40th IEEE International Conference on Distributed Computing Systems July 8 - 10, 2020, Singapore. (c) IEEE 2020

# Quality of Experience-Aware User Allocation in Edge Computing Systems: A Potential Game

Phu Lai*, Qiang He*, Guangming Cui*, Feifei Chen†, Mohamed Abdelrazek†,
John Grundy‡, John Hosking§, and Yun Yang*
*Swinburne University of Technology, Australia. Email: {tlai, qhe, gcui, yyang}@swin.edu.au
†Deakin University, Australia. Email: {feifei.chen, mohamed.abdelrazek}@deakin.edu.au
‡Monash University, Australia. Email: john.grundy@monash.edu
§The University of Auckland, New Zealand. Email: j.hosking@auckland.ac.nz

***Abstract*—As many applications and services are moving towards a more human-centered design, app vendors are taking the quality of experience (QoE) increasingly seriously. End-to-end latency is a key factor that determines the QoE experienced by users, especially for latency-sensitive applications such as online gaming, health care, critical warning systems and so on. Recently, edge computing has emerged as a promising solution to the high latency problem. In an edge computing environment, edge servers are deployed at cellular base stations, offering processing power and low network latency to users within their geographic proximity. In this paper, we tackle the user allocation problem in edge computing from an app vendor's perspective, where the vendor needs to decide which edge servers to serve which users in a specific area. Also, the vendor must consider the various levels of quality of service (QoS) for its users. Each QoS level results in a different QoE level; thus, the app vendor needs to decide the QoS level for each user so that the overall user experience is maximized. To tackle the NP-hardness of this problem, we formulate it as a potential game then propose QoEGame, an effective and efficient game-theoretic approach that admits a Nash equilibrium as a solution to the user allocation problem. Being a distributed algorithm, QoEGame is able fully utilize the distributed nature of edge computing. Finally, we theoretically and empirically evaluate the performance of QoEGame, which is illustrated to be significantly better than the state of the art and other baseline approaches.**

***Index Terms*—edge computing, user allocation, quality of experience, quality of service, game theory**

## I. Introduction

In recent years, we are witnessing a rapid growth of mobile and IoT devices, including smartphones, wearables, environmental sensors, self-driving vehicles, etc. This comes with a rich variety and sophistication of applications and services, such as facial recognition, interactive VR/AR gaming, ultra-low latency streaming, and so on. They usually require intensive processing power and large energy capacity, which are not available on thin clients such as mobile or IoT devices. Traditionally, heavy computation tasks are offloaded to app vendors' servers in the cloud. Nevertheless, maintaining a low-latency connection to users is a major challenge for app vendors and service providers due to the skyrocketing number of connected devices, the increasing network traffic and computational load, plus the long distance between end-users and the cloud.

Network latency significantly impacts application performance, quality of service, and user experience. This is one of the main reasons why edge computing, sometimes referred to as fog computing [1], has emerged to tackle the challenge of high network latency. Mobile edge computing (MEC) [2] is one of the most popular edge computing paradigms, which takes advantage of the highly distributed cellular base station environment. In an MEC system, numerous edge servers, which provide both processing power and storage, are deployed at, or near base stations [3]. App vendors can deploy their apps on edge servers, which are in closer proximity to their users than the cloud, to remarkably reduce the latency of accessing those apps [4], [5].

In an MEC environment, edge servers are densely distributed. The coverage areas of adjacent edge servers[1] usually partially overlap to avoid non-service areas [6], [7] – the areas in which users cannot be served by any edge server. A user located in the overlapping area will be allocated to one of the edge servers covering them (*proximity constraint*) as long as that edge server has sufficient computing resources (*resource constraint*), e.g. CPU, RAM, storage, or bandwidth, to serve the user. Compared to a cloud server, a typical edge server comes with very limited computing resources due to its size limit [8], [9]. Thus, an ineffective user-to-edge-server allocation will exhaust edge server computing resources rather quickly, leaving no available computing resources to serve more users. In addition, this user allocation problem is getting more complicated since many applications and services support dynamic quality of service (QoS), or different levels of service performance, which can be presented by display resolution [10], frame rate and bitrate [11], data rate [12], network loss and jitter [13], etc. Naturally, a higher QoS level is achieved by a series of computation tasks with higher complexity, hence requires more computing resources. For example, high-definition graphics rendering or highly accurate data analysis would require more CPU, RAM, or bandwidth of an edge server. Greedily assigning high QoS levels to users

This research is partly funded by Australian Research Council Discovery Project grants (DP170101932 and DP180100212).

[1]In this paper, we speak interchangeably of an edge server's coverage area and a base station's coverage area.

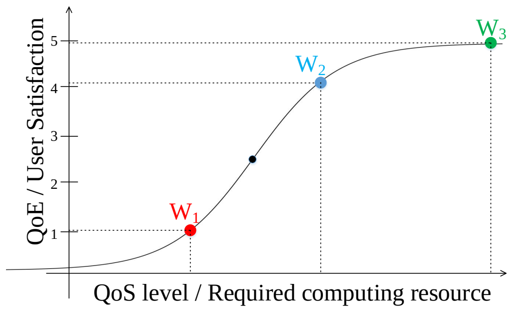


Fig. 1: QoS - QoE correlation

will also exhaust edge server computing resources quickly.

Unlike the computation offloading problem which challenges the edge infrastructure providers, e.g., AT&T or T-Mobile, this edge user allocation (EUA) problem challenges the app vendors who hire computing resources to serve their own users. It has been gaining a lot of attention [7], [14]–[16] in recent years. However, most of the existing works do not address the user quality of experience (QoE), which is a key criterion in assessing any app vendor's success. In this paper, we solve the EUA problem with the objective of maximizing the total QoE of all the users in a particular area. A general consensus is that a higher QoS level results in a higher QoE level. In fact, many works [13], [17], [18] have shown a quantitative correlation between QoS and QoE (Figure 1). This correlation can be leveraged by app vendors to better utilize the resources on edge servers. In general, a user's QoE increases by increasing its QoS level. However, the user's QoE tends to converge at some point, e.g., $W_3$ in Figure 1, and remains virtually unchanged at the highest level regardless of further increases in the QoS level. Taking advantage of this characteristic, an app vendor can maximize its users' satisfaction, measured by their total QoE, by jointly making two decisions – 1) a proper selection of a QoS level for each user, and 2) a proper user-to-edge-server allocation. In this paper, we study the quasi-static scenario where users are relatively stationary during the allocation process, not roaming across edge servers quickly [15], [19]–[22], e.g. surveillance cameras, traffic sensors, mobile or IoT users who are not moving at a high speed.

Solving this QoE-aware EUA problem effectively in an MEC system is challenging due its NP-hardness [16]. Furthermore, mobile users, which are a major stakeholder in MEC, could suffer a 30% performance penalty compared to non-mobile users with wired access [23], [24]. This calls for an efficient approach for finding solutions to the QoE-aware EUA problems. In this paper, we propose QoEGame, a game-theoretic approach for solving QoE-aware EUA problems. The main contributions of this paper include:

- We formulate this problem as a potential game [25] that aims to maximize the overall user QoE. The game is then theoretically analyzed and proven to admit a Nash equilibrium.

