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

Songyuan Li, *Student Member, IEEE*, Jiwei Huang, *Member, IEEE*, Jia Hu, *Member, IEEE*, and Bo Cheng, *Member, IEEE*

***Abstract*—With the increasing prevalence of online services mounted on IoT devices, edge computing gains significant momentum over conventional cloud-centric architecture. 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 IoT 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 article, we study the Edge Resource Allocation (ERA) problem across multiple service requests with the objective of overall QoE maximization, which has non-polynomial computational complexity. To attack the $\mathcal{NP}$-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. Finally, the performance and convergence of our algorithm is evaluated both theoretically and experimentally, which indicates its significant advantages over the state-of-the-art approaches.**

***Index Terms*—Edge computing, quality of experience (QoE), scheduling, resource allocation, game theory.**

I. 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 interacting information with upstream servers in an efficient manner. Given this, the traditional cloud-centric paradigm is called for an evolution towards a burgeoning computing architecture namely edge computing [3], [4]. Unlike the monolithic architecture of cloud computing, edge servers are geographically deployed in close proximity to diversified IoT devices, and service providers place their online services on edge servers to interact with IoT users in real time. In this way, processing capabilities of IoT devices can be considerably augmented by nearby edge servers. In addition to this, numerous service requests from IoT devices can be efficiently processed in a decentralized manner by geo-distributed edge severs, therefore sharply reducing the network congestion caused by massive IoT service requests. Taking it by and large, edge computing significantly boosts the processing efficiency of IoT service requests, with the overall service latency including computational and network latency broadly reduced. Edge computing has been perceived as a promising technical solution for the IoT application scenario.

Edge resource management is a critical research issue in edge computing [5], [6], which jointly considers request scheduling and resource allocation on edge servers. Although many previous works have thoughtfully studied on the edge resource management problem, there still exist several research avenues for further in-depth study. *Firstly*, since various IoT users[1] propose their service requests from separate geographical locations [7], edge servers are thereupon deployed at varied regions based on the service popularity of each geographical district [8]. For this reason, a centralized resource allocation scheme, such as [9]–[17], cannot accommodate itself to the large-scale distributed edge computing environment. Instead, it necessitates a decentralized implementation which effectively supports the distributed edge resource management. *Secondly*, several existing studies on edge computing have explored the resource management problem with a specified Quality-of-Service (QoS) optimization objective, such as, cost efficiency [18], [19], service latency minimization [9], energy efficiency [12], [13], revenue maximization [14], [20], etc. Nevertheless, the edge resource management scheme considering the impact of Quality of Experience (QoE) still needs further extensive investigations.

QoE is widely perceived as a user-centric indicator which evaluates the user's satisfaction when experiencing a software service [21]. As defined by the International

Manuscript received January 7, 2021; revised August 29, 2021; accepted September 29, 2021. Date of publication October 7, 2021; date of current version June 9, 2022. This work is supported by Beijing Nova Program (No. Z201100006820082), National Natural Science Foundation of China (Nos. 61972414, 61902029, 61973161), Beijing Natural Science Foundation (No. 4202066), National Key Research and Development Program of China (No. 2018YFB1003800), and the Fundamental Research Funds for Central Universities (No. 2462018YJRC040). The associate editor coordinating the review of this article and approving it for publication was A. Zanella. *(Corresponding author: Jiwei Huang.)*

Songyuan Li and Jia Hu are with the Department of Computer Science, College of Engineering, Mathematics, and Physical Sciences, University of Exeter, Exeter EX4 4PY, U.K. (e-mail: lisy@ieee.org; j.hu@exeter.ac.uk).

Jiwei Huang is with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China (e-mail: huangjw@cup.edu.cn).

Bo Cheng is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: chengbo@bupt.edu.cn).

Digital Object Identifier 10.1109/TCCN.2021.3118460

[1]In this article, we speak interchangeably of an IoT device and an IoT user.

Telecommunication Union (ITU-T), QoE generally represents "the overall acceptability of an application or service subjectively perceived by the end-user". Therefore, with more services and applications developed according to the human-centered design, service providers gradually place greater importance on the QoE improvement. In contrast to many QoS-driven resource allocation methods, a QoE-aware resource allocation approach can directly strengthen the perceived quality by users, thereby more effectively consolidating the user loyalty and reducing the service relinquish rate. As for the edge computing environment, each edge server typically has limited computational resources [22], [23]. Thus, it becomes even more essential to elaborately distribute edge resources amongst IoT users, making the most users' interest fulfilled. An effective QoE-aware edge resource management scheme is exactly anticipated to better optimize the utilization of limited edge resources to improve the overall user satisfaction. It emphatically prevents each IoT user from greedily occupying excessive edge resources [16], [24]. The additional edge resources allocated beyond one's requirement contribute very little towards further improving its own user satisfaction, but undertake the cost of downgrading the service satisfaction of other IoT users instead.

In this article, we put forward a decentralized QoE-aware resource allocation scheme for edge computing. To be specific, we think of the QoE-aware Edge Resource Allocation (ERA) problem from the perspective of service providers who lease edge resources to serve their own IoT users. We adopt a fine-grained edge resource management strategy to solve the ERA problem, attentively studying the amount of edge resources partitioned for each IoT service request served at a shared edge server. Through conducting the overall QoE maximization across multiple IoT users, our ERA scheme can enhance the user's engagement and loyalty towards service providers. In the considered scenario, multiple IoT users concurrently submit service requests and compete against each other for limited edge resources. We formulate the ERA problem which determines each IoT service request's target edge server together with the amount of edge resources for allocation. Arising from the $\mathcal{NP}$-hardness [25] of our ERA problem and the inherent nonlinearity of QoE functions, it is with great computational intractability to solve the ERA problem in a classical centralized manner.

Considering the computational challenge, we adapt ERAGame, a game-theoretical approach to find out the solution of our ERA problem in a decentralized way. Each IoT user is modeled as a strategic player who makes independent decision on resource allocation with the purpose of maximizing its own QoE. In this way, ERAGame alleviates the burden of centralized optimization through enabling each IoT user to individually determine their own edge resource allocation, while the collective objective of QoE maximization is sufficiently guaranteed. Benefited from the distributed nature of ERAGame, a decentralized ERA algorithm namely QoE-DEER is carefully designed to find out a Nash Equilibrium (NE) of ERAGame. Our QoE-DEER algorithm is theoretically proved to provide a solution equivalent to the centralized one for the ERA problem, within the finite decision iterations. Thanks to its decentralized implementation, our QoE-DEER algorithm demonstrates substantial performance and scalability in the large-scale distributed edge computing environment, making itself applicable in realistic scenarios. The original contributions of this article include:

1) We formulate the QoE-aware Edge Resource Allocation (ERA) problem as a potential game [26] termed ERAGame, which aims to achieve overall QoE maximization. Among the non-unique Nash Equilibrium (NE) solutions, an edge resource preemption rule is incorporated into ERAGame, making the ERAGame converge to the NE solution with the optimal overall QoE.
2) Based on the formulation of ERAGame, a decentralized ERA algorithm named QoE-DEER is developed, while a cooperative messaging mechanism is designed to make the QoE-DEER algorithm implementable in reality. The performance and convergence of our QoE-DEER algorithm are both theoretically proved and analyzed.
3) Our decentralized QoE-DEER algorithm is evaluated through simulations based on the realistic EUA dataset [27] by using the *iFogSim* [28] toolkit. The experimental results demonstrate that our QoE-DEER algorithm outperforms the state-of-the-art QoS and QoE-aware approaches.

The rest of this article is organized as follows. In Section II, we investigate the related work. In Section III, we introduce our system model. In Section IV, we present the QoE model for edge computing, based on which the QoE-aware ERA problem is formulated. Then, Section V depicts the game-theoretical design of ERAGame. In Section VI, we develop a QoE-aware decentralized ERA algorithm named QoE-DEER, and then theoretically analyzes its algorithmic convergence and performance. In Section VII, we conduct simulations to evaluate our approach. In Section VIII, we conclude our work and look ahead future directions.

