QoE-aware User Allocation in Edge Computing Systems with Dynamic QoS☆

Phu Lai[a], Qiang He[a,*], Guangming Cui[a], Xiaoyu Xia[b], Mohamed Abdelrazek[b], Feifei Chen[b], John Hosking[c], John Grundy[d], Yun Yang[a]

[a]*Swinburne University of Technology, Hawthorn, Australia*
[b]*Deakin University, Burwood, Australia*
[c]*The University of Auckland, Auckland, New Zealand*
[d]*Monash University, Clayton, Australia*

Abstract

As online services and applications are moving towards a more human-centered design, many 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, autonomous vehicles, critical warning systems and so on. Edge computing has then been introduced as an effort to reduce network latency. In a mobile edge computing system, edge servers are usually deployed at, or near cellular base stations, offering processing power and low network latency to users within their proximity. In this work, we tackle the edge user allocation (EUA) problem from the perspective of an app vendor, who 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 leads to a different QoE level. Thus, the app vendor also needs to decide the QoS level for each user so that the overall user experience is maximized. We first optimally solve this problem using Integer Linear Programming technique. Being an $\mathcal{NP}$-hard problem, it is intractable to solve it optimally in large-scale scenarios. Thus, we propose a heuristic approach that is able to effectively and efficiently find sub-optimal solutions to the QoE-aware EUA problem. We conduct a series of experiments on a real-world dataset to evaluate the performance of our approach against several state-of-the-art and baseline approaches.

Keywords: User allocation, Edge computing, Quality of Service, Quality of

☆This is an extended and revised version of a preliminary conference paper that was presented in ICSOC 2019 [1].

*Corresponding author

Email addresses: tlai@swin.edu.au (Phu Lai), qhe@swin.edu.au (Qiang He), gcui@swin.edu.au (Guangming Cui), xiaoyu.xia@deakin.edu.au (Xiaoyu Xia), mohamed.abdelrazek@deakin.edu.au (Mohamed Abdelrazek), feifei.chen@deakin.edu.au (Feifei Chen), j.hosking@auckland.ac.nz (John Hosking), john.grundy@monash.edu (John Grundy), yyang@swin.edu.au (Yun Yang)

1. Introduction

Mobile and Internet-of-Things (IoT) devices, including smartphones, smart appliances, environmental sensors, etc., have become an integrated part of the modern society [2]. Its rapid growth has fueled the diversity and sophistication of online applications and services such as real-time facial recognition [3], interactive gaming [4], etc., which require intensive computing power and cause high power consumption. Since thin clients, such as mobile and IoT devices, have limited processing capabilities and battery power, heavy computing tasks are often offloaded to a remote cloud server. Nevertheless, the skyrocketing number of connected devices, the busy network traffic, and the continuously increasing computational workloads are posing the challenge of maintaining a low-latency connection to end-users for app vendors.

Fog computing – sometimes referred to as *edge computing* [5] – has been proposed and implemented to facilitate services that require real-time decision making in large scale [6]. A mobile edge computing (MEC) system involves numerous distributedly deployed edge servers, usually near cellular base stations [7]. This highly distributed architecture remarkably reduces end-to-end network latency, thanks to the close distance between edge servers and end-users. To avoid non-service areas, i.e., the areas that are not covered by any edge server, the coverage of adjacent edge servers are partially overlapped in most cases [8, 9, 10]. A user in the overlapping area can be allocated to one of its neighbor edge servers (*proximity constraint*) that have adequate computing resources (CPU, RAM, storage, or bandwidth – *capacity constraint* [1]). In this paper, we study the quasi-static scenarios where users are relatively static, not roaming across edge servers quickly [11, 8, 9, 4], e.g. surveillance cameras, traffic sensors, mobile or IoT users who stay in one location. Table 1 lists the acronyms used in this paper and their descriptions.

Table 1: Glossary of acronyms used in this paper

Acronyms	Description
MEC	Mobile Edge Computing
IoT	Internet of Things
EUA	Edge User Allocation
QoS	Quality of Service
QoE	Quality of Experience
QoEUA	QoE-aware User Allocation, i.e,. the name of our proposed heuristic
ILP	Integer Linear Programming

[1] Edge server capacity and computing resources are used interchangeably in some parts of this paper.