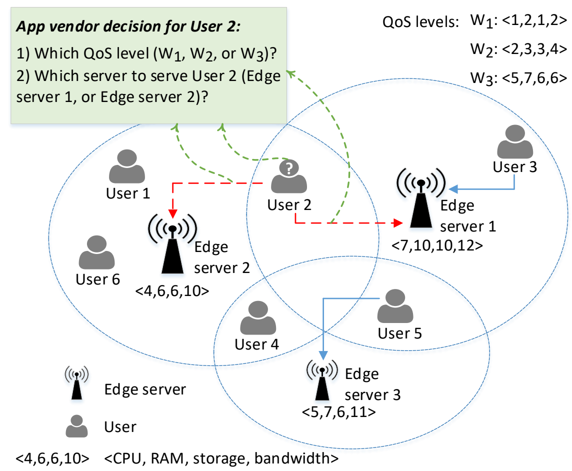


Fig. 2: Example MEC scenario

- Due to the $\mathcal{NP}$-hardness of the problem, finding an optimal solution to this problem is intractable, especially when a small geographic area could involve a great number of users. To effectively deal with its high complexity, we propose QoEGame, a distributed iterative algorithm for finding a Nash equilibrium. This algorithm simulates each user as a player in the game, whose decision, along with the decisions of other players, will benefit towards the final objective – maximizing the total QoE of all users.
- Extensive evaluations based on a real-world dataset are carried out to demonstrate the effectiveness and efficiency of QoEGame. The results show that QoEGame significantly outperforms the state of the art and other baseline approaches.

The remainder of the paper is organized as follows. We provide a motivating example in Section II. Section III formulates the QoE-aware EUA problem, which is then modeled as a potential game in Section IV. In Section V, we propose QoEGame, a game-theoretic approach for solving this problem. It is experimentally evaluated in Section VI. Section VII summarizes the key related work. Finally, we conclude the paper and discuss future work in Section VIII.

## II. Motivating Example

Let us consider a typical game streaming service. Game video frames are rendered on the game vendor's servers and then streamed to player devices. For the majority of players, there is no perceptible difference between 1080p and 1440p video resolution on their mobile devices, or even between 1080p and UHD from a distance farther than 1.5x the screen height regardless of the screen size [26]. Servicing a high-definition video quality like 1440p or UHD certainly consumes more resources (bandwidth, CPU, and GPU), which might be unnecessary since most players on their mobile devices are likely to be satisfied with 1080p. Instead, those resources can be utilized to serve players who are currently unhappy with the service, e.g. those experiencing poor 240p or 360p graphic, or those not able to play at all due to all nearby servers being overloaded. Therefore, the QoS level of some users can be lowered, potentially without causing any noticeable QoE downgrade, in order to better service users experiencing low QoS levels. In this way, the users' overall satisfaction can be maximized.

In this context, this research aims to allocate app users to edge servers and select QoS levels for them so that their total QoE is maximized. Take Figure 2 for example, there are three possible QoS levels, namely $W_1, W_2,$ and $W_3$, which consumes $\langle 1,2,1,2\rangle$, $\langle 2,3,3,4\rangle$, and $\langle 5,7,6,6\rangle$ units of $\langle CPU, RAM, storage, bandwidth\rangle$, respectively. User 2 can be allocated to either edge server 1 or edge server 2, which has less computing resources than edge server 1. Allocating user 2 to edge server 2 with a high QoS level will exhaust the resources that could be used to serve users 1 and 6, or to upgrade their QoS levels. Allocating user 2 to edge server 1 would result in a higher total QoE.

## III. Problem Formulation

### A. System Model

*Edge servers:* An MEC system in a particular area consists of a set of $m$ edge servers denoted by $\mathcal{S} = \{s_1, ..., s_m\}$. Each edge server $s_j \in \mathcal{S}$, $j = 1, ..., m$, has a certain amount of computing resources $c_j = (c_j^d)$, a $|\mathcal{D}|$-dimensional vector, where $d \in \mathcal{D} = \{CPU, memory, storage, ...\}$. Each edge server covers a specific geographical area $cov(s_j)$, as shown in Figure 2.

*Edge users:* Let $\mathcal{U} = \{u_1, ..., u_n\}$ denote the set of $n$ user $u_i$, $i = 1, ..., n$ in the area. In the EUA problem, every user must be allocated to an edge server unless all the servers covering the user have exhausted their computing resources. A user will be directly connected to the app vendor's cloud server if none of those edge servers has sufficient computing resources, or if the user is not located within the coverage of any edge servers.

For an application or service, there are $q$ pre-defined QoS levels, denoted by $\mathcal{W} = \{W_1, ..., W_q\}$. Each QoS level $W_k, k = 1, ..., q$, requires a specific amount of computing resources $W_k = (W_k^d)$, a $|\mathcal{D}|$-dimensional vector, $d \in \mathcal{D}$. A higher QoS level requires a higher amount of resources.

### B. Allocation Decisions

In the QoE-aware EUA problem, an app vendor needs to jointly make two decisions for each user $u_i \in \mathcal{U}$:

**Definition 1.** *(*Server Allocation Decision*) Given the set of edge servers $\mathcal{S} = \{s_1, ..., s_m\}$, let $a_i \in \{0\} \cup \mathcal{S}$ denote the edge server which user $u_i$ is allocated to. $a_i = 0$ when $u_i$ is unallocated.*

**Definition 2.** *(*QoS Selection Decision*) Given the set of QoS levels $\mathcal{W} = \{W_1, ..., W_q\}$, user $u_i$ is assigned a QoS level $b_i \in \{0\} \cup \mathcal{W}$ once allocated to an edge server. $b_i = 0$ when $u_i$ is unallocated.*

**Definition 3.** *(*Allocation Decision Profile*) Each user $u_i \in \mathcal{U}$ is associated with a pair of decisions $p_i = (a_i, b_i)$ as defined above. An allocation decision profile is a set of allocation decisions, one for each user, denoted by $\boldsymbol{p} = (p_1, ..., p_n) = ((a_1, b_1), ..., (a_n, b_n))$.*

User $u_i$ can only be allocated to one of the neighbor edge servers $\mathcal{S}_{u_i}$, which are the edge servers that have user $u_i$ in their coverage areas (*proximity constraint*):

$$a_i \in \{0\} \cup \mathcal{S}_{u_i}, \text{ where } \mathcal{S}_{u_i} = \{s_j \in \mathcal{S} | u_i \in cov(s_j)\}, \forall u_i \in \mathcal{U} \quad (1)$$

and the accumulated resource demands of all users allocated to an edge server must not exceed the available computing resources of that edge server (*resource constraint*). Let $\mathcal{U}_{s_j}^{alct} = \{u_i \in \mathcal{U} | a_i = s_j\}$ denote the set of users allocated to edge server $s_j$, we have:

$$\sum_{u_i \in \mathcal{U}_{s_j}^{alct}} b_i \preceq c_j, \quad \forall s_j \in \mathcal{S} \quad (2)$$

We use $\mathcal{U}_{s_j} = \{u_i \in \mathcal{U} | u_i \in cov(s_j)\}$ to denote the set of users located within edge server $s_j$'s coverage.

### C. System Benefit (QoE) Model

In the QoE-aware EUA problem, an app vendor benefits from satisfying its users, or maximizing their users' QoE. In general, a higher QoS level results in a higher QoE level. As demonstrated in [13], [17], [18], QoS and QoE exhibit a nonlinear correlation. When the QoS reaches a particular level, a user's QoE shows a very trivial improvement regardless of a noticeable increase in the QoS. Take the model in Figure 1 for example, the QoE gained from the $W_2 - W_3$ upgrade is nearly 1. In the meantime, the QoE gained from the $W_1 - W_2$ upgrade is approximately 3 at the expense of a little extra resource. The logistic function (3) has been widely acknowledged and employed in a lot of works [27]–[29] to model the correlation between QoE and QoS due to its generality and simplicity, which increase the generalizability of this work.

$$E_{\mathbf{p}}(p_i) = \frac{L}{1 + e^{-\alpha(x_i - \beta)}} \quad (3)$$

where $E_{\mathbf{p}}(p_i)$ represents the QoE level of user $u_i$ given its QoS level $b_i$, $L > 0$ is the maximum value of QoE, $\beta > 0$ controls where the mid-point of the QoE function is on the x-axis (QoS level in Figure 1), $\alpha > 0$ controls the growth rate of the QoE level (how steep the change from the minimum to the maximum QoE level is). $x_i = (\sum_{d \in \mathcal{D}} b_i^d)/|\mathcal{D}|$, where $b_i^d$ is the normalized amount of type-$d$ resource required by user $u_i$, $d \in \mathcal{D}$. We let $E_{\mathbf{p}}(p_i) = 0$ if user $u_i$ is unallocated.