II. Related Work

With the increasing momentum of IoT technology and applications, edge computing has gradually been a widely-focused research problem, especially the resource management problem in edge computing environment. Thus far, many existing works have already investigated the edge resource management problem, respectively in the two aspects of *computation offloading* and *edge resource allocation* (abbr., ERA). In terms of the computation offloading problem [9]–[11], [18], [19], it simply studies selecting which of IoT service requests are served at edge servers, regardless of internal resource partition within the edge server. But for the ERA problem [12]–[14], [16], [17], [20], [29], [30], it further determines how many edge resources are partitioned for each IoT service request that is served at a shared edge server. Comparatively speaking, the study on ERA problem takes a more fine-grained edge resource management strategy than the computation offloading problem. As listed in Table I, we categorize and compare some representative related works according to a set of criteria. Interested readers may refer to

TABLE I
COMPARISON OF DIFFERENT EDGE RESOURCE MANAGEMENT APPROACHES

Category	Reference	Optimization Mode	Optimality	Optimization Objective
Computation Offloading	[9]	centralized	optimal	service latency minimization
	[18]	distributed	near-optimal	cost efficiency
	[19]	distributed	near-optimal	cost efficiency + user-incentive maximization
	[10]	centralized	optimal	QoE maximization (*QoE function by specific scenarios**)
	[11]	centralized	optimal	QoE maximization (*Mean Opinion Score*)
Edge Resource Allocation	[17]	centralized	optimal	cost efficiency
	[12], [13]	centralized	near-optimal	energy efficiency
	[14]	centralized	optimal	revenue maximization
	[20]	distributed	optimal	revenue maximization + economical efficiency
	[15], [16]	centralized	near-optimal	QoE maximization (*logistic function*)
	[29]	distributed	near-optimal	QoE maximization (*logistic function*)
	[30]	distributed	near-optimal	QoE maximization (*no restrictions on QoE function types*)
	This work	distributed	optimal	QoE maximization (*no restrictions on QoE function types*)

* Remark within parentheses indicates the adopted QoE function type or QoE modeling method.

this comparative table and learn more about the corresponding references in details.

1) *Computation Offloading Problem:* From the perspective of edge infrastructure providers (e.g., T-Mobile, and AT&T), it basically focuses on the design of computation task scheduling or service caching approach, aimed to improve the service efficiency of edge infrastructure providers. Here, the internal resource allocation scheme within the edge server is seldom addressed. Ma *et al.* [9] proposed a cooperative service caching and workload scheduling algorithm for service latency minimization in mobile edge computing, which was designed on the basis of water filling and Gibbs sampling. Hong *et al.* [18] utilized the game-theoretical approach to present a distributed multi-hop computation offloading scheme in the IoT-edge-cloud environment, with the objective of minimizing the computation cost. He *et al.* [19] formulated the computation task scheduling process in edge computing as a potential game, where the corresponding approximation ratios for cost efficiency and user-incentive maximization were theoretically provided. Song *et al.* [10] put forward a QoE-driven edge service caching algorithm for the Internet of Vehicles based on deep reinforcement learning, where the corresponding QoE function was carefully defined according to the specific application scenario. He *et al.* [11] evaluated the QoE metric with the Mean Opinion Score (MOS) and then developed a QoE-aware content caching method in IoT-edge networks, where the MOS was commonly applied as a typical measure of QoE metric.

2) *Edge Resource Allocation Problem:* It generally deals with a more refined resource partition scheme for each edge server, which suffers from much higher computational complexity than simply handling the computation offloading issue. From the standpoint of service providers who hire edge infrastructures to serve their own IoT users, the ERA problem determines the specific amount of edge resources allocated for each IoT service request, making the maximum IoT users served at edge and satisfied with a pre-defined optimization objective. Ma *et al.* [17] presented a cost-effective resource allocation framework for the cloud-assisted edge computing environment, which could dynamically optimize the computation capacity of edge nodes. Li and Huang [12] designed the dynamic resource provision algorithm for the edge computing paradigm, which achieved energy efficiency together with the heterogeneous performance requirement of IoT users guaranteed. Chen *et al.* [13] optimized the energy efficiency of edge resource allocation scheme through dynamically modulating the CPU-frequency scaling of edge servers. Huang *et al.* [14] proposed a joint task scheduling and resource allocation approach in mobile edge computing, which realized revenue maximization with the linear programming technique utilized. Chen *et al.* [20] put forward a decentralized edge resource allocation algorithm based on the leader-follower game, with the purpose of satisfying the users' and edge infrastructure providers' utility in different aspects. Lai *et al.* [15] formulated the correlation between QoS and QoE metrics using the logistic function, and then adapted the greedy-like approach to approximate the overall QoE maximization in the edge computing environment. Continuing with his previous works [15], [16], Lai *et al.* [29] further adopted the game-theoretical method to solve out the QoE-driven edge resource allocation problem in a distributed manner, where the approximation ratio of QoE maximization was theoretically proved. Ma *et al.* [30] characterized the joint cooperation and competition of QoE-aware edge resource allocation as a decentralized cyclic game which approximated the globally QoE-optimized state, where no restrictions on QoE function types were required.

To summarize, a distributed computation offloading or resource allocation scheme generally demonstrates a better scalability in the large-scale edge computing environment, where the process of computation offloading or resource allocation can be more efficiently executed in a parallel fashion. Nonetheless, according to the above literature review, very few distributed computation offloading or resource allocation methods in edge computing present the optimality guarantee. It follows that, the design of a distributed ERA approach

TABLE II
SUMMARY OF KEY NOTIONS

Symbol	Definition
$\mathcal{U}$	Set of M IoT devices/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 IoT 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 computational 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 computational 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 $\mathbf{s}_i$

having the optimality guarantee meets with several research challenges. In addition to this, the existing QoE-aware methods mainly evaluate the QoE metric based on a pre-defined QoE function (e.g., MOS-based, logistic, exponential, logarithmic, and scenario-tailored functions). Comparatively speaking, a QoE-aware approach without restrictions on QoE function types not only manifests greater applicability in general cases, but also better supports each IoT user's personalization of its own QoE function.

In this article, our QoE-aware decentralized ERA scheme differs from the existing literature. Unlike the references [18], [19], [29], [30], we put forward a distributed resource allocation approach in edge computing, which provides the QoE-optimality guarantee rather than an approximate optimization solution. Besides, our proposed QoE-aware method is with no restrictions on QoE function types, which is distinguished from the references [10], [11], [15], [16], [29]. In a nutshell, our QoE-aware decentralized ERA scheme demonstrates greater applicability, together with the heterogeneity of QoE functions across different IoT users addressed.

III. SYSTEM MODEL

We consider an edge computing system consisting of *N* IoT users and *M* edge servers. The finite set of *N* IoT users is denoted by $\mathcal{U} = \{u_1, \ldots, u_N\}$, while the finite set of *M* edge servers is represented by $\mathcal{E} = \{e_1, \ldots, e_M\}$. Multiple IoT users concurrently propose their service requests to nearby edge servers for processing. Edge servers are geographically placed based on the service popularity [8], with a limited signal coverage area covering a group of IoT users. In our considered scenario, each IoT 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 IoT user u_i proposes a single service request, and the user who proposes multiple service requests can be regarded as a group of IoT users. The number of CPU cycles required for processing user u_i's service request is h_i. Meanwhile, if the IoT user u_i chooses to process its service request at the edge server, 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 the IoT user u_i, and we also define $\mathcal{U}_j$ as the set of IoT users covered by the edge server e_j.

Resource: We assume that each IoT device has limited resources with the local CPU frequency of f_i^L. Besides, each edge server e_j is equipped with c_j units of computational resources. More complicated ERA scenario of multidimensional edge resources (e.g., RAM, disk storage, network bandwidth, etc.) will be explored in our future work. For simplicity, let f_j^E denote the CPU frequency per unit computational resource for the edge server e_j.

Network: Suppose that the IoT device accesses and interacts with edge servers via the wireless communication network whose bandwidth is *B*, then we apply the Shannon formula to estimate the data transmission rate between IoT device and edge server. The communication distance between IoT device 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, the data transmission rate from u_i to e_j can be calculated in (1), as adopted in [31] according to the Shannon formula.

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

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 of each IoT device u_i, and σ^2 is the noise power of wireless communication environment.

IV. PROBLEM STATEMENT

A. QoS Model

In order to present the correlation between QoS and QoE metrics, we firstly analyze the QoS gained by IoT users in edge computing. Service latency, which is considered as one of the most important QoS attributes in edge computing [32], [33], is adopted as the QoS metric in this article. Let $\boldsymbol{s}_i = (x_i, a_i)$ denote the ERA strategy for the IoT 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 the IoT user u_i decides to process its own service request locally, (x_i, a_i) is defined as (0, 0).

When the IoT user u_i conducts local computing with $x_i = 0$ and $a_i = 0$, then u_i will process its service request at the IoT device. Therefore, the service latency for IoT user u_i by local

computing is formulated by (2).

$$t_i^L = \frac{h_i}{f_i^L} \tag{2}$$

When the IoT 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 processing capacity for a_i units of computational resources at edge server e_{x_i} is indicated by $a_i \cdot f_{x_i}^E$. The processing capacity for a_i units of edge resources conforms to the additivity of unit processing capacity $f_{x_i}^E$, which can be implemented by Round Robin CPU scheduling.