From the perspective of an app vendor, the users-to-edge-servers allocation needs to be determined so that some optimization objectives are satisfied. This problem is referred to as an edge user allocation (EUA) problem [10]. An edge computing system is immensely dynamic and heterogeneous. Many applications and services support the dynamic quality of service (QoS), or different levels of service performance, which can be represented by display resolution [12], frame rate and bitrate [13], data rate [14], network loss and jitter [15], resource consumption [16, 17], etc. Therefore, apart from deciding which edge servers to serve which users, an app vendor also needs to decide the QoS level for each user. Naturally, a higher QoS level is achieved by a series of computation tasks with higher complexity, thus consumes more computing resources. For example, high-definition graphics rendering or highly accurate data analysis would require more CPU, RAM, or bandwidth, on an edge server. Compared to a datacenter server in cloud computing, a typical edge server possesses a very limited amount of computing resources [18]. Thus, greedily assigning high QoS levels to users would exhaust edge servers' computing resources even quicker.

Unlike the computation offloading problem which challenges the edge infrastructure providers, the EUA problem poses a new challenge to app vendors, who hire computing resources at the edge to serve their users. Recently, this problem has been attracting a lot of attention [19, 9, 10]. Nevertheless, none of the existing works tackles user quality of experience (QoE), which is a critical aspect in any user-oriented, human-centered applications and services. In this work, we solve the EUA problem with the objective of maximizing the total QoE of all users in a particular area. A common consensus is that a higher QoS level leads to a greater QoE level experienced by users. Since a high QoS level is usually very resource-demanding, it is not always possible to serve users with high QoS levels. Many researches [15, 20, 21] have demonstrated a quantitative relationship between QoS and QoE (Figure 1). In this model, there is a characteristic that can be leveraged by app vendors to effectively utilize the edge server's computing resources. Generally, a user's QoE increases with an increase in the 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 any further increases in the QoS level. Therefore, any QoS level higher than W_3 might not benefit users much.

We refer to the above problem as a *QoE-aware edge user allocation* (EUA) problem and make the following main contributions in this paper:

- We formally define the QoE-aware EUA problem and show that it is an $\mathcal{NP}$-hard problem.
- We propose an optimal approach based on Integer Linear Programming (ILP) for solving this problem exactly.
- In our previous work [1], we introduced a heuristic approach to tackle the complexity of the problem. However, its effectiveness under resource-scarce scenarios needs to be improved. As a result, in this paper, we

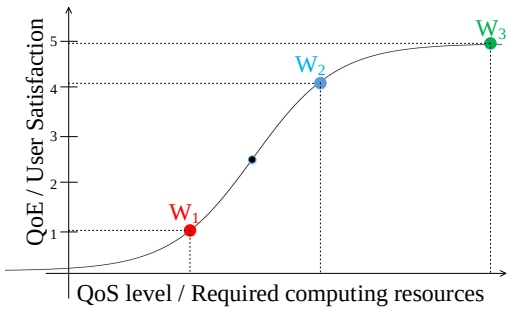


Figure 1: Quality of Experience - Quality of Service correlation

introduce QoEUA – a new heuristic that performs better than our previous heuristic under resource-scarce circumstances.

- Comprehensive experiments based on a real-world dataset are conducted to demonstrate the effectiveness and efficiency of QoEUA against several baseline and state-of-the-art approaches.

The rest of this paper is organized as follows. Section 2 provides an example that motivates this research. Section 3 formulates the QoE-aware problem. Section 4 proposes two approaches to this problem – an optimal approach to find exact solutions and an efficient heuristic to find sub-optimal solutions. Section 5 evaluates the proposed approaches. Section 6 reviews the existing literature. Finally, we conclude the paper and point out future work in Section 7.