### D. Optimization Model

Given a set of users $\mathcal{U} = \{u_1, ..., u_n\}$, a set of edge servers $\mathcal{S} = \{s_1, ..., s_m\}$, and a set of QoS levels $\mathcal{W} = \{W_1, ..., W_q\}$, the QoE-aware EUA problem can be formulated as a constrained optimization problem as follows:

$$\begin{aligned} &\max \sum_{u_i \in \mathcal{U}} E_{\mathbf{p}}(p_i) \\ &\text{s.t. } (1), (2) \end{aligned} \quad (4)$$

This formulation maximizes the total QoE of all users while satisfying the proximity constraint (1) and resource constraint (2). The solution to this problem is an allocation decision profile $\mathbf{p}$. [16] proves that this problem is $\mathcal{NP}$-hard by reducing the Partition problem to a special case of the decision version of this QoE-aware EUA problem.

## IV. QoE-aware User Allocation Game

In this section, we introduce QoEGame, a game-theoretic approach for effectively and efficiently solving the QoE-aware EUA problem. Traditionally, game theory has been widely applied in numerous areas as a powerful method for analyzing the interactions of players pursuing their own individual interests. In this paper, the players, i.e., the app users in the EUA problem, make decisions that could benefit other users as well, without significantly sacrificing their own benefit. Furthermore, QoEGame allows app vendors to efficiently solve the QoE-aware EUA problem in a distributed fashion by making allocation decisions for each user individually on each edge server, effectively leveraging the distributed characteristic of edge computing. App vendors do not have to suffer the high computational complexity of finding centralized optimal solutions. This is critical since users in an edge computing environment are usually highly latency-sensitive.

### A. Game Formulation

We formulate a QoE-aware EUA game that finds a decision profile which effectively selects QoS levels for users and allocates them to edge servers. The decision profile consists of two decisions for each user $u_i \in \mathcal{U}$, namely an edge server allocation decision $a_i$ and a QoS level selection decision $b_i$. Following the rules of the game, those decisions are determined so that the app vendor's objective is achieved. Let $\mathbf{p}_{-i} = (p_1, ..., p_{i-1}, p_{i+1}, ..., p_n)$ denote the allocation strategy of all users except user $u_i$. Note that in the EUA problem, a user makes decisions that benefit the whole system's goal, i.e. maximizing the total QoE of all users, instead of selfishly making decisions for its own benefit. In other words, the decision made by a user could allow other users to make "good" decisions accordingly. Based on other users' decisions $\mathbf{p}_{-i}$, a user $u_i$ can make a suitable decision $p_i$ so that the total QoE of all users is maximized (4).

Then, we model the above QoE-aware problem as a game $\Gamma = (\mathcal{U}, \{\mathcal{P}_i\}_{u_i \in \mathcal{U}}, \{E_i\}_{u_i \in \mathcal{U}})$, where the set of players is the set of users $\mathcal{U}$, $\mathcal{P}_i$ is the set of possible allocation strategies for users $u_i$, and $E_i$ is the benefit function which measures the benefit (total QoE) produced by user $u_i$'s decision $p_i \in \mathcal{P}_i$. In the game, users' allocation strategies might conflict. For example, in Figure 2, allocating users 2 and 4 to edge server 2 might exhaust its available computing resources, preventing users 1 and 6 from using the app with high QoS levels or even from being served by edge server 2. A better solution would be to allocate users 2 and 4 to edge servers 1 and 3, respectively, if they have sufficient resources, and users 1 and 6 to edge server 2. In this way, the total QoE of all users is maximized, every user is happy and does not desire to deviate from their existing allocation strategies. Next, we investigate whether this game admits at least one Nash equilibrium – a stable state of the game in which no player can make a decision that improves its own benefit if other players' strategies remain unchanged [30]. In the game, it is a stable state where no user can make a decision that improves the overall benefit of all neighbor users instead of its own benefit because our objective is to maximize the total benefit of all the users, as discussed above.

**Definition 4.** (Nash Equilibrium) *An allocation decision profile $\boldsymbol{p}^* = (p_1^*, ..., p_n^*)$ is a Nash equilibrium if no user can unilaterally update its decision to increase the system benefit:*

$$E_{\boldsymbol{p}^*_{-i}}(p_i^*) \geq E_{\boldsymbol{p}^*_{-i}}(p_i), \forall p_i \in \mathcal{P}_i, \forall u_i \in \mathcal{U} \tag{5}$$

**Lemma 1.** *Given a Nash equilibrium $\boldsymbol{p}^*$ of the game, the allocation decision $p_i^* \in \mathcal{P}_i$ made for each user $u_i \in \mathcal{U}$ is the best response to the decisions $\boldsymbol{p}_{-i}$ made by the other $n-1$ users.*

*Proof:* See Appendix A. ■

Lemma 1 guarantees that if a Nash equilibrium does indeed exist, QoEGame allows users to self-organize into a mutually agreed strategy. This eliminates the burden of high computational complexity of finding optimal solutions to large-scale QoE-aware EUA problems.

### B. Game Property

A critical property of a *potential game* is that it admits at least one Nash equilibrium [25]. In this section, we confirm the existence of a Nash equilibrium in the QoE-aware EUA game by proving that this is a potential game.

**Definition 5.** (Potential Game) *A game is a potential game if the following holds for a potential function $\phi(\boldsymbol{p})$:*

$$E_{\boldsymbol{p}_{-i}}(p_i) < E_{\boldsymbol{p}_{-i}}(p_i') \Rightarrow \phi_{\boldsymbol{p}_{-i}}(p_i) < \phi_{\boldsymbol{p}_{-i}}(p_i') \tag{6}$$

*for any $u_i \in \mathcal{U}$, $p_i, p_i' \in \mathcal{P}_i$ and $\boldsymbol{p}_{-i} \in \prod_{l \neq i} \mathcal{P}_l$.*

Based on Definition 5, we define a potential function:

$$\phi_{\mathbf{p}_{-i}}(p_i) = \frac{1}{2} \sum_{u_i \in \mathcal{U}} \sum_{u_j \neq u_i} \sum_{d \in \mathcal{D}} b_i^d \cdot \sum_{d \in \mathcal{D}} b_j^d \tag{7}$$

Now, we prove that the QoE-aware EUA game formulated in Section IV-A is a potential game with potential function $\phi_{\mathbf{p}_{-i}}(p_i)$ defined in (7).

**Theorem 1.** *The QoE-aware EUA game is a potential game with the potential function $\phi_{\boldsymbol{p}_{-i}}(p_i)$.*

*Proof:* See Appendix B. ■

## V. Distributed User Allocation Algorithm

In this section, we introduce QoEGame – an iterative and distributed user allocation algorithm for finding a Nash equilibrium in a potential game. QoEGame is inspired by *best response dynamics* [31], an evolutionary process that involves a finite number of iterations. In every iteration, each individual user develops the best allocation strategy in response to other users' strategies. It is important to note that the actual computation happens on edge servers, not on user devices. The process ends when no user desires to update their decisions, i.e. a Nash equilibrium. This is called the *Finite Improvement Property* of potential games.