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

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

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

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

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

B. QoE Model

1) Correlation Analysis Between QoS and QoE: As a user-centric indicator evaluating the degree of user satisfaction, QoE jointly depends on the subjective user preference and the objective performance of software services [44]. On the one hand, a higher QoS level meritedly makes for a higher QoE level. In the edge computing system, no IoT user will decline to experience a higher QoS level; each IoT user will gain greater satisfaction with a higher QoS level provided. On the other hand, each IoT user generally has the differentiated expectation on QoS ranging from low to high, which results in disparate QoS-QoE correlations. For some IoT users, their service requests with strong performance susceptibility usually have a high QoS demand, thus requiring to consume more computational resources to reach the satisfactory QoS level. Whereas, service requests from the other IoT users with moderate performance susceptibility have a medium QoS demand, in needless of many computational resources to make these users satisfied. Exemplified by the video analytic service [45], most of service requests merely needs the service latency below one second. For them, there is almost no perceptible difference between 100 and 1,000 milliseconds. However, the other service requests strictly expects of service latency within 100 milliseconds. In this case, a noticeable response delay could be recognized although service latency is slightly more than 100 milliseconds.

Pertinent literatures on QoE [46]–[49] have demonstrated the nonlinear correlation between QoS and QoE metrics. Commonly speaking, the QoE level gained by an IoT user is by no means proportional to its attained QoS level. After an IoT user's QoS earns a particular level, the corresponding QoE will exhibit very minor advancement regardless of a remarkable QoS increase, following the well-known Weber-Fechner Law [50]. As shown in Table III, several existing works adopted the QoE function in various forms, such as, MOS-based, logistic, sigmoid, exponential, logarithmic, and scenario-tailored functions, but they all exhibit the nonlinear correlation trend between QoS and QoE metrics. These QoE functions with different forms can be all applied into our proposed work, with no priori assumption on specific QoE function types. The concrete form of QoE function cannot affect the practicality of our proposed QoE-aware ERA approach.

TABLE III
SUMMARY OF QOE MODELING METHODS

QoE Function Type	Reference
MOS-based function	[11] [34]
QoE function by specific scenarios	[10] [35] [36]
logistic/sigmoid function	[15] [16] [29] [37] [38]
exponential function	[39] [40] [41]
logarithmic function	[21] [42] [43]

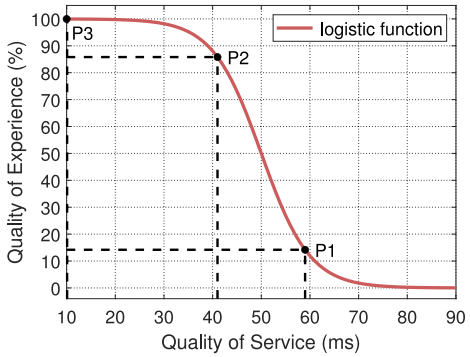

Fig. 1. Correlation between Quality of Experience and Quality of Service.

2) QoE Formulation: As previously clarified, the QoE is non-linearly correlated with the QoS. In our work, there is no priori restriction on QoE function types. But in order to expressly present the details of our proposed approach, we selectively choose the logistic function, which is a generalization of sigmoid function, as the QoE function adopted in our work. Thanks to its simplicity and generality, the logistic function (6)-(7) has been broadly acknowledged and applied in several existing works [15], [16], [29], [37], [38] to quantify the correlation between QoS and QoE metrics. As shown in Fig. 1, having a lower service latency (i.e., a quantitative representation of QoS) generally improves the QoE level. From the point P2 to P3, the QoE improvement gained tends to converge; the QoE keeps virtually unchanged in close proximity to the highest QoE level, with no regard to a noticeable increase in the QoS level. On the contrast, the QoE keeps steady increase from the point P1 to P2 with the cost of a little QoS progress. Such a QoE variation trend with QoS matches very well with our aforementioned QoS-QoE correlation analysis, conforming to the general opinion on QoS-QoE correlation trend [46]–[49].

Since the QoE metric is usually evaluated according to various levels, we apply the percentum to evaluate the QoE metric

in this article. Each IoT user can gain the maximum QoE of 100%. When the IoT 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).

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

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.

Likewise, when the IoT user u_i conducts edge computing based on $\boldsymbol{s}_i = (x_i, a_i)$, the QoE gained by user u_i can be formulated as (7).

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

C. Optimization Problem

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

When the IoT user u_i selects to conduct local computing, 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 the IoT user u_i selects to conduct edge computing with $\boldsymbol{s}_i = (x_i, a_i)$, 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 divided into two cases. Compared with local computing, if the IoT user u_i gains the higher QoE by edge computing through its ERA decision $\boldsymbol{s}_i = (x_i, a_i)$, 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 IoT 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 the user utility $\pi_i(\boldsymbol{s}_i)$ carefully defined, our QoE-aware ERA problem is thereby given, 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 IoT users with edge resources exceeding its resource capacity of c_j. And the user constraints (12) imply that each IoT user u_i can either be scheduled to one of its accessible edge servers, or process its own service request at the local IoT device.

Solving the QoE-aware ERA problem suffers from several computational challenges. It can be found that our QoE-aware ERA problem affiliates to the family of bin packing problem, which is $\mathcal{NP}$-hard to solve in a centralized manner [25]. Each edge server is conceived as a bin with finite computational resources, and our objective is to determine the ERA scheme across multiple IoT users with the overall QoE maximized. Besides, the optimization objective (10) formulated by QoE functions (see Eqs. (6), (7)) is in the nonlinear form, making the complexity of problem solving increased as well [51]. Hence, it necessitates an efficient and effective solution for our QoE-aware ERA problem.

V. Edge Resource Allocation Game

A. Game Formulation

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

Definition 1 (QoE-Incentive ERAGame): A strategic game $\mathcal{G}$ formulating the edge resource competition amongst *N* IoT 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* IoT users here. Players compete against each other to gain a higher QoE via occupying more edge resources.
- $\boldsymbol{S}_i$ is the set of feasible strategies for IoT user u_i. Its strategy $\mathbf{s}_i \in \boldsymbol{S}_i$ specifies the edge server x_i where the user u_i's service request is scheduled, together with the amount of edge resources (i.e., a_i) allocated for computational processing.
- π_i is the utility function of IoT user u_i, formulated by (8) or (9), to evaluate the QoE obtained by user u_i via adopting a strategy $\mathbf{s}_i \in \boldsymbol{S}_i$.

In our ERAGame, all the IoT users would like to be allocated more edge resources with the purpose of gaining higher QoE. However, each IoT 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 IoT user u_i makes up the *strategy profile* $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_N)$. During the game phase, suppose that an IoT user u_i originally selects a strategy $\mathbf{s}_i$, but finds another feasible strategy $\mathbf{s}_i'$ -> $\mathbf{s}_i' \in \mathbf{S}_i$ which gains higher utility (i.e., QoE). Incentivised by the benefit of this discovery, the IoT user u_i would naturally expect to update its strategy decision as $\mathbf{s}_i'$.

Resulted from the finite resource supply of edge servers, there should be conflicts amongst multiple IoT users, which

gives rise to the edge resource competition. To mitigate the conflicts amongst IoT users, the concept of Nash equilibrium (NE) [52] is thereby applied to manage the competitive behavior between IoT 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 $\mathbf{s}^*$ satisfying that for each player $u_i \in \mathcal{U}$,

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

Note that, $\mathbf{s}_{-i}$ denotes the strategy profile except the IoT user u_i, and 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 edge resource competition amongst IoT users. Any IoT user cannot arbitrarily increase its edge resource allocation, regardless of the current edge resource usage by other users. In addition, we define $\pi_{-i}(\mathbf{s}_i, \mathbf{s}_{-i})$ as the overall QoE gained by all IoT users except u_i under the strategy profile $\mathbf{s} = (\mathbf{s}_i, \mathbf{s}_{-i})$.

B. Edge Resource Preemption Rule

Among the non-unique NE solutions [52], we put forward an *edge resource preemption* rule on strategy decision to make the ERAGame admit at an NE solution with the optimal overall QoE. The preemption rule is designed with the principle that the strategy change requested by an IoT user u_i is adopted, which can bring the QoE upgrade to not only the IoT user u_i itself (i.e., π_i) but the overall system (i.e., $\sum_{u_i \in \mathcal{U}} \pi_i$) as well.

As depicted in Fig. 2, there exist two cases identified for classified discussion on strategy change. Specifically speaking.

- *Case* 1: *Strategy change of IoT user u_i is approved by occupying idle edge resources.*
  - In this case, there are idle edge resources which are free to be allocated. Thus, the IoT user u_i will naturally decide to occupy idle edge resources, thereby gaining the higher QoE level.
- *Case* 2: *Strategy change of IoT user u_i is approved by preempting edge resources from other users.*
  - Here, the additional edge resources that the IoT user u_i requests to realize a strategy change $\mathbf{s}_i' = (x_i', a_i')$ have been occupied by other users, because of which no more edge resources can be freely allocated to u_i. Therefore, an *edge resource preemption* rule is required to examine whether such a strategy change $\mathbf{s}_i'$ proposed by the IoT user u_i contributes to the overall QoE upgrade, at the cost of reducing the corresponding amount of edge resources from other users.