2. Motivating Example

Consider a typical game streaming service as an example, game video frames are rendered on the game vendor's servers then streamed to player's devices. For most players, the difference between 1080p and 1440p display resolution is barely perceptible on a mobile device, or even between 1080p and UHD from a distance farther than 1.5x the screen height regardless of the screen size [22]. Thus, it might be unnecessary to serve users with 1440p or UHD video resolution since it would certainly consume more computing resources (CPU, RAM, or bandwidth). Instead, those resources could be utilized to serve players who are unsatisfied with their existing service, e.g. those experiencing poor graphics, or those not being able to play at all due to all nearby edge servers being overloaded. Thus, the game vendor can lower the QoS levels of high-demanding players, potentially without any noticeable QoE downgrade, to better service unsatisfied players. In this way, the game vendor can maximize its players' overall satisfaction, measured by their total QoE.

In this context, our research targets at allocating app users to edge servers and selecting 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 $\langle \text{CPU}, \text{RAM},$

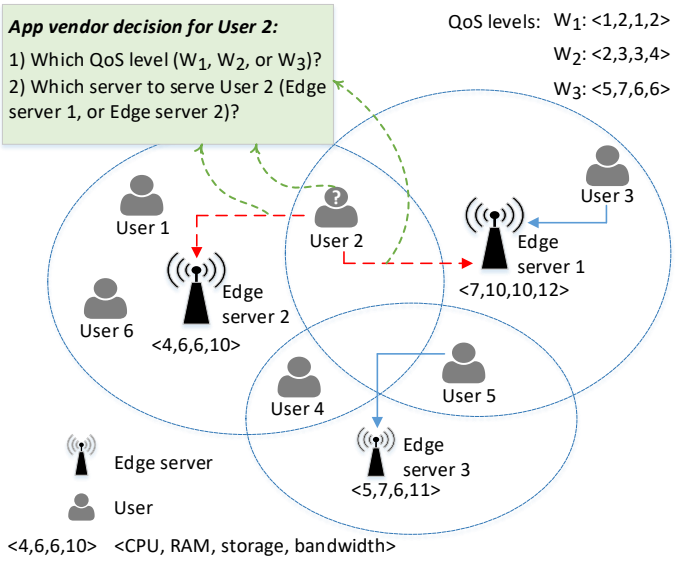


Figure 2: EUA example scenario with dynamic QoS

storage, bandwidth⟩, 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. Thus, allocating user 2 to edge server 1 would result in a higher total QoE.

The QoE-aware EUA problem in real-world scenarios is significantly more complex than this example. It is not always possible to find an optimal allocation in a prompt manner. Therefore, app vendors need to have an efficient yet effective approach for finding a near-optimal solution to this problem.

3. Problem Definition

This section defines the QoE-aware EUA problem. Table 2 summarizes the notations and definitions used throughout this paper. Given a finite set of m edge servers $\mathcal{S} = \{s_1, s_2, ..., s_m\}$, and n users $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ in a particular area, we aim to allocate users to edge servers so that the total user satisfaction, i.e. QoE, is maximized. In the EUA problem, every user covered by edge servers must be allocated to an edge server unless all the nearby edge servers have reached their maximum resource capacity. If a user is not located in the coverage areas of any edge servers, or the computing resources of all nearby servers have been exhausted, they will be directly connected to the app vendor's central cloud server.

Table 2: Key Notations

Notation	Description
$\mathcal{S} = \{s_1, s_2, ..., s_m\}$	a finite set of edge server s_j, where $j = 1, 2, ..., m$.
$\mathcal{D} = \{CPU, RAM,$ $storage, bandwidth\}$	a set of computing resource types.
$c_j = \langle c_j^1, c_j^2, ..., c_j^d \rangle$	the available capacity of an edge server $s_j \in \mathcal{S}$. c_j is a $d-$dimensional vector with each dimension c_j^k representing the available amount of resource type $k \in \mathcal{D}$ on edge server s_j.
$\mathcal{U} = \{u_1, u_2, ..., u_n\}$	a finite set of user u_i, where $i = 1, 2, ..., n$.
$\mathcal{W} = \{W_1, W_2, ..., W_q\}$	a set of predefined QoS level W_l, where $l = 1, 2, ..., q$. A higher QoS level requires more computing resources than a lower one.
$w_i = \langle w_i^1, w_i^2, ..., w_i^d \rangle$	a $d-$dimensional vector representing the amount of computing resources required by user $u_i \in \mathcal{U}$. w_i is selected from the set $\mathcal{W}$, $w_i \in W$.
$\mathcal{U}(s_j)$	a set of users allocated to edge server $s_j, \mathcal{U}(s_j) \subseteq \mathcal{U}$.
$\mathcal{S}(u_i)$	a set of user u_i's neighbor edge servers – edge servers that cover user u_i, $\mathcal{S}(u_i) \subseteq \mathcal{S}$.
s_{u_i}	the edge server assigned to serve user u_i, $s_{u_i} \in \mathcal{S}$.
$cov(s_j)$	the coverage of edge server s_j.