### A. Algorithm Design

QoEGame (Algorithm 1) is a distributed and iterative mechanism that is able to find a Nash equilibrium of the game. Given a set of users $\mathcal{U}$, edge servers $\mathcal{S}$, and available QoS levels $\mathcal{W}$, QoEGame allocates users to edge servers with suitable QoS levels so that the total QoE of all users is maximized.

**Algorithm 1** QoEGame

1: **initialization:**
2: each user $u_i$ chooses an allocation decision $p_i = (a_i, b_i) = (0, 0)$, $\forall u_i \in \mathcal{U}$.
3: **end initialization**
4: **repeat**
5:     **for** each user $u_i \in \mathcal{U}$ **do**
6:         **if** $u_i$ is unallocated, $a_i = b_i = 0$ **then**
7:             find the decision $p'_i = (a'_i, b'_i)$ that benefits $u_i$ the most, i.e. highest QoE. $a'_i \in \mathcal{S}_{u_i}$, $b'_i \in \mathcal{W}$.
8:         **else** $\triangleright$ $u_i$ is allocated to an edge server $s_j$, $a_i \neq 0, b_i \neq 0$
9:             find the decision $p'_i = (a'_i, b'_i)$ that is the most beneficial for all involved users $\mathcal{U}_{s_j}$.
10:         **end if**
11:         **if** $p'_i > p_i$ **then**
12:             contend $p'_i$ for the decision update opportunity.
13:             **if** $u_i$ wins the decision update contention **then**
14:                 apply decision $p'_i$.
15:             **end if**
16:         **end if**
17:     **end for**
18: **until** no users need to update their decisions

Initially, no user is allocated and every user $u_i$ starts with an allocation decision $p_i = (a_i, b_i) = (0, 0)$, $\forall u_i \in \mathcal{U}$ ( Lines 1-3). After that, leveraging the Finite Improvement Property, the algorithm goes through an iterative process that allows every user to update their decisions iteration by iteration. The updated decision $p'_i$ must produce a higher total QoE compared to the previous decision $p_i$.

In each iteration, each user $u_i$ individually finds an optimal allocation decision $p'_i$ (Lines 6-10). If $p'_i$ leads to a higher QoE than the previous decision $p_i$, user $u_i$ will submit a request to contend for the opportunity to update $p_i$ to $p'_i$ (Lines 11-12). Once all the users have submitted their requests for decision update, the request with the greatest QoE improvement will be chosen as the sole winner in that iteration (Lines 13-14) and the allocation strategy will be updated accordingly (note that this strategy is not final and can be updated in future iterations). A request for decision update might involve one or more users. For example, user $u_i$ might want to lower its QoS level so that other users can utilize the released resources. If this request is selected as the winner, the allocation of all involved users will be updated accordingly. The requests for decision update that did not win will not be updated in the next iteration. All users affected by the latest decision update are required to update their decisions in the next iteration.

We now discuss the process for finding an optimal allocation decision for each user (Lines 6-10) in more detail. There are two possible cases based on a user's allocation status in the previous iteration. First, if $u_i$ has not been allocated, it will select an edge server that can serve it with the highest possible QoS level (greedy-like approach). Secondly, if $u_i$ has already been allocated to an edge server $s_j$, it will find a decision that is the most beneficial for all the involved users $\mathcal{U}_{s_j}$, i.e. users located within server $s_j$'s coverage area. User $u_i$ can freely move to another edge server and select another QoS level. The resources released by $u_i$'s decision or any available resources can then be utilized to serve more users or to increase the QoS levels of allocated users. QoEGame is a distributed algorithm since the process of finding an optimal allocation decision is executed for each individual user in parallel on edge servers.

**Convergence analysis.** The Finite Improvement Property of the potential game ensures that the allocation process will reach a Nash equilibrium after a finite number of iterations. Let $T$ be the total number of iterations, $Q_i \triangleq \sum_{d \in \mathcal{D}} b_i^d$, $Q_{min} \triangleq \min(Q_i)$, $Q_{max} \triangleq \max(Q_i)$, $i = 1, ..., n$, the following Theorem 2 holds.

**Theorem 2** (Upper Bound of Convergence Time)**.** *The maximum convergence time of QoEGame, measured by the number of decision iterations, is $n^2 Q_{max}^2 / (2(n-1)Q_{min})$.*

*Proof:* See Appendix C. ■

### B. Price of Anarchy in Total QoE

The design of QoEGame involves non-deterministic factors – if there are multiple users proposing allocation decisions with the same system benefit improvement, one of them will be randomly selected as the winner in that iteration. This leads to the fact that there might be more than one Nash equilibrium in the game. Thus, we evaluate the performance of QoEGame by analyzing the Price of Anarchy (PoA) in the total QoE, which indicates the ratio between the worst Nash equilibrium and the optimal allocation strategy [25]. Let $\chi$ denote the set of decision profiles that are able to reach different Nash equilibria in the game and $\mathbf{p}^* = (p_1^*, p_2^*, ..., p_n^*)$ denote the optimal decision profile. Given a decision profile $\mathbf{p} \in \chi$, let $poa_{QoE}(\mathbf{p})$ be the PoA measured by the ratio between the total QoE produced by $\mathbf{p}$ and $\mathbf{p}^*$, $poa_{QoE}(\mathbf{p})$ is calculated as follows:

$$poa_{QoE}(\mathrm{p}) = \frac{\min\limits_{\mathbf{p} \in \chi} \sum\limits_{u_i \in \mathcal{U}} E_{\mathbf{p}}(p_i)}{\sum_{u_i \in \mathcal{U}} E_{\mathbf{p}^*}(p_i)} \tag{8}$$

As discussed in Section III, there are two possibilities for the allocation of a user $u_i$: 1) $u_i$ can be allocated ($p_i \neq (0, 0)$) and 2) $u_i$ cannot be allocated ($p_i = (0, 0)$). The QoE benefit of unallocated users is zero. Thus, in the discussion in this section, we omit the QoE of unallocated users. According to (3), a higher QoS level leads to a higher QoE level. Let $QoE(\mathbf{p}) = E_{\mathbf{p}}(p_i) = L/(1+e^{-\alpha(\frac{Q_i}{|\mathcal{D}|}-\beta)})$. Then, $QoE_{max}(\mathbf{p}) = L/(1+e^{-\alpha(\frac{Q_{max}}{|\mathcal{D}|}-\beta)})$ and $QoE_{min}(\mathbf{p}) = L/(1+e^{-\alpha(\frac{Q_{min}}{|\mathcal{D}|}-\beta)})$.

Based on the above definitions, we have Theorem 3.

**Theorem 3.** *Given a decision profile* $\boldsymbol{p} \in \chi$ *that achieves a Nash equilibrium in the QoE-aware EUA game and the optimal decision profile* $\boldsymbol{p}^*$, *the PoA of the game* $poa_{QoE}(\boldsymbol{p})$ *measured by the ratio between the total QoE achieved by* $\boldsymbol{p}$ *and* $\boldsymbol{p}^*$, *satisfies:*

$$1 \geq poa_{QoE}(\boldsymbol{p}) \geq \frac{\sum_{u_i \in \mathcal{U}} QoE_{min}(\boldsymbol{p}) I_{\{p_i \neq (0,0)\}}}{\sum_{u_i \in \mathcal{U}} QoE_{max}(\boldsymbol{p}^*) I_{\{p_i \neq (0,0)\}}} \quad (9)$$

*where* $I_{\{condition\}}$ *is an indicator function which returns 1 if condition is true, and 0 otherwise.*

*Proof:* See Appendix D. ■

## VI. Empirical Evaluation

We have performed a series of experiments on a widely-used real-world dataset to evaluate the performance of QoEGame against existing approaches.