From now on, we start by introducing the details of our *edge resource preemption* rule. Without loss of generality, we assume that the IoT user u_i is the one who requests a strategy change $\mathbf{s}_i' = (x_i', a_i')$ in need of preempting the edge resources from other users. Let Δa_i denote the number of edge resource units that u_i expects to preempt; $p(u_i)$ represents the set of IoT users that could be preempted by u_i, since they take up the computational resources of edge server x_i' -> $e_{x_i'}$. The IoT user u_i gains the QoE improvement of $\Delta\pi_i$ through preempting Δa_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 QoE loss of $\Delta\pi_{-i}$. Based on these well-defined notations, the *edge resource preemption* rule is formally defined in Definition 3. Note that, after preemption, strategies of all the IoT users but $\{u_i\} \cap p(u_i)$ remain unchanged.

Definition 3 (Preemption Rule): The IoT user u_i can partially preempt the edge resources held by other users $u_k \in p(u_i)$, if the overall QoE is upgraded after preemption, i.e.,

$$\Delta\pi_i > \Delta\pi_{-i}. \tag{14}$$

The condition (14) triggering the preemption ensures that, the QoE improvement of user u_i (denoted by $\Delta\pi_i$) overcomes the simultaneous QoE decrease of users $u_k \in p(u_i)$ (denoted by $\Delta\pi_{-i}$). Besides, since strategies of all the IoT users but $\{u_i\} \cap p(u_i)$ are kept unchanged during the preemption, the QoE of IoT users $\mathcal{U} - \{u_i\} \cap p(u_i)$ 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 our intention of making the ERAGame converge to the NE solution with the optimal overall QoE.

C. Preemption-Based QoE Improve Algorithm (PRIM)

As demonstrated in Algorithm 1, we propose a Preemption-based QoE Improvement algorithm (PRIM), on account of the edge resource preemption rule introduced in Section V-B. Our PRIM algorithm is designed for each IoT user to determine whether a strategy change $\mathbf{s}_i'$ proposed by the IoT user u_i can benefit to the overall QoE improvement, where the edge resource preemption is enabled. If so, the PRIM algorithm will return the strategy profile $\mathbf{s}'$ ready for update, including the new strategy $\mathbf{s}_i'$ proposed by u_i and the correspondingly renewed strategy profile $\mathbf{s}_{-i}'$ of other users; otherwise, no strategy change will be made on the strategy profile.

With regard to our concrete design of PRIM algorithm, on the one hand, if there are sufficient idle edge resources for the IoT user u_i to update its strategy as $\mathbf{s}_i'$ gaining a higher QoE, then the strategy $\mathbf{s}_i'$ will be adopted. On the other hand, if no enough edge resources are available for the IoT user u_i to adopt the new strategy $\mathbf{s}_i'$, then the minimum QoE decrease $\Delta\pi_{-i}$ of users $p(u_i) = \{u_k \in \mathcal{U}{:}x_k = x_i' \textit{ and } k \neq i\}$ should be calculated and compared with $\Delta\pi_i$ to justify whether to conduct the edge resource preemption.

It is worth noting that, the minimum QoE decrease $\Delta\pi_{-i}$ for IoT users $p(u_i)$ can be calculated within finite iterations in an efficient computational complexity. In the first-round iteration, we sort up the IoT users $u_k \in p(u_i)$, who make edge resources insufficient for 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 computational complexity of $O(m \log m)$, where $m = |p(u_i)|$ is generally less than *N*. From the user rank *P* sorted up, we pick out the IoT user u_v who has the minimum QoE decrease in *P* and temporarily release one unit of edge resources from the IoT user u_v.

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

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

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

if $\pi_i(\mathbf{s}'_i) > \pi_i(\mathbf{s}_i)$ **then**
  if $a'_i + \sum_{u_k \in \mathcal{U}: x_k = x'_i, k \neq i} a_k \leq c_{x'_i}$ **then**
    return $\mathbf{s}' \leftarrow \{\mathbf{s}'_i, \mathbf{s}'_{-i}\}$;
  else
    $\Delta a_i \leftarrow a'_i - c_{x'_i} + \sum_{u_k \in \mathcal{U}: x_k = x'_i, k \neq i} a_k$;
    Order users $u_k \in \mathcal{U}: x_k = x'_i$ *and* $k \neq i$ by $\hat{\pi}_k$ in an ascending rank *P*;
    Initialize $\mathbf{s}'_{-i} \leftarrow \mathbf{s}_{-i}$;
    while $\Delta a_i > 0$ **do**
      Select the top-ranked user u_v in *P*;
      Try to update $\mathbf{s}'_v \leftarrow (x_v, a_v - 1)$;
      Reorder $u_k \in \mathcal{U}: x_k = x'_i$ *and* $k \neq i$ by $\hat{\pi}_k$, with the update value of $\hat{\pi}_v$;
      $\Delta a_i \leftarrow \Delta a_i - 1$;
    $\Delta \pi_i \leftarrow \pi_i(\mathbf{s}'_i, \mathbf{s}'_{-i}) - \pi_i(\mathbf{s}_i, \mathbf{s}_{-i})$;
    $\Delta \pi_{-i} \leftarrow \pi_{-i}(\mathbf{s}_i, \mathbf{s}_{-i}) - \pi_{-i}(\mathbf{s}'_i, \mathbf{s}'_{-i})$;
    if $\Delta \pi_i > \Delta \pi_{-i}$ **then**
      return $\mathbf{s}' \leftarrow \{\mathbf{s}'_i, \mathbf{s}'_{-i}\}$;
    else
      return $\mathbf{s}' \leftarrow \mathbf{s}$;
else
  return $\mathbf{s}' \leftarrow \mathbf{s}$;

Next, the second-round iteration starts up. Again, the IoT users $u_k \in p(u_i)$ are reordered by $\hat{\pi}_k$, before which $\hat{\pi}_v$ for the user u_v needs to be recalculated. The reordering operation can be performed on the basis of the user rank *P* previously sorted up in the last-round iteration, only needing to locate the insert position of user u_v in *P*. Such the reordering process can be implemented with the computational complexity of $O(m)$. Having all the IoT users reordered, we take the similar previous approach to pick out one of IoT users with one unit of edge resources temporarily released. Such the iterative process repeats until Δa_i units of edge resources temporarily released from IoT users $p(u_i)$. While the iterative process is terminated, the minimum QoE decrease $\Delta \pi_{-i}$ for IoT users $p(u_i)$ is obtained to testify whether the preemption condition (14) is satisfied. If satisfied, then Δa_i units of edge resources temporarily released ahead will be confirmed to be preempted by the IoT user u_i. Since the sort-up operation contributes the majority of computational complexity, thus our PRIM algorithm has the computational complexity of $O(m(\log m + \Delta a_i - 1))$.

Remark: The PRIM algorithm will work as a called function in our QoE-aware decentralized ERA algorithm (i.e., *QoE-DEER*, detailed later in Section VI). In the decentralized QoE-DEER algorithm, each IoT user makes local decision on the request of strategy change by invoking the PRIM algorithm.

VI. Decentralized Algorithm

A. 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 the optimal overall QoE. The NE solution with the overall QoE-optimal state specifies the ERA scheme across multiple IoT users. To make the ERA process conducted in a decentralized manner, a *cooperative messaging* mechanism is firstly designed, based 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 IoT users and edge servers.

- *Begin Message (BM):* The ERAGame does not reach an NE solution, as long as there exists at least one strategy requested for update. Here, each edge server e_j will broadcast the BM to its affiliated IoT users $u_i \in \mathcal{U}_j$ to continue the ERAGame.
- *Information Message (IM):* While broadcasting the BM, each edge server e_j also informs its affiliated IoT users $u_i \in \mathcal{U}_j$ of the ERA status about other users $u \in \cup_{j \in \mathcal{E}_i} U_j$, via sending the IM message.
- *Strategy Message (SM):* With the IM received, each IoT 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 ERAGame design, only an IoT user is permitted for strategy change in each decision iteration. Thus, all edge servers will negotiate and then send only one AM to an IoT user who is granted permission for strategy change. The negotiation is conducted based on the all-come-then-improve policy.
- *Update Message (UM):* Once after determining which one of IoT users is permitted for strategy change, all edge servers should also notify the other IoT users whose allocated resources are partially preempted, via sending the UM message.