A user u_i can only be allocated to an edge server s_j if it is located in server s_j's coverage area $cov(s_j)$ (proximity constraint (1)). We denote $\mathcal{S}_{u_i}$ as the set of all user u_i's neighbor edge servers – those that cover user u_i. Taking Figure 2 for example, user u_4 can be allocated to edge server s_2 or s_3. Server s_1 can serve users u_2, u_3, or/and u_5 as long as it has sufficient computing resources.

$$u_i \in cov(s_j), \forall u_i \in \mathcal{U}; \forall s_j \in \mathcal{S} \tag{1}$$

If a user u_i is allocated to an edge server, it will be assigned a QoS level, which corresponds to a specific required amount of computing resources $w_i = \langle w_i^1, w_i^2, ..., w_i^d \rangle$, where each dimension w_i^k represents the amount of resource type $k \in \mathcal{D}$, e.g. CPU, RAM, storage, or bandwidth. w_i is selected from a predetermined set of q QoS levels $\mathcal{W}$, ranging from low to high resource demanding. The total computing resources assigned to all users allocated to an edge server must not exceed the available capacity of that edge server (capacity constraint (2)). The available capacity of an edge server $s_j \in \mathcal{S}$ is denoted as $c_j = \langle c_j^1, c_j^2, ..., c_j^d \rangle$, where each dimension c_j^k represents the amount of resource type $k \in \mathcal{D}$. In Figure 2, users u_4 and u_5 cannot both receive QoS level W_3 on server s_3 because the total required resources would be 2 x $\langle 5, 7, 6, 6 \rangle = \langle 10, 14, 12, 12 \rangle$, exceeding edge server s_3's available resources $\langle 5, 7, 6, 11 \rangle$.

$$\sum_{u_i \in \mathcal{U}(s_j)} w_i \preceq c_j, \quad \forall s_j \in \mathcal{S} \tag{2}$$

Each QoS level results in a different QoE level. As stated in [15, 21, 20], QoS is non-linearly correlated with QoE. When the QoS reaches a specific level, a user's QoE improves very trivially regardless of a noticeable increase in the QoS. For example, in the model in Figure 1, 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 cost of a little extra resource. Several related works model the correlation between QoE and QoS using the sigmoid function [23, 24, 25]. In this research, we use a logistic function (Equation 3), a generalized version of the sigmoid function, to model the QoS-QoE correlation. This gives us more control over the QoE model, including the QoE growth rate, making the model more generalizable to different domains.

$$E_i = \frac{L}{1 + e^{-\alpha(x_i - \beta)}} \tag{3}$$

where L is the maximum value of QoE, β controls where the QoE growth should be, or the mid-point of the QoE function, α controls the growth rate of the QoE level (how steep the change from the minimum to maximum QoE level is), E_i represents the QoE level given user u_i's QoS level w_i, and $x_i = \sum_{k \in \mathcal{D}} w_i^k / |\mathcal{D}|$, where w_i^k is the normalized amount of type-k resource required by user u_i. We let $E_i = 0$ if user u_i is unallocated. Regarding the applicability of this QoE model in real-world scenarios, this model well aligns with video streaming application [26] and potentially many other applications too. mLab, Huawei's in-house research laboratory focusing on the analysis of Internet usage from a network and end-user experience perspective, shows a mapping of video definition and mean opinion score (i.e., a quantitative representation of user QoE) [27] that closely follows Equation (3).

Our objective is to find a users-to-edge-servers allocation including each individual user's QoS level so that the total QoE of all users is maximized:

$$maximize \sum_{i=1}^{n} E_i \tag{4}$$

where E_i is the QoE level of user u_i, and n is the number of users. The following theorem proves the hardness of the QoE-aware EUA problem.