### A. Performance Benchmark

QoEGame is compared against five representative approaches, i.e. an optimal approach, three state-of-the-art approaches for solving the QoE-aware EUA problem, and a random baseline approach:

- *Optimal*: This is the optimal approach based on integer linear programming technique introduced in [16], which finds optimal solutions to QoE-aware EUA problems, i.e. the solutions with the highest total QoE. This approach is implemented with the IBM ILOG CPLEX Optimizer solver[2].
- *ICSOC19*: Proposed in [16], this greedy approach allocates each user to an edge server that has the most available computing resources. Each user is then assigned the highest possible QoS level given the computing resources available on the edge server serving it.
- *TPDS19* [15]: This approach solves the EUA problem with the objectives of maximizing the number of allocated users and minimizing the overall system cost calculated based on the costs of required computing resources on edge servers. Since TPDS19 does not consider dynamic QoS, users' QoS levels are randomly pre-specified.
- *ICSOC18* [7]: This approach models the EUA problem as a variable-sized vector bin packing problem and proposes an optimal approach that maximizes the number of allocated users while minimizing the number of edge servers required to serve the allocated users. Similar to TPDS19, this approach does not consider dynamic QoS either. Thus users are assigned the same QoS levels as TPDS19.

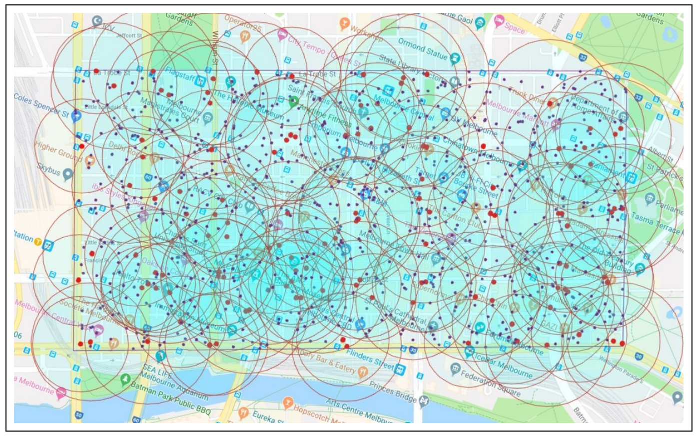

Fig. 3: Base stations in Melbourne CBD (1,800m × 1,000m)

- *Random*: This approach allocates each user to a random edge server as long as that edge server has sufficient computing resources to accommodate this user and has this user within its coverage area. The QoS level to be assigned to this user is randomly determined based on the edge server's remaining computing resources. For example, if the maximum QoS level that the edge server can offer the user is $W_2$, the user will be randomly assigned either $W_1$ or $W_2$.

All the experiments are conducted on a Windows computer equipped with Intel Core i5-7400T processor (4 CPUs, 2.4GHz) and 8GB RAM.

### B. Experimental Settings

The experiments are conducted on the EUA dataset[3] [7], which contains the geographical locations of end-users and all cellular base stations in Australia. This dataset was also used in [15], [16], and [7] to evaluate ICSOC19, TPDS19, and ICSOC18.

*Edge servers:* To capture the characteristics of a 5G environment [32], we simulate a 1.8 km$^2$ Melbourne CBD area (Figure 3) covered by 125 base stations, each equipped with an edge server. The coverage radius of each edge server is randomly generated within 100m - 150m. The computing resources available on the edge servers are randomly generated following a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where $\mu$ is the average capacity of each resource type in $\mathcal{D}$, and the standard deviation $\sigma = 10$ for all conducted experiments. Since a normal distribution might contain negative numbers, any negative amount of computing resources generated is rounded up to 1.

*Edge users:* We assume that for each user, there are three possible QoS levels $\mathcal{W} = \{<1,2,1,2>, <2,3,3,4>, <5,7,6,6>\}$, and $\mathcal{D} = \{CPU, RAM, storage, bandwidth\}$. We have conducted experiments with other settings and achieved similar results. Thus, we select those three QoS levels as representative in this section. Different values of the parameters in the QoE model (3) have also been tested. In this

[2]www.ibm.com/analytics/cplex-optimizer/

[3]www.github.com/swinedge/eua-dataset/

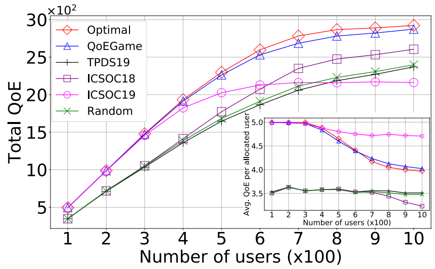


Fig. 4: Total QoE vs. number of users (Set #1).

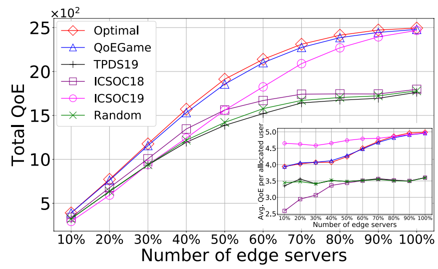


Fig. 5: Total QoE vs. number of edge servers (Set #2).

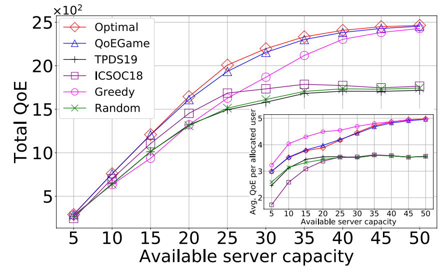


Fig. 6: Total QoE vs. edge server's available computing resources (Set #3).

TABLE I: Experimental Settings

| | Users | Edge servers | Available resources ($\mu$) |
|---|---|---|---|
| Set #1 | 100, ..., 1000 | 70% | 35 |
| Set #2 | 500 | 10%, ..., 100% | 35 |
| Set #3 | 500 | 70% | 5, 10, ..., 50 |

section, we employ $L = 5, \alpha = 1.5$, and $\beta = 2$, and present the corresponding results.

To comprehensively analyze the performance of QoEGame in various EUA scenarios, we conduct a series of experiments with different varying parameters, including the number of users, number of edge servers, and edge servers' available computing resources. Table I summarizes the settings of the experiments, which will be discussed in the next section. Each experiment is repeated 100 times to obtain 100 different user distributions and the results are then averaged. This allows extreme cases, such as overly dense or sparse user/server distributions, to be neutralized. To evaluate the performance of the approaches in achieving the optimization objective, which is to maximize the total QoE of all users as discussed in Section III, we compare the total QoE of all users achieved by the six approaches, the higher the better. To evaluate the approaches from another perspective, we also measure the number of users that are allocated to edge servers by each approach, the higher the better. The efficiency of QoEGame is also evaluated.

## C. Experimental Results

Figures 4, 5, and 6 demonstrate the effectiveness of all approaches in experiment Sets #1, #2, and #3 in terms of the total QoE of all users. Figures 7, 8, and 9 demonstrate their effectiveness in terms of the number of allocated users. In general, Optimal, being the optimal approach, clearly achieves the highest QoE compared to all other approaches across all experiments. This comes at the cost of its very high computational overhead (could go up to over 3 seconds as demonstrated in [16]) and is thus inapplicable in real-world 5G scenarios, where low latency is critical. QoEGame achieves a QoE performance very close to Optimal and clearly outperforms all other approaches. At the same time, QoEGame is able to allocate a good number of users to edge servers. We will also demonstrate the efficiency of QoEGame, measured by its convergence time, i.e. the number of iterations taken to reach a Nash equilibrium. This is a critical and machine-independent efficiency indicator for game-theoretic approaches [20], [21], [33], [34].