Based on these messages, the decentralized QoE-DEER algorithm is given, as shown in Algorithm 2. Note that, Δt is predefined to denote the required time to transmit a message between IoT users and edge servers. To clarify the message protocol running in the QoE-DEER algorithm, the flow graph for 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 2-3): If the ERAGame has not reached an NE solution, all edge servers will respectively broadcast the BM to their own affiliated IoT users, to continue the game. Meanwhile, each edge server updates and packs the current ERA status as an IM message. The IM is sent to the affiliated IoT users of each edge server, providing users with reference information to determine whether requesting a strategy change.

Phase 2 (Lines 4-10): With sufficient IMs received, each IoT user u_i will invoke the PRIM algorithm to locally determine whether there is a new strategy profile $\mathbf{s}' = (\mathbf{s}'_i, \mathbf{s}'_{-i})$ both increasing *its own* QoE and benefiting to the *overall*

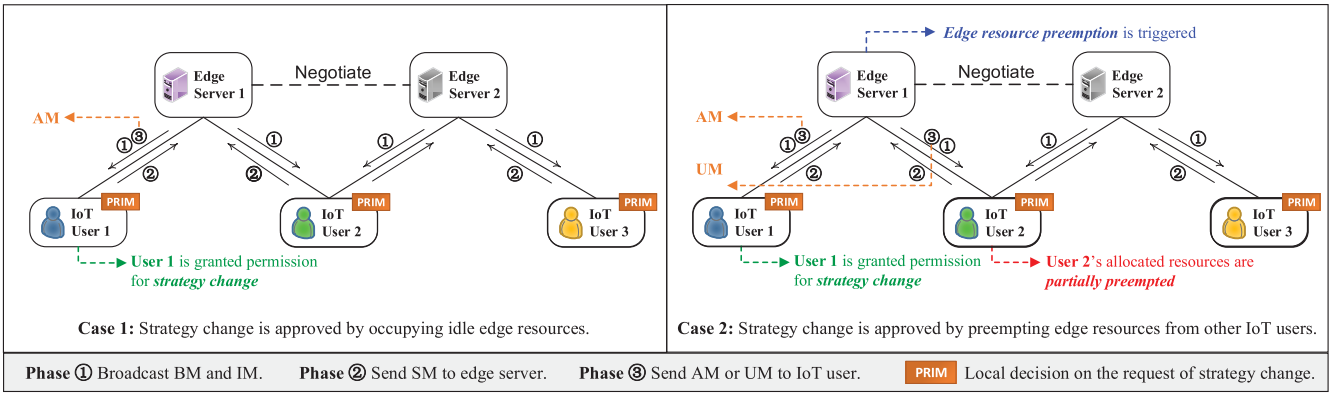

Fig. 2. Flow graph for the cooperative message passing in each time-slot decision.

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* $\hat{\mathbf{s}}_i \in \mathbf{S}_i$ **do**
8 $\mathbf{s}' \leftarrow PRIM(\hat{\mathbf{s}}_i, \mathbf{s}')$;
9 **if** $\mathbf{s}' \neq \mathbf{s}(t)$ **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* Δt **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** $\mathbf{s}(t+1)$; ▷ Reach the Nash Equilibrium.

QoE improvement. If such an eligible strategy profile $\mathbf{s}'$ non-uniquely exists, the IoT user u_i will finalize the new strategy profile $\mathbf{s}'$ requested for strategy change, according to the best-response guideline [52]. Finally, the IoT user u_i will send SM to the edge server specified by $\mathbf{s}'_i$, requesting permission for updating the strategy profile as $\mathbf{s}'$.

Phase 3 (Lines 11-18): Since edge servers won't receive SMs later than Δt if any IoT user requests for strategy change, thus edge servers should receive all SMs within Δt and then negotiate each other to decide which of strategy-change requests is adopted. The negotiation process is based on the all-come-then-improve policy; in specific, the strategy-change request $\mathbf{s}^*$ gaining the greatest overall-QoE improvement is selected out amongst all requests. Then, $\mathbf{s}^*$ is adopted. In response to the SM received by edge server, an AM will be sent back to the IoT user requesting the strategy change of $\mathbf{s}^*$; meanwhile, UMs should be sent to the IoT users whose part of allocated edge resources are preempted. On the contrast, if none of SMs is received by edge servers within Δt, the ERAGame has reached an NE solution, thereby the QoE-DEER algorithm terminates.

Our decentralized QoE-DEER algorithm for each time-slot decision can be approximately accomplished within the maximum time of $3\Delta t$, based on the message complexity analysis for distributed algorithms [53], [54]. In *Phase 1*, each edge server can broadcast BMs and send IMs to its affiliated IoT users in parallel, which takes at most Δt. Then, in *Phase 2*, if requesting for strategy change, the IoT user will send the SM to edge server; the SM won't be received by edge server later than Δt. 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 relevant IoT users that their allocated resources are partially preempted; the SM and UMs should be delivered at the respective target user within Δt. To sum up the above three phases, our decentralized QoE-DEER algorithm is estimated to be accomplished at the time cost of $3\Delta t$.

B. Algorithm Analysis

1) Convergence Analysis: We investigate whether our QoE-DEER algorithm can converge at an NE solution within the finite number of decision iterations. Since our QoE-DEER algorithm is the decentralized implementation of ERAGame, 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 [26]. Therefore, the convergence of QoE-DEER algorithm is ensured if we prove the ERAGame as a potential game. The definition of potential game is preliminarily given as follows.

Definition 4 (Potential Game): A game is a potential game, if there exists a potential function Φ:$\mathbf{S} \rightarrow \mathbb{R}$ such that for

each player $u_i \in \mathcal{U}$, $\Phi(\mathbf{s}') > \Phi(\mathbf{s})$ holds for any strategy improvement from $\mathbf{s}$ to $\mathbf{s}'$ satisfying $\pi_i(\mathbf{s}'_i) > \pi_i(\mathbf{s}_i)$, where $\mathbf{s} = (\mathbf{s}_i, \mathbf{s}_{-i})$ and $\mathbf{s}' = (\mathbf{s}'_i, \mathbf{s}'_{-i})$.

Note that, since edge resource preemption is enabled to reach an NE solution with the optimal overall QoE in our game formulation, thus the strategy $\mathbf{s}_i$ of more than one IoT users could be simultaneously changed during a 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 IoT 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 optimum of potential function [55]. The potential function monotonically increases with each strategy improvement until reaching the optimum of 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 [18], [19], [30]. We constructively prove the ERAGame as a potential game in Theorem 1, and 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 $\Phi(\mathbf{s})$.

Proof: Suppose that an IoT user $u_i \in \mathcal{U}$ 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 edge resource preemption rule, the IoT user u_i may need to conduct resource preemption to change its strategy from $\mathbf{s}_i$ to $\mathbf{s}'_i$. In the meantime, edge resources occupied by other IoT users $\{u_k \in \mathcal{U}{:}x_k = x'_i \text{ and } k \neq i\}$ could be partially preempted. Here, we apply $p(u_i)$ to denote the set of IoT users preempted by u_i. Assume that an IoT 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 $\pi_i(\mathbf{s}_v) - \pi_i(\mathbf{s}'_v)$. Then, the total QoE decrease for all IoT users $u_v \in p(u_i)$ should be $\sum_{u_v \in p(u_i)} (\pi_i(\mathbf{s}_v) - \pi_i(\mathbf{s}'_v))$.

Given the preemption condition (14), edge resource preemption is triggered only when

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

which thereby follows that

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

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-DEER algorithm can coverage at an NE solution within the finite number of decision iterations.

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 QoE-aware ERA problem.

As discussed in Section IV-C, our QoE-aware ERA problem targets the overall QoE maximization across multiple IoT users. Considering the computational intractability of our ERA problem, we design the QoE-DEER algorithm based on ERAGame which solves our ERA problem in a decentralized manner. Nevertheless, there might be non-unique NEs in the ERAGame [52]. Thus, it is worthy investigating whether the QoE-DEER algorithm eventually achieves the NE solution equivalently maximizing the overall QoE for our ERA problem. Theorem 2 has proved the equivalent optimality as follows.

Theorem 2: The QoE-DEER algorithm can find out the optimal edge resource allocation scheme, where the overall QoE is maximized.

Proof: Let $\mathbf{s} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ denote the strategy profile solved by the decentralized QoE-DEER algorithm, and $\mathbf{s}^* = \{\mathbf{s}^*_1, \ldots, \mathbf{s}^*_n\}$ to denote the optimal strategy profile which maximizes the overall 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 overall 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 a group of IoT users who gain a higher QoE in $\mathbf{s}^*_i$ than in $\mathbf{s}_i$. Thus, all the IoT users can be divided into two groups $\mathcal{G}_1$ and $\mathcal{G}_2$.