Theorem 1. *The QoE-aware EUA problem is $\mathcal{NP}$-hard.*

Proof. See Appendix A. □

4. Our Approach

We first formulate the QoE-aware EUA problem as an integer linear programming (ILP) problem to find its optimal solution. After that, we propose a heuristic approach to efficiently solve the problem in large-scale scenarios.

4.1. Integer Linear Programming Model

From the app vendor's perspective, the optimal solution to the QoE-aware EUA problem must achieve the greatest overall user QoE while satisfying a

number of constraints. The ILP model of the QoE-aware EUA problem can be formulated as follows:

$$\text{maximize} \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{q} E^l x_{ijl} \tag{5}$$

$$\text{subject to: } x_{ijl} = 0 \qquad \forall l \in \{1,...,q\}, \forall i,j \in \{i,j | u_i \notin cov(s_j)\} \tag{6}$$

$$\sum_{i=1}^{n}\sum_{l=1}^{q} W_l^k x_{ijl} \leq c_j^k \qquad \forall j \in \{1,...,m\}, \forall k \in \{1,...,d\} \tag{7}$$

$$\sum_{j=1}^{m}\sum_{l=1}^{q} x_{ijl} \leq 1 \qquad \forall i \in \{1,...,n\} \tag{8}$$

$$x_{ijl} \in \{0,1\} \qquad \forall i \in \{1,...,n\}, \forall j \in \{1,...,m\}, \forall l \in \{1,...,q\}$$

x_{ijl} is the binary indicator variable such that,

$$x_{ijl} = \begin{cases} 1, & \text{if user } u_i \text{ is allocated to server } s_j \text{ with QoS level } W_l \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The objective (5) maximizes the total QoE of all users. In (5), n is the number of users, m is the number of edge servers, and q is the number of QoS levels. The QoE level E^l corresponding to a QoS level W_l, $\forall l \in \{1,...,q\}$, can be pre-calculated since the QoS level set $\mathcal{W}$ is pre-determined. Constraint (6) enforces the *proximity constraints*. Users not located within a server's coverage area will not be allocated to that server. A user may be located within the overlapping coverage area of multiple edge servers. *Capacity constraint* (7) makes sure that the aggregate resource demands of all users allocated to an edge server must not exceed the capacity of that server. Constraint family (8) ensures that every user is allocated to at most one edge server with one QoS level. In other words, a user can only be allocated to either an edge server or the app vendor's cloud server.

By solving this ILP problem with an Integer Programming solver, e.g. IBM ILOG CPLEX[2], or Gurobi[3], an optimal solution to the QoE-aware EUA problem can be found.

4.2. Heuristic Approach

Due to the $\mathcal{NP}$-hardness of the problem, computing an optimal solution will be intractable for large-scale scenarios. This is demonstrated in our experimental results presented in Section 5. In this section, we propose QoEUA – an effective and efficient heuristic approach for finding sub-optimal solutions to the QoE-aware EUA problem. The pseudocode is presented in Algorithm 1.

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

Algorithm 1 QoEUA

1: **initialization:**
2: a set of edge servers $\mathcal{S}$, a set of users $\mathcal{U}$, and a set of QoS levels $\mathcal{W}$
3: all users $u_j, \forall u_j \in \mathcal{U}$, are unallocated
4: **end initialization**
5: sort $\mathcal{U}$ in ascending order of the number of neighbor edge servers (i.e., users who are covered by fewer edge servers are prioritized, being the first to be allocated)
6: **repeat**
7: **for** each user $u_i \in \mathcal{U}$ **do**
8: $\mathcal{S}(u_i) \triangleq$ user u_i's neighbor edge servers;
9: allocate user u_i to an edge server $s_j \in \mathcal{S}(u_i)$ which has the most available capacity, and increase user u_i's current QoS level W_l by one level, i.e., W_{l+1}
10: **end for**
11: **until** no users can improve their QoS levels

Given a set of edge servers $\mathcal{S}$, a set of users $\mathcal{U}$, and a set of QoS levels $\mathcal{W}$ (lines 1-4), QoEUA tries to allocate the users in $\mathcal{U}$ to the edge servers in $\mathcal{S}$. Initially, all the users are unallocated. QoEUA first sorts the users in $\mathcal{U}$ in ascending order of the number of their neighbor edge servers (line 5). In other words, users who are covered by fewer edge servers are to be allocated before those who are covered by more edge servers. This helps increase the probability of those users being allocated to edge servers.