### 1) Effectiveness:

**Experiment Set #1**. In this set of experiments, the number of users is varied from 100 users to 1,000 users in steps of 100. The number of edge servers is fixed at 70% of all edge servers in the simulated area. Figure 4 shows the total QoE of all users in the experiments. Under all experimental settings, the difference in the total QoE achieved by Optimal and QoEGame is very marginal, which, with the theoretical analysis in Section V-B, confirms the near-optimality of QoEGame. From 100 to 400 users, ICSOC19 achieves a QoE almost as high as Optimal and QoEGame. This occurs under those settings because the available computing resources are redundant and therefore almost all users receive the highest QoS level. However, as the number of users continues to increase while the total amount of computing resources is fixed, the average amount of computing resources for each user becomes more scarce. As a result, the performance of ICSOC19 deteriorates quickly. Random and the other two state-of-the-art approaches, i.e. ICSOC18 and TPDS19, are outperformed by Optimal and QoEGame since they do not consider the scenario where the QoS level of a user can be dynamically adjusted. TPDS19 is even worse than Random since it focuses on minimizing system costs.

Figure 7 shows the percentage of allocated users. We can observe a decreasing trend here. Clearly, since the amount of computing resources is fixed, introducing more users will increase the number of users who cannot be allocated to any edge servers. ICSOC18, aiming to maximize the number of users served, is obviously able to allocate the most users as also demonstrated in Figures 8 and 9, closely followed by Optimal and QoEGame. Random and TPDS19 allocate fewer users than ICSOC18, Optimal, and QoEGame. ICSOC19 is by far the worst because it rapidly exhausts the available computing resources on edge servers.

Note that given the same amount of computing resources under all experimental settings, increasing the number of users will consequently decrease the average QoE of each allocated user as shown in the inset graph in Figure 4 since the same amount of computing resources is now to be shared among more users. ICSOC19 appears to be the best approach in

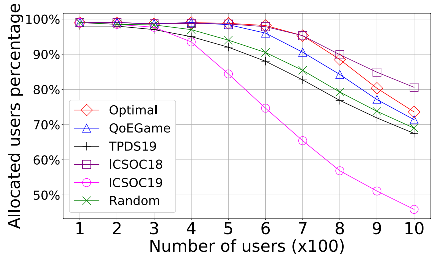


Fig. 7: Percentage of allocated users vs. number of users (Set #1).

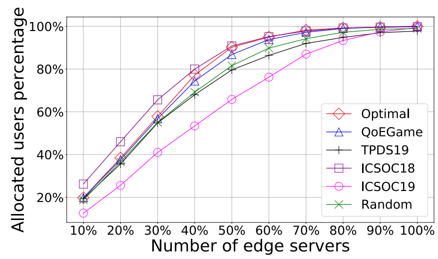


Fig. 8: Percentage of allocated users vs. number of edge servers (Set #2).

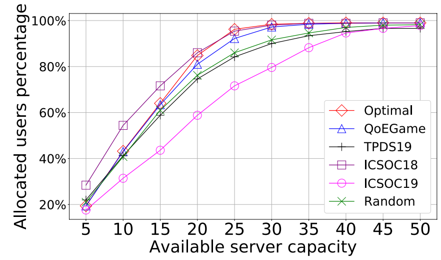


Fig. 9: Percentage of allocated users vs. edge server's available computing resources (Set #3).

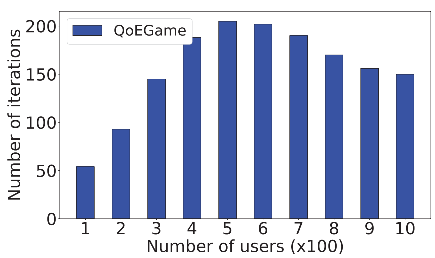


Fig. 10: Number of decision iterations vs. number of users (Set #1).

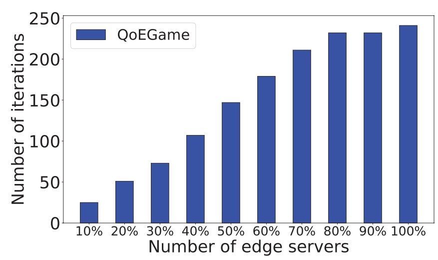


Fig. 11: Number of decision iterations vs. number of edge servers (Set #2).

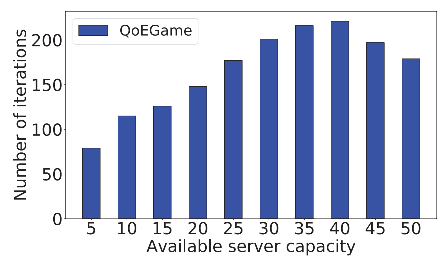


Fig. 12: Number of decision iterations vs. edge server's available capacity (Set #3).

this aspect since every user is greedily assigned the highest possible QoS level on an edge server. Nevertheless, this results in an extremely low number of allocated users. This can be observed in all other sets of experiments.

**Experiment Sets #2 & #3**. In these experiment sets, we vary the number of edge servers available to serve users, from 10% to 100% in steps of 10% (Figures 5 and 8), and the amount of available computing resources on edge servers, from 5 to 50 in steps of 5 (Figures 6 and 9). The total QoE depicted in Figures 5 and 6 exhibits a trend similar to experiment Set #1 (Figure 4), where QoEGame achieves a very near-optimal performance and outperforms all other approaches by considerable margins. As we increase the number of edge servers and the available computing resources, the total computing resources become more redundant, allowing more users to enjoy the highest QoS level. This is the reason why ICSOC19 also gradually approaches the performance of the Optimal. Additionally, as the computing resources become redundant, more users can be allocated to edge servers as illustrated in Figures 8 and 9. As can be seen in the figures, Optimal and QoEGame continue to outperform TPDS19 and Random. The percentage of users allocated by ICSOC19 is remarkably lower compared to all other approaches.

Optimal and QoEGame also outperform TPDS19, ICSOC18, and Random, in terms of the average QoE per allocated user (inset graphs in Figures 5 and 6). ICSOC19 achieves the best performance in this aspect due to the same reason discussed for experiment Set #1.

*2) Efficiency:*
It is intractable to find an optimal solution to the $\mathcal{NP}$-hard QoE-aware EUA problem with Optimal in large-scale scenarios. As a result, we introduce QoEGame, an iterative and distributed algorithm. In this section, we discuss the efficiency of QoEGame and how well it scales with different experiment parameters. Figures 10, 11, and 12 demonstrate the convergence time of QoEGame in experiment Sets #1, #2, and #3, respectively, measured by the average number of iterations required by QoEGame to reach a Nash equilibrium. In Figure 10, the number of iterations increases linearly with the increase in the number of users from 100 to 500 users. From 500 users onwards, the convergence time decreases at a slower pace than it increases. The rationale for these phenomena lies behind the competitiveness of the game. In the 100 - 500 user range, there are still sufficient computing resources to accommodate almost all the users (Figure 7). More users thus lead to more possible decisions to be made for each individual user, hence the increase in the convergence time. As the number of users enters the range of 600 - 1,000, most of the extra users are unallocated due to the scarcity of computing resources. However, because of the high density of users, who collectively contribute to finding the solution in parallel, QoEGame is able to find a Nash equilibrium quicker.

In experiment Set #2 (Figure 11), increasing the number of edge servers increases the possibility of decision updates, i.e. more decision updates, or more required iterations, hence the gradual increase in the convergence time of QoEGame. In experiment Set #3, as shown in Figure 12, increasing the available computing resources also increases the decision update possibility. However, after a certain point, 40 in this case, the convergence time decreases since there are now relatively redundant computing resources and more users can be served with high QoS levels without much competition.