The first group $\mathcal{G}_1$ comprises the IoT 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 IoT users. For these users, in the worst case, their QoE may be degraded from $\mathbf{s}$ to $\mathbf{s}^*$, i.e., $\sum_{u_i \in \mathcal{G}_2} \pi_i(\mathbf{s}^*_i) \leq \sum_{u_i \in \mathcal{G}_2} \pi_i(\mathbf{s}_i)$. Then, the QoE decrement is correspondingly represented by $\Delta D = \sum_{u_i \in \mathcal{G}_2} (\pi_i(\mathbf{s}_i) - \pi_i(\mathbf{s}^*_i))$.

Due to 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 IoT users in $\mathcal{G}_1$ should be greater than the QoE decrement brought by users in $\mathcal{G}_2$, i.e., $\Delta I > \Delta D$, thus triggering the preemption condition (14) further upgrading the overall QoE. Note that, the decentralized QoE-DEER algorithm would not quit the loop until no IoT user can update its strategy to improve the overall QoE, even enabled by edge resource preemption. In other words, $\mathbf{s}$ cannot be the finalized strategy profile obtained by the QoE-DEER algorithm, which implies a contradiction with the initial assumption. Therefore, the decentralized QoE-DEER algorithm can equivalently maximize the overall QoE as the centralized one for our ERA problem. ∎

VII. Evaluation

A. Experimental Setup

We simulate an edge computing environment with multiple IoT users and edge servers by using the *iFogSim* [28] toolkit, where the realistic EUA dataset [27] is adopted. As a dependable simulation framework for edge computing, the

iFogSim toolkit has been widely adopted for simulating edge resource management [56]–[58]. It models the real-world edge computing environment, where the impact of different edge resource management techniques can be studied in various metrics. Meanwhile, as for the EUA dataset, it records the geographical information of cellular base stations and IoT 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 geographical location of IoT users is derived through the IP lookup service http://ip-api.com/, which converts the IP address of IoT users into the corresponding geographical location. The IP address blocks allocated to Australia is provided by the Asia Pacific Network Information Centre (APNIC). Since edge servers are normally deployed near cellular base stations [59], [60], 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 IoT 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 13.

For the IoT 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 IoT 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 IoT user u_i's service request is 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 IoT service request is 5.

For the wireless communication between IoT devices and edge servers, we set the transmitting power P_i for each IoT device u_i as 0.5 W, and the channel bandwidth B as 10 MHz [61], [62]. The noise power of wireless communication environment τ^2 is set as -87 dBm [63], while the channel power per unit communication distance γ_0 is set as -50 dB [31], [64].

B. Performance Benchmarks

Our proposed QoE-DEER algorithm is compared against four representative benchmark approaches, including a *QoE-optimal* baseline approach, two state-of-the-art approaches (i.e., *QoEUA*, and *QoS-aware*), and a *randomized* baseline approach.

- *QoE-Optimal:* This is an optimal ERA approach which achieves the overall QoE maximization by introducing the integer-programming technique. Here, our QoE-aware ERA problem (i.e., Eq. (10)) is settled in a centralized manner. Note that, this QoE-optimal method is implemented with the IBM ILOG CPLEX Optimization solver [65].
- *QoEUA:* This is a greedy-like heuristic ERA approach for overall QoE maximization, which synthesizes the state-of-the-art methods proposed in [15], [16]. Service requests of IoT users are conducted scheduling and resource allocation on edge servers, according to the non-increasing order of the number of their accessible edge servers [16]. Then, in a greedy-like manner [15], each scheduled user is then subsequently allocated to its accessible edge server with the most idle computational resources, which brings its greatest possible QoE.
- *QoS-aware:* This approach is developed by applying minor modification to our problem domain. Specifically, the optimization objective is displaced by the classical service-latency minimization. In other words, multiple IoT users participate in the ERAGame process, but with a distinctive objective of reducing their respective service latency. The corresponding NE solution can be reached and finalized as the QoS-aware ERA scheme.
- *Random:* This approach schedules and allocates the IoT service requests on edge servers, following a random fashion. According to a random rank, each IoT user u_i randomly selects the target edge server e_{x_i} and the corresponding edge resource allocation a_i. Suppose that all accessible edge resources have been occupied by other IoT users, then u_i would have to conduct local computing with no edge resources occupied.

Note that, all the experimental results about the QoE-DEER, *QoE-Optimal*, *QoEUA*, and *QoS-aware* algorithms are averaged over 30 runs, while the experimental results correlated to the *Random* algorithm are averaged over 3,600 runs. In each run, we randomly a specified number of IoT users and edge servers with different geographical locations from the EUA dataset.

C. NE Performance

To evaluate the effectiveness of our QoE-DEER algorithm, we investigate the performance of NE solution at which the QoE-DEER algorithm finally converges. Given the edge computing environment, the NE performance is evaluated from three aspects, which are the overall QoE, the average QoE per IoT user served at edge, and the number of IoT users served at edge. Our QoE-DEER algorithm is also compared against several performance benchmarks, as previously introduced in Section VII-B.

1) NE Performance v.s. Number of IoT Users N: We evaluate the NE performance through configuring the number of IoT users N from 170 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 Fig. 3(a). At various IoT user scales, our QoE-DEER algorithm always achieves the highest overall/average QoE over the state-of-the-art approaches. On the contrast, the *QoS-aware* approach generally neglects the diversity of user preferences on QoS, regardless of strengthening the perceived quality by IoT users (i.e., QoE). Since

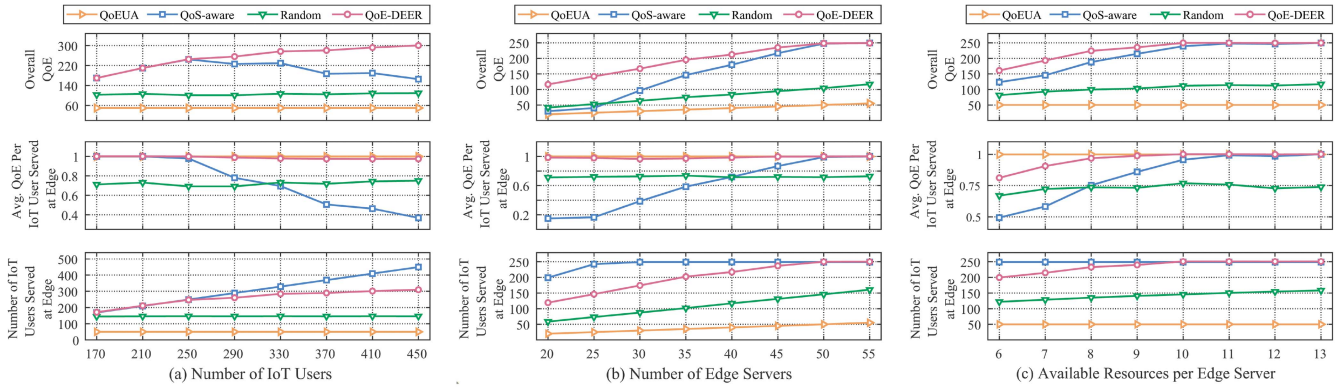

Fig. 3. NE performance of QoE-DEER algorithm under different system scales.

edge servers typically provide a lot higher computational performance than IoT devices, the *QoS-aware* approach would conduct service-latency reduction by allocating the maximum IoT users served at edge. Hence, the overall/average QoE acquired by *QoS-aware* approach gradually collapses with the increasing number of IoT users, where limited edge resource supply progressively becomes more constrained.

For the *QoEUA* and *random* approaches, they demonstrate the worst performance. In the *QoEUA* algorithm, the overall QoE and the number of IoT users served at edge remain almost unchanged although the number of IoT users scales up. This is because, each IoT user greedily takes up edge resources; once an IoT user is scheduled to an edge server, then the user will monopolize all available computational resources of the edge server. In Fig. 3(a), all available edge resources are occupied with no idle computational resources left beyond 50 IoT users. In terms of the overall QoE improvement, the *random* baseline performs even better than *QoEUA*, thanks to its randomized ERA strategy which is non-greedy. Comparatively speaking, the *random* approach better optimizes the utilization of limited edge resources than *QoEUA*, and thus realizes a higher overall user satisfaction (i.e., QoE).

2) NE Performance v.s. Number of Edge Servers M: We adjust the number of edge servers *M* from 20 to 55, with *N* and c_j respectively set as 250 and 10. Here, the experimental results on NE performance are shown in Fig. 3(b). Compared with the state-of-the-art approaches, our QoE-DEER algorithm gains the highest overall QoE, although they all present a growing trend in NE performance with the expanding scale of edge servers. More edge servers deployed into the edge computing environment implies a better capacity to serve user requests at edge, thus the number of IoT users served at edge increases. Thanks to more IoT users served at edge servers, the overall QoE gets improved accordingly.

