," *Computers and Intractability*, vol. 340, 1979.

[18] G. Castellano, F. Esposito, and F. Risso, "A distributed orchestration algorithm for edge computing resources with guarantees," in *IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 2548–2556.

[19] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Transactions on Cloud Computing*, 2019, DOI: 10.1109/TCC.2019.2903240.

[20] S. Li and J. Huang, "Energy efficient resource management and task scheduling for IoT services in edge computing paradigm," in *IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2017, pp. 846–851.

[21] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "TOFFEE: task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, 2019, DOI: 10.1109/TCC.2019.2923692.

[22] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for service-oriented resource allocation in edge computing," *IEEE Transactions on Services Computing*, 2020, DOI: 10.1109/TSC.2020.2966196.

[23] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Transactions on Cloud Computing*, 2018, DOI: 10.1109/TCC.2018.2844379.

[24] S. Josilo and G. Dan, "Joint management of wireless and computing resources for computation offloading in mobile edge clouds," *IEEE Transactions on Cloud Computing*, 2019, DOI: 10.1109/TCC.2019.2923768.

[25] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, vol. 108, pp. 273–287, July 2020.

[26] L. Zhou, "QoE-driven delay announcement for cloud mobile media," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 1, pp. 84–94, 2017.

[27] D. Chemodanov, P. Calyam, S. Valluripally, H. Trinh, J. Patman, and K. Palaniappan, "On qoe-oriented cloud service orchestration for application providers," *IEEE Transactions on Services Computing*, 2018, DOI: 10.1109/TSC.2018.2866851.

[28] M. Saleem, Y. Saleem, and M. Hayat, "Stochastic QoE-aware optimization of multisource multimedia content delivery for mobile cloud," *Cluster Computing*, 2019, DOI: 10.1007/s10586-019-03007-y.

[29] H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu, "Placing virtual machines to optimize cloud gaming experience," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 42–53, 2015.

[30] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya,

"Quality of experience (QoE)-aware placement of applications in fog computing environments," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, October 2018.

[31] S. Hong and H. Kim, "QoE-aware computation offloading to capture energy-latency-pricing tradeoff in mobile clouds," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 2174–2189, 2019.

[32] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *International Conference on Service-Oriented Computing (ICSOC)*, 2019, pp. 86–101.

[33] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[34] M. Alreshoodi and J. Woods, "Survey on QoE/QoS correlation models for multimedia services," *International Journal of Distributed and Parallel Systems*, vol. 4, no. 3, pp. 53–72, 2013.

[35] M. Fiedler, T. Hossfeld, and P. TranGia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.

[36] T. Hobfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of YouTube QoE via crowdsourcing," in *IEEE International Symposium on Multimedia (ISM)*, 2011, pp. 494–499.

[37] M. Hemmati, B. McCormick, and S. Shirmohammadi, "QoE-aware bandwidth allocation for video traffic using sigmoidal programming," *IEEE MultiMedia*, vol. 24, no. 4, pp. 80–90, 2017.

[38] P. Hande, S. Zhang, and M. Chiang, "Distributed rate allocation for inelastic flows," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1240–1253, 2007.

[39] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[40] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[41] M. J. Osborne and A. Rubinstein, "A course in game theory, MIT Press," 1994.

[42] D. Monderer and L. S. Shapley, "Potential games," *Games & Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[43] J. R. Marden, G. Arslan, and J. S. Shamma, "Cooperative control and potential games," *IEEE Transactions on Systems, Man and Cybernetics Part B (Cybernetics)*, vol. 39, no. 6, pp. 1393–1407, 2009.

[44] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoTcedgecloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[45] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *International Conference on Service-Oriented Computing (ICSOC)*, 2018, pp. 230–245.

[46] Y. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: a key technology towards 5G," *ETSI White Paper*, pp. 1–16, 2015.

[47] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

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

[49] E. Tanghe, W. Joseph, L. Verloock, L. Martens, H. Capoen, K. V. Herwegen, and W. Vantomme, "The industrial indoor channel: large-scale and temporal fading at 900, 2400, and 5200 mhz," *IEEE Transactions on Wireless Communications*, vol. 7, no. 7, pp. 2740–2751, 2008.

[50] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," *Concurrency & Computation Practice & Experience*, 2016, DOI: 10.1002/cpe.3975.

Songyuan Li is currently a master candidate in the State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunication. He received the B.Eng. degree from Beijing University of Posts and Telecommunications in 2018. He was the recipient of China National Scholarship in 2019. He has published articles in international journals and conference proceedings, including the PEER-TO-PEER NETWORKING AND APPLICATIONS, IEEE ICWS, IEEE SCC, and IEEE ISPA. His current research interests include cloud computing, services computing, performance evaluation, and optimization.

Jiwei Huang received the B.Eng. and Ph.D. degrees in computer science and technology from Tsinghua University, in 2009 and 2014, respectively. He was a Visiting Scholar with the Georgia Institute of Technology. He is currently an Associate Professor and the Dean of the Department of Computer Science and Technology, China University of Petroleum, Beijing. He has published one book and more than 40 articles in international journals and conference proceedings, including the IEEE TRANSACTIONS ON SERVICES COMPUTING, the IEEE TRANSACTIONS ON CLOUD COMPUTING, ACM SIGMETRICS, IEEE ICWS, and IEEE SCC. His research interests include services computing, cloud computing, and performance evaluation. He is a member of the IEEE and ACM.

Bo Cheng received the Ph.D. degree in computer science and engineering from University of Electronic Science and Technology of China in 2006. He is now a professor in the State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunications. His current research interests include network services and intelligence, Internet of Things technology, communication software and distributed computing, etc. He is a member of the IEEE and ACM.

Junliang Chen received the B.S. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1955, and the Ph.D. degree in electrical engineering from the Moscow Institute of Electrical Telecommunication, Moscow, former Soviet Union, in 1961. He was a Visiting Scholar at the University of California, Berkeley and the University of California, Los Angeles from 1979 to 1981. He has been with the Beijing University of Posts and Telecommunications, Beijing, China, since 1955, where he is currently the Chairman and a professor with the Institute of Networking Technology. His current research interests include communication networks and next-generation service creation technology. He was elected as a member of the Chinese Academy of Sciences in 1991 and a member of the Chinese Academy of Engineering in 1994 for his contributions to fault diagnosis in stored program control exchange. He received the First, Second, and Third prizes of the National Scientific and Technological Progress Award in 1988, 2004, and 1999, respectively.