## VII. Related Work

Edge computing, sometimes referred to as fog computing, was introduced by Cisco [1] in 2012 to overcome a key issue of cloud computing – high latency. Compared to the traditional cloud computing paradigm, edge computing possesses numerous unique properties, including wide-spread geographical distribution, a sizeable number of nodes, location awareness, the predominant role of wireless access, and a strong presence of streaming and real-time applications. Those properties enable edge computing to deliver a new generation of services and applications at the edge of the network, further extending the existing cloud computing paradigm [4].

QoE management and QoE-aware resource allocation have long been a challenge even before the cloud computing era [35]. Su et al. [36] develop a framework for resource allocation among media cloud, brokers and mobile social users that maximizes media cloud's profit and user's QoE. While the concept of brokers is similar to edge servers in our work, there are some essential differences. In their work, the broker is just a middleware for transferring tasks between the cloud and mobile users. Edge servers, on the other hand, are capable of processing computation tasks. He et al. [37] study the trade-off between QoE and system costs of virtual machine provisioning in a centralized video-streaming cloud environment. In the aforementioned works, QoE is measured by the processing, playback, or downloading rate. We consider a more general scenario where QoE is measured based on QoS levels, which are represented by the required amount of computing resources.

Relevant QoE-focused problems have started gaining attraction in the field of edge computing as well. Chen et al. [38] introduce an architecture that integrates resource-intensive computing with mobile applications while leveraging mobile cloud computing. They aim to provide a new line of personalized and QoE-aware applications. The authors of [39] and [40] solve the application placement problem in the edge computing environment. In their works, the QoE of a user is estimated based on three levels (low, medium, and high) of access rate, processing time, and required resources. The user allocation problem that we are dealing with can be regarded as the next step after the application placement phase. Hong et al. [41] address the QoE-aware computation offloading scheduling problem in mobile clouds from a networking perspective, where the energy consumption of mobile devices and latency must be considered in most cases. In contrast, in this paper, the user allocation problem is tackled from the app vendor's perspective, who tries to allocate its own users rather than dealing with low-level computation tasks.

Besides the aforementioned literature, there are a number of works on user allocation in edge computing [7], [14]–[16]. [14] considers the user mobility scenario where users can move from one place to another, which requires reallocating users among edge servers. We, on the other hand, study a quasi-static scenario. Furthermore, they do not consider the dynamic of user's QoS, thus they measure the capacity of an edge server by a fixed number of users that can be allocated to it. [7], [15] tackle the quasi-static scenario but lack the consideration of dynamically adjustable QoS levels and QoE. As a results, the approaches proposed in those works are not suitable for solving the QoE-aware EUA problem as demonstrated in Section VI-C. The heuristic approach proposed in [16] is the most relevant to our work. Nevertheless, it is very ineffective in resource-scare scenarios, which are very common in edge computing. QoEGame has been shown to be very effective in all scenarios and able to fully leverage the distributed characteristic of an edge computing system.

## VIII. Conclusions and Future Work

User quality of experience (QoE) is of great significance for any applications and services that are human-centric. However, there is very limited work in this area in edge computing. In this paper, we investigate the edge user allocation problem, in which an app vendor needs to allocate its own users to proper edge servers and at the same time, achieve its optimization objectives. We consider the scenario where the quality of service (QoS) level of a user can be dynamically adjusted depending on the current state of the system, e.g. the available computing resources on the edge servers. Each QoS level can be mapped to a QoE level, or how satisfied a user is with the service given a delivered QoS level. Our goal is to maximize the total QoE experienced by all the users. We formulate this problem as a potential game and introduce QoEGame – an iterative and distributed algorithm to find a Nash equilibrium in the game. The effectiveness and efficiency of QoEGame are theoretically and empirically demonstrated via a series of experiments conducted on a real-world dataset, against a number of baseline and state-of-the-art approaches.

Being a new problem and has not been studied extensively, there are many possible directions for future work; for example, QoE-aware user allocation in time-varying scenarios, user's mobility, QoE-aware service migration, web data caching, service recommendation at the edge, etc. Furthermore, a finer-grained QoE model that considers more domain-specific factors could also be developed.

## Appendix A
## Proof of Lemma 1

If the allocation decision $p_i^*$ made by user $u_i$ is not the best decision in $\mathcal{P}_i$, there must be another better decision $p_i \in \mathcal{P}_i$ that increases the system benefit (total QoE of all users), i.e. $E_{\mathbf{p}^*_{-i}}(p_i^*) < E_{\mathbf{p}^*_{-i}}(p_i)$. As a result, changing from $p_i^*$ to $p_i$ leads to greater system benefit. This is in contradictory to (5), where no user can unilaterally increase the overall benefit in a Nash equilibrium.

## Appendix B
## Proof of Theorem 1

Let us assume that a user $u_i$ has two allocation decisions $p_i$ and $p_i'$ such that $E_{\mathbf{p}_{-i}}(p_i) < E_{\mathbf{p}_{-i}}(p_i')$. According to (3),

$E_{\mathbf{p}_{-i}}(p_i) < E_{\mathbf{p}_{-i}}(p'_i)$ implies:

$$\frac{L}{1+e^{-\alpha(x_i-\beta)}} < \frac{L}{1+e^{-\alpha(x'_i-\beta)}}$$

where $x_i = (\sum_{d\in\mathcal{D}} b_i^d)/|\mathcal{D}|$. Since $L, \alpha, \beta > 0$, we have $E_{\mathbf{p}_{-i}}(p_i) < E_{\mathbf{p}_{-i}}(p'_i) \Rightarrow x_i < x'_i$. Therefore,

$$\frac{\sum_{d\in\mathcal{D}} b_i^d}{|\mathcal{D}|} < \frac{\sum_{d\in\mathcal{D}} b_i^{d'}}{|\mathcal{D}|}, \text{ which implies } \sum_{d\in\mathcal{D}} b_i^d < \sum_{d\in\mathcal{D}} b_i^{d'} \tag{10}$$

Based on (10), we have:

$$\phi_{\mathbf{p}_{-i}}(p_i) - \phi_{\mathbf{p}_{-i}}(p'_i) = (\sum_{d\in\mathcal{D}} b_i^d - \sum_{d\in\mathcal{D}} b_i^{d'}) \sum_{u_j\neq u_i} \sum_{d\in\mathcal{D}} b_j^d < 0$$

Therefore, $\phi_{\mathbf{p}_{-i}}(p_i) < \phi_{\mathbf{p}_{-i}}(p'_i)$, i.e., Theorem 1 holds.