Compared with the *QoS-aware* approach, our QoE-DEER algorithm demonstrates its advantage on overall/average QoE, especially when a fewer number of edge servers are deployed. When less adequate edge servers (i.e., $M \leq 45$ in Fig. 3(b)) are geographically deployed, the *QoS-aware* approach generally gains the overall/average QoE at a low level. In similar to Fig. 3(a), the *QoEUA* algorithm performs even worse than the *random* baseline in Fig. 3(b). The greedy-like *QoEUA* approach achieves the highest average QoE as our QoE-DEER algorithm, but incurs the cost of allocating the least number of IoT users served at edge. The utilization of limited edge resources is relatively low for the *QoEUA* algorithm, merely with a small proportion of IoT users served at edge. Hence, the overall QoE improvement is very limited in the *QoEUA* approach.

3) NE Performance v.s. Available Resources per Edge Server c_j: We investigate the impact of edge server's computational capacity c_j on NE performance, as depicted in Fig. 3(c). In specific, the number of available edge resources per edge server c_j is regulated from 6 to 13, while *M* and *N* is respectively set as 250 and 50. Under various settings on c_j, our QoE-DEER algorithm always outperforms the performance benchmarks on the overall QoE improvement. With reasonably sufficient computational resources equipped per edge server (i.e., $c_j \geq 8$ in Fig. 3(c)), our QoE-DEER algorithm can also acquire the highest average QoE per IoT user served at edge as the *QoEUA* approach. Meanwhile, when $c_j > 9$, the growth rate on NE performance for our QoE-DEER algorithm slows down. This is because, nearly all 250 users have been scheduled to be served at edge servers, with their QoE maximized at the highest level. Although more edge resources are supplied, their QoE cannot be virtually upgraded furthermore, merely resulting in the waste of idle edge resources.

The greedy-like *QoEUA* algorithm achieves the highest average QoE, but in an inefficient way. In Fig. 3(c), each IoT user would monopolize all of available resources on its dispatched edge server with no idle resources reserved for other IoT users, no matter how many available resources c_j is configured at each edge server. Hence, for the *QoEUA* algorithm, it is of little help to improve the overall QoE through configuring a greater c_j per edge server. In the meantime, the overall/average QoE acquired by the *QoS-aware* approach generally collapses when limited edge resource supply gets more constrained. In Fig. 3(c), the *QoS-aware* approach would acquire the overall/average QoE at a lower level than our QoE-DEER algorithm, when less sufficient computational resources are equipped at each edge server (i.e., $c_j \leq 10$).

4) QoE-DEER Algorithm v.s. QoE-Optimal Baseline: In order to evaluate the optimality of our proposed approach, we compare our QoE-DEER algorithm against the competitive

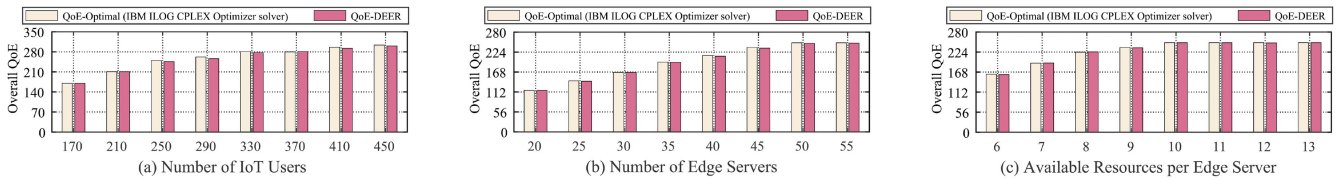

Fig. 4. NE performance of QoE-DEER algorithm against the *QoE-Optimal* baseline.

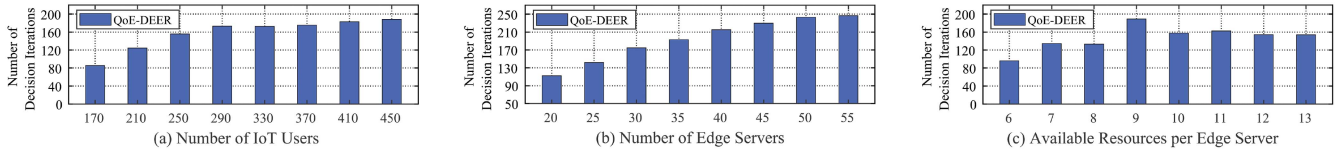

Fig. 5. Convergence of QoE-DEER algorithm under different system scales.

QoE-Optimal baseline approach. Recall that, the *QoE-Optimal* approach provides a centralized ERA solution for overall QoE maximization, based on the integer-programming technique; whereas, our QoE-DEER algorithm solves the ERA problem for overall QoE maximization in a decentralized manner. The comparative experimental result under different system scales is illustrated in Fig. 4. As the number of IoT users, edge servers, or available resources per edge server increases, both our QoE-DEER algorithm and the *QoE-Optimal* approach correspondingly gain a higher overall QoE. More importantly, the difference in the overall QoE gained by these two approaches is very minimal, within the excusable deviation range by IBM ILOG CPLEX Optimization solver. Given the above analysis, the optimality of our QoE-DEER algorithm is experimentally validated.

D. Algorithmic Convergence

We experimentally analyze the convergence of our QoE-DEER algorithm. The convergence of our QoE-DEER algorithm has been theoretically proved in Theorem 1. To measure the convergence of our QoE-DEER algorithm, we adopt the number of decision iterations required for reaching the finalized NE solution. Similar to Section VII-C, we evaluate the algorithmic convergence under diverse system scales, varied by different number of IoT users, edge servers, or available resources per edge server, as shown in Fig. 5. Each experimental result representing the number of decision iterations is averaged over 30 runs. In each run, a specified number of edge servers and IoT users are randomly selected from the EUA dataset, which have various geographical locations.

Fig. 5(a) exhibits the QoE-DEER convergence with the number of IoT users *N*. On the one hand, more IoT users participating in the ERAGame process generally result in more possible decisions to be made by IoT users, hence requiring a greater number of decision iterations to converge at the NE solution. On the other hand, more IoT users engaging into the ERAGame process also entails a fiercer competition amongst IoT users. Especially as the number of IoT users enters the range greater than 250, many IoT users are simply forced to conduct local computing due to the intensified shortage of edge resources. Thus, there is little need for extra decision iterations to find out the NE solution, when the number of IoT users further scales up. As in Fig. 5(a), no remarkable increase trend is exhibited in the number of required decision iterations, when the number of IoT users is greater than 250.

Fig. 5(b) describes the QoE-DEER convergence with the number of edge servers *M*. Since more edge servers participating in the ERA process implies the enlargement of strategy space $\mathbf{S}_i$ for each IoT user u_i, hence lowering down the algorithmic convergence rate. Thus, the number of decision iterations increases with the number of edge servers.

Fig. 5(c) shows the QoE-DEER convergence with the number of available resources per edge server c_j. In the initial range from $c_j = 6$ to 9, the number of decision iterations increases with the available resources c_j per edge server as well. Nevertheless, the number of decision iterations decreases after the certain point of $c_j = 9$. This is because there are relatively redundant edge resources after $c_j = 9$, and then more IoT users can be allocated the greater amount of edge resources without much competition.

VIII. Conclusion

In this article, we propose a game-theoretic approach which solves the Edge Resource Allocation (ERA) problem maximizing the overall QoE across multiple IoT users. To attack the computational intractability of our ERA problem, we formulate the QoE-aware ERA problem as a potential game named ERAGame, where each IoT user chooses its ERA decision based on its own QoE interests. Given the ERAGame, the ERA problem can be solved in a decentralized manner, where the edge resource preemption is leveraged to make the ERAGame reach the Nash Equilibrium which equivalently maximizes the overall QoE for the ERA problem. A cooperative message mechanism is designed to make our decentralized QoE-DEER approach applicable. 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 primary idea proposed in this article. It can more effectively handle various service requests arriving at different times, where a more generalized QoE optimization

objective considering the temporal dynamics could be then 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 the real-world IoT scenario should provide us with useful reference on user/system behaviors. Here, some other important QoE-related factors (e.g., security, and interoperability) can be further considered in our optimization design.

REFERENCES

[1] M. Raja, V. Ghaderi, and S. Sigg, "WiBot! In-vehicle behaviour and gesture recognition using wireless network edge," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (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 *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 1796–1807.

[3] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[4] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[5] A. Galanopoulos, T. Salonidis, and G. Iosifidis, "Cooperative edge computing of data analytics for the Internet of Things," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1166–1179, Dec. 2020.

[6] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 242–253, Jan. 2021.

[7] J. Huang, C. Zhang, and J. Zhang, "A multi-queue approach of energy efficient task scheduling for sensor hubs," *Chin. J. Electron.*, vol. 29, no. 2, pp. 242–247, 2020.

[8] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4702–4711, Oct. 2018.

[9] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2020, pp. 2076–2085.