The user sorting is then followed by an iterative process (lines 6-11). In each iteration, QoEUA allocates users one by one in the order of their appearances in the sorted list $\mathcal{U}$ (line 7). For each user $u_i \in \mathcal{U}$, QoEUA retrieves the set of its neighbor edge servers $\mathcal{S}(u_i)$, i.e., servers that have the user u_i in their coverage areas (line 8). User u_i is then allocated to an edge server $s_j \in \mathcal{S}(u_i)$ that has the most available capacity (line 9). In this way, edge server s_j will be more likely to have sufficient capacity to accommodate other users or to increase the QoS levels of existing users later on. If user u_i has not been allocated before, it will be assigned the lowest QoS level, i.e., W_1. If user u_i has been allocated a QoS level W_l before, it will try to increase its current QoS level by one level, i.e., W_{l+1}. The resource and proximity constraints must be fulfilled at all times. Note that an allocated user is able to switch edge servers during the allocation process. QoEUA completes when no users can improve their QoS levels anymore.

The running time of QoEUA consists of: (1) p iterations, which costs $\mathcal{O}(p)$, and in each iteration, (2) iterating through all n users, which costs $\mathcal{O}(n)$, and (3) sorting a maximum of m neighbor edge servers for each user, which costs $\mathcal{O}(m \log m)$, to obtain the server that has the most remaining resources. Thus, the overall time complexity of this heuristic approach is $\mathcal{O}(pnm \log m)$, which is p times higher than the heuristic proposed in [1]. However, p is found to be at most only 4 in our experiments.

The high efficiency of QoEUA allows app vendors to continuously run it, or execute it on demand, to respond to user mobility. The consideration of user mobility would not affect the initial problem formulation and the proposed heuristic in some situations. Specifically, when a user moves outside the coverage of its serving edge server, it will be disconnected from the edge server; the occupied computing resources on that edge server will be released; QoEUA will then consider it as a new user. That user, together with other new users who need to be allocated, will be allocated to edge servers based on the rules defined in Algorithm 1, lines 5-11 (one can now consider the set $\mathcal{U}$ as a set of new users who need to be allocated). This is feasible as long as migrating users across edge servers does not incur extra costs, or if the extra costs are trivial. The extra costs could be the migration cost or service reconfiguration cost [28]. In some use cases, those extra costs could be relatively insignificant. Taking video streaming for example [26], where videos encoded with different resolutions are cached on edge servers, which allows a user to access them with low latency, switching the user across edge servers only requires a very small amount of data to be transferred, e.g., which video the user is watching, the position in the video where the user left off, and the resolution of the video. For applications and services where the extra costs are noticeable, the new costs will need to be modeled. Thus, the initial problem formulation and the proposed solution will need to be modified.

5. Experimental Evaluation

In this section, we evaluate the proposed approaches by a series of experiments. All the experiments are conducted on a Windows machine equipped with Intel Core i5-7400T processor (4 CPUs, 2.4GHz) and 8GB RAM. The ILP model in Section 4.1 is solved with IBM ILOG CPLEX Optimizer solver.

5.1. Benchmark Approaches

Our optimal approach, referred to as Optimal hereafter, and the QoEUA heuristic are compared to several baselines and state-of-the-art approaches for solving the EUA problem:

- *Random*: Each user is allocated to a random edge server as long as that server has sufficient remaining 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 server's remaining resources. For example, if the maximum QoS level can be achieved the server is W_2, the user will be randomly assigned either W_1 or W_2.
- *ICSOC19* [1]: This is the greedy-like heuristic proposed in our previous work.
- *TPDS20* [9]: 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 TPDS20 does not consider dynamic QoS, users' QoS levels are randomly pre-specified.

- *ICSOC18* [10]: This work 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 TPDS20, this work does not consider dynamic QoS either. Thus, users are assigned the same QoS levels as TPDS20.