## Appendix C
## Proof of Theorem 2

According to (7), we have:

$$0 \leq \phi_{\mathbf{p}_{-i}}(p_i) \leq \frac{1}{2} \sum_{u_i\in\mathcal{U}} \sum_{u_j\in\mathcal{U}} Q_{max} = \frac{1}{2} n^2 Q_{max}^2$$

A decision change from $p_i$ to $p'_i$ of a user $u_i$ leads to an increase in the system benefit defined in Section III-C, i.e., $E_{\mathbf{p}}(p_i) < E_{\mathbf{p}}(p'_i)$. According to Definition 5, it also results in an increase in the potential function $\phi$, denoted by $\delta_i$, i.e.,

$$\phi_{\mathbf{p}_{-i}}(p_i) + \delta_i \leq \phi_{\mathbf{p}}(p'_i)$$

According to the proof of Theorem 1, we have:

$$\phi_{\mathbf{p}}(p'_i) - \phi_{\mathbf{p}}(p_i) = (\sum_{d\in\mathcal{D}} b_i^{d'} - \sum_{d\in\mathcal{D}} b_i^d) \sum_{u_j\neq u_i} \sum_{d\in\mathcal{D}} b_j^d > 0$$

Since $b_i^d$ is the normalized amount of type-$d$ resource required by user $u_i$, we have $\sum_{d\in\mathcal{D}} b_i^{d'} - \sum_{d\in\mathcal{D}} b_i^d \geq 1$. Thus,

$$\phi_{\mathbf{p}}(p'_i) - \phi_{\mathbf{p}}(p_i) \geq 1 \sum_{u_j\neq u_i} \sum_{d\in\mathcal{D}} b_j^d = \sum_{u_j\neq u_i} Q_j \geq (n-1)Q_{min} = \delta_i$$

We have $\delta_i = (n-1)Q_{min}$ representing the minimum improvement of the potential function between two iterations, and $\frac{1}{2}n^2Q_{max}^2$ representing the maximum value of the potential function. Therefore, the number of iterations required satisfies:

$$R \leq \frac{n^2 Q_{max}^2}{2(n-1)Q_{min}}$$

Therefore, Theorem 2 holds.

## Appendix D
## Proof of Theorem 3

Case 1: For any allocation decision profile $\mathbf{p} \in \chi$, we have $QoE(\mathbf{p}) \leq QoE(\mathbf{p}^*)$. Thus, $1 \geq poa_{QoE}(\mathbf{p})$.

Case 2: For any allocation decision profile $\mathbf{p} \in \chi$ that is not the optimal decision profile ($\mathbf{p} \neq \mathbf{p}^*$), there is at least one user not allocated to the most suitable edge server or assigned the most suitable QoE level. The minimum QoE incurred by this user is $QoE_{min}(\mathbf{p})$. Thus, the minimum total QoE incurred by $\mathbf{p}$ is:

$$\sum_{u_i\in\mathcal{U}} QoE_{min}(\mathbf{p}) I_{\{p_i\neq(0,0)\}}$$

Similarly, for the optimal decision profile $\mathbf{p}^*$, the maximum QoE of one user is $QoE_{max}(\mathbf{p}^*)$, and the maximum total QoE incurred by $\mathbf{p}^*$ is:

$$\sum_{u_i\in\mathcal{U}} QoE_{max}(\mathbf{p}^*) I_{\{p_i\neq(0,0)\}}$$

Therefore, the minimum PoA of the total QoE is

$$poa_{QoE}(\mathbf{p}) \geq \frac{\sum_{u_i\in\mathcal{U}} QoE_{min}(\mathbf{p}) I_{\{p_i\neq(0,0)\}}}{\sum_{u_i\in\mathcal{U}} QoE_{max}(\mathbf{p}^*) I_{\{p_i\neq(0,0)\}}}$$

Combining Case 1 and Case 2, we prove the theorem:

$$1 \geq poa_{QoE}(\mathbf{p}) \geq \frac{\sum_{u_i\in\mathcal{U}} QoE_{min}(\mathbf{p}) I_{\{p_i\neq(0,0)\}}}{\sum_{u_i\in\mathcal{U}} QoE_{max}(\mathbf{p}^*) I_{\{p_i\neq(0,0)\}}}$$

## References


[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the 1st MCC Workshop on Mobile Cloud Computing*. ACM, 2012, pp. 13–16.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

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

[4] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, "Improving web sites performance using edge servers in fog computing architecture," in *Proceedings of International Symposium on Service-Oriented System Engineering*. IEEE, 2013, pp. 320–323.

[5] N. Takahashi, H. Tanaka, and R. Kawamura, "Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing," in *Proceedings of International Conference on Mobile Cloud Computing, Services, and Engineering*. IEEE, 2015, pp. 233–234.

[6] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proceedings of International Conference on Computer Communications*. IEEE, 2018, pp. 207–215.

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

[8] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[9] X. Cao, J. Zhang, and H. V. Poor, "An optimal auction mechanism for mobile edge caching," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2018, pp. 388–399.

[10] M. S. Mushtaq, B. Augustin, and A. Mellouk, "Empirical study based on machine learning approach to assess the qos/qoe correlation," in *Proceedings of European Conference on Networks and Optical Communications*. IEEE, 2012, pp. 1–7.

[11] H.-J. Hong, C.-F. Hsu, T.-H. Tsai, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Enabling adaptive cloud gaming in an open-source cloud gaming platform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 2078–2091, 2015.

[12] J. Zheng, Y. Cai, Y. Liu, Y. Xu, B. Duan, and X. S. Shen, "Optimal power allocation and user scheduling in multicell networks: Base station cooperation using a game-theoretic approach," *IEEE Transactions on Wireless Communications*, vol. 13, no. 12, pp. 6928–6942, 2014.

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

[14] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, H. Liu, Y. Qin, and P. Chen, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proceedings of International Conference on Web Services*. IEEE, 2019, pp. 91–98.

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

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

[17] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *Proceedings of International Symposium on Multimedia*. IEEE, 2011, pp. 494–499.

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

[19] Y. Zeng, Y. Huang, Z. Liu, and Y. Yang, "Joint online edge caching and load balancing for mobile data offloading in 5g networks," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2019, pp. 923–933.

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

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

[22] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[23] M. Rajiullah, A. Lutu, A. S. Khatouni, M.-R. Fida, M. Mellia, A. Brunstrom, O. Alay, S. Alfredsson, and V. Mancuso, "Web experience in mobile networks: Lessons from two million page visits," in *Proceedings of The Web Conference*. ACM, 2019, pp. 1532–1543.

[24] J. Nejati and A. Balasubramanian, "An in-depth study of mobile browser performance," in *Proceedings of International Conference on World Wide Web*. ACM, 2016, pp. 1305–1315.

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

[26] 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. International Society for Optics and Photonics, 2015, p. 93960Y.

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

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

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

[30] R. B. Myerson, *Game theory*. Harvard University Press, 2013.

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

[32] W. H. Chin, Z. Fan, and R. Haines, "Emerging technologies and research challenges for 5g wireless networks," *IEEE Wireless Communications*, vol. 21, no. 2, pp. 106–112, 2014.

[33] B. Yang, Z. Li, S. Chen, T. Wang, and K. Li, "Stackelberg game approach for energy-aware resource allocation in data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 12, pp. 3646–3658, 2016.

[34] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for joint cooperation and competition of edge resource allocation," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2019, pp. 503–513.

[35] T. Hobfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of qoe management for cloud applications," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 28–36, 2012.

[36] Z. Su, Q. Xu, M. Fei, and M. Dong, "Game theoretic resource allocation in media cloud with mobile social users," *IEEE Transactions on Multimedia*, vol. 18, no. 8, pp. 1650–1660, 2016.

[37] J. He, Y. Wen, J. Huang, and D. Wu, "On the cost–qoe tradeoff for cloud-based video streaming under amazon ec2's pricing models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 669–680, 2013.

[38] M. Chen, Y. Zhang, Y. Li, S. Mao, and V. C. Leung, "Emc: Emotion-aware mobile cloud computing in 5g," *IEEE Network*, vol. 29, no. 2, pp. 32–38, 2015.

[39] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of experience (qoe)-aware placement of applications in fog computing environments," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, 2019.

[40] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "Mefore: Qoe based resource estimation at fog to enhance qos in iot," in *Proceedings of International Conference on Telecommunications*. IEEE, 2016, pp. 1–5.

[41] S.-T. Hong and H. Kim, "Qoe-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds," in *Proceedings of International Conference on Sensing, Communication, and Networking*. IEEE, 2016, pp. 1–9.