[10] C. Song, W. Xu, T. Wu, S. Yu, P. Zeng, and N. Zhang, "QoE-driven edge caching in vehicle networks based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5286–5295, Jun. 2021.

[11] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric IoT," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 3, pp. 781–796, Jul.–Sep. 2020.

[12] S. Li and J. Huang, "Energy efficient resource management and task scheduling for IoT services in edge computing paradigm," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. (ISPA)*, 2017, pp. 846–851.

[13] 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 Trans. Cloud Comput.*, early access, Jun. 20, 2019, doi: 10.1109/TCC.2019.2923692.

[14] J. Huang, S. Li, and Y. Chen, "Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing," *Peer-to-Peer Netw. Appl.*, vol. 13, no. 5, pp. 1776–1787, 2020.

[15] P. Lai *et al.*, "Edge user allocation with dynamic quality of service," in *Proc. Int. Conf. Service Orient. Comput. (ICSOC)*, 2019, pp. 86–101.

[16] P. Lai *et al.*, "QoE-aware user allocation in edge computing systems with dynamic QoS," *Future Gener. Comput. Syst.*, vol. 112, pp. 684–694, Nov. 2020.

[17] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, Jul.–Sep. 2021.

[18] 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.

[19] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.

[20] 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 Gener. Comput. Syst.*, vol. 108, pp. 273–287, Jul. 2020.

[21] Y. Wang *et al.*, "A data-driven architecture for personalized QoE management in 5G wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 1, pp. 102–110, Feb. 2017.

[22] 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.

[23] X. Cao, J. Zhang, and H. V. Poor, "An optimal auction mechanism for mobile edge caching," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2018, pp. 388–399.

[24] A. Lachat, J.-C. Gicquel, and J. Fournier, "How perception of ultra-high definition is modified by viewing distance and screen size," in *Image Quality and System Performance XII*, vol. 9396, M.-C. Larabi and S. Triantaphillidou, Eds. Bellingham, WA, USA: Int. Soc. Opt. Photon., 2015, pp. 306–313. doi: 10.1117/12.2081140.

[25] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," in *Computers and Intractability*, vol. 340. New York, NY, USA: W.H. Freeman, 1979.

[26] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.

[27] P. Lai *et al.*, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Int. Conf. Service Orient. Comput. (ICSOC)*, 2018, pp. 230–245.

[28] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017.

[29] P. Lai *et al.*, "Quality of experience-aware user allocation in edge computing systems: A potential game," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2020, pp. 223–233.

[30] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for service-oriented resource allocation in edge computing," *IEEE Trans. Services Comput.*, vol. 13, no. 4, pp. 723–734, Jul./Aug. 2020.

[31] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, Jun. 2017.

[32] J. Huang, S. Li, Y. Chen, and J. Chen, "Performance modelling and analysis for IoT services," *Int. J. Web Grid Services*, vol. 14, no. 2, pp. 146–169, 2018.

[33] W. Miao, G. Min, X. Zhang, Z. Zhao, and J. Hu, "Performance modelling and quantitative analysis of vehicular edge computing with bursty task arrivals," *IEEE Trans. Mobile Comput.*, early access, Jun. 7, 2021, doi: 10.1109/TMC.2021.3087013.

[34] L. Zhou, "QoE-driven delay announcement for cloud mobile media," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 1, pp. 84–94, Jan. 2017.

[35] H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu, "Placing virtual machines to optimize cloud gaming experience," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 42–53, Jan.–Mar. 2015.

[36] M. Saleem, Y. Saleem, and M. Hayat, "Stochastic QoE-aware optimization of multisource multimedia content delivery for mobile cloud," *Clust. Comput.*, vol. 23, no. 2, pp. 1381–1396, 2020.

[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, Oct.–Dec. 2017.

[38] G. Zou *et al.*, "TD-EUA: Task-decomposable edge user allocation with QoE optimization," in *Proc. Int. Conf. Service Orient. Comput. (ICSOC)*, 2020, pp. 215–231.

[39] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014.

[40] N. Zhang, S. Zhang, J. Zheng, X. Fang, J. W. Mark, and X. Shen, "QoE driven decentralized spectrum sharing in 5G networks: Potential game approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 7797–7808, Sep. 2017.

[41] X. He, K. Wang, H. Huang, and B. Liu, "QoE-driven big data architecture for smart city," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 88–93, Feb. 2018.

[42] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks," *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1431–1445, Oct. 2013.

[43] R. Sun *et al.*, "QoE-driven transmission-aware cache placement and cooperative beamforming design in cloud-RANs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 636–650, Jan. 2020.

[44] R. K. P. Mok, R. K. C. Chang, and W. Li, "Detecting low-quality workers in QoE crowdtesting: A worker behavior-based approach," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 530–543, Mar. 2017.

[45] G. Ananthanarayanan *et al.*, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[46] M. Alreshoodi and J. Woods, "Survey on QoE/QoS correlation models for multimedia services," *Int. J. Distrib. Parallel Syst.*, vol. 4, no. 3, pp. 53–72, 2013.

[47] C. Liang, Y. He, F. R. Yu, and N. Zhao, "Enhancing video rate adaptation with mobile edge computing and caching in software-defined mobile networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 7013–7026, Oct. 2018.

[48] N. Banovic-Curguz and D. Ilisevis, "Mapping of QoS/QoE in 5G networks," in *Proc. Int. Convent. Inf. Commun. Technol. Electron. Microelectron. (MIPRO)*, 2019, pp. 404–408.

[49] Z. Hu, H. Yan, T. Yan, H. Geng, and G. Liu, "Evaluating QoE in VoIP networks with QoS mapping and machine learning algorithms," *Neurocomputing*, vol. 386, pp. 63–83, Apr. 2020.

[50] P. Reichl, S. Egger, R. Schatz, and A. D'Alconzo, "The logarithmic nature of QoE and the role of the Weber–Fechner law in QoE assessment," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2010, pp. 1–5.

[51] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 334–334, 1997.

[52] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.

[53] A. C.-C. Yao, "Some complexity questions related to distributive computing (preliminary report)," in *Proc. ACM Symp. Theory Comput. (STOC)*, 1979, pp. 209–213.

[54] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1432–1465, 2nd Quart., 2020.

[55] J. R. Marden, G. Arslan, and J. S. Shamma, "Cooperative control and potential games," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1393–1407, Dec. 2009.

[56] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, 2017, pp. 1222–1228.

[57] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, Mar./Apr. 2017.

[58] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-aware application module management for fog computing environments," *ACM Trans. Internet Technol.*, vol. 19, no. 1, pp. 1–9, 2019.

[59] Y. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, 2015.

[60] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1050–1060, Jul.–Sep. 2021.

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

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

[63] E. Tanghe *et al.*, "The industrial indoor channel: Large-scale and temporal fading at 900, 2400, and 5200 MHz," *IEEE Trans. Wireless Commun.*, vol. 7, no. 7, pp. 2740–2751, Jul. 2008.

[64] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," *Concurrency Comput. Pract. Exp.*, vol. 29, no. 16, 2016, Art. no. e3975, doi: 10.1002/cpe.3975.

[65] *IBM CPLEX Optimizer: High-Performance Mathematical Programming Solver for Linear Programming, Mixed-Integer Programming and Quadratic Programming*. Accessed: Aug. 29, 2021. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

Songyuan Li (Student Member, IEEE) received the B.Eng. and M.Eng. degrees in computer science and technology from the Beijing University of Posts and Telecommunications, China, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree in computer science with the College of Engineering, Mathematics, and Physical Sciences, University of Exeter, U.K. He has published articles in international journals and conference proceedings, including the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE ICWS, IEEE SCC, and IEEE ISPA. His current research interests include edge/cloud/services computing, performance modeling and optimization, and distributed machine learning.

Jiwei Huang (Member, IEEE) 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 a Professor and the Dean with the Department of Computer Science and Technology, China University of Petroleum, Beijing, China, and the Director with the Beijing Key Laboratory of Petroleum Data Mining. He has published one book and more than 50 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 ACM.

Jia Hu (Member, IEEE) received the B.Eng. and M.Eng. degrees in electronic engineering from the Huazhong University of Science and Technology, China, in 2006 and 2004, respectively, and the Ph.D. degree in computer science from the University of Bradford, U.K., in 2010. He is a Senior Lecturer with the Department of Computer Science, College of Engineering, Mathematics, and Physical Sciences, University of Exeter, U.K. His research interests include mobile and ubiquitous computing, resource optimization, applied machine learning, and network security.

Bo Cheng (Member, IEEE) received the Ph.D. degree in computer science and engineering from the University of Electronic Science and Technology of China in 2006. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. He has published more than 60 articles in international journals and conference proceedings, including the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE ICDCS, and IEEE ICWS. His current research interests include network services and intelligence, Internet of Things technology, communication software, and distributed computing. He is a member of ACM.