5.2. Experimental Settings

The experiments are conducted on the EUA dataset[4] [10], which contains the geographical locations of end-users and all cellular base stations in Australia. This dataset is also used in [9], [1], and [10] to evaluate TPDS20, ICSOC19, and ICSOC18, respectively.

Edge servers: To capture the characteristics of a MEC environment [29], we simulate a highly dense urban area of 1.8 km^2 covered by 125 base stations, each equipped with an edge server. The coverage radius of each edge server is randomly generated within 100-150m. The computing resources available on the edge servers, or their capacities, are randomly generated by following a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the average capacity of each resource type in $\mathcal{D}$, and the standard deviation $\sigma = 10$ for all conducted experiments.

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 $\mathcal{D} = \{\text{CPU}, \text{RAM}, \text{storage}, \text{bandwidth}\}$. Those four types of resources are the representative ones. The proposed approaches can accommodate other types of resources that are specific to app vendors' applications. We have conducted experiments with other settings and achieved similar results. Thus, we select those three QoS levels as representative in our experiments. Different values for the QoE model (3) have also been tested. In the experiments, we set $L = 5$, $\alpha = 1.5$, and $\beta = 2$ as representative.

To comprehensively analyze the performance of our approaches 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 server capacity. Table 3 summarizes the settings of the experiments, which will be discussed in the next section. Note that the values specified in the table are representative. Other experiments with different values other than those have been conducted, which yield similar results. 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 3, we compare the total QoE of all users achieved by the six approaches, the higher the better. In addition, we measure the number of users allocated to edge servers by each approach, the higher the better. The efficiency of all approaches is also evaluated.

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

Table 3: Experimental Settings

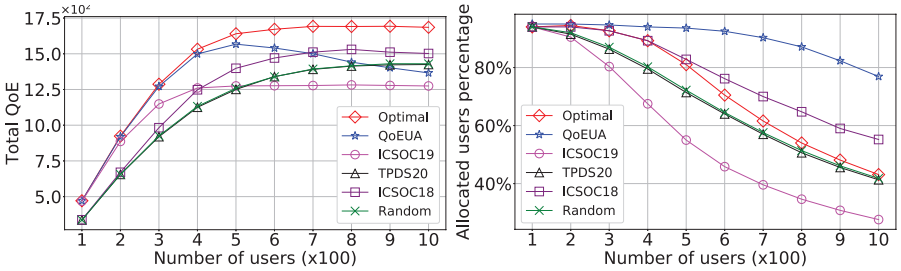
	Number of users	Number of servers	Edge server's capacity
Set #1	$100, 200, ..., 1000$	50%	35
Set #2	500	$10\%, 20\%, ..., 100\%$	35
Set #3	500	50%	$15, 20, ..., 60$

5.3. Experimental Results and Discussion

1) Effectiveness: Figures 3, 4, and 5 depict the results of three experiment Sets #1, #2, and #3, respectively, measured by the overall QoE of all users in the experiment. We additionally measure the number of users allocated to edge servers. In general, Optimal, being the optimal approach, obviously achieves the greatest QoE under all experimental settings, closely followed by QoEUA.

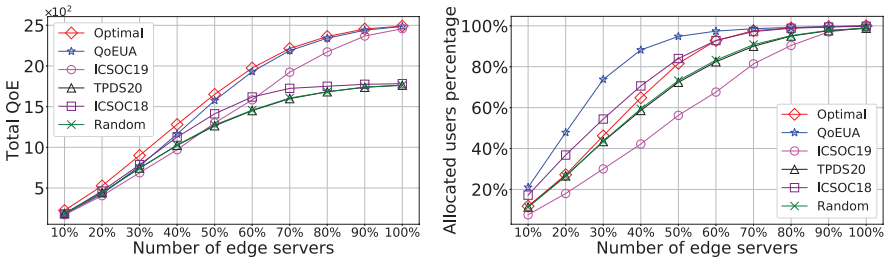
In experiment Set #1 (Figure 3), we vary the number of users from 100 to 1,000 in steps of 100. In general, as the number of users increases, the total QoE also increases until it can no longer increase since the computing resources are exhausted to serve a large number of users. From 100 to 600 users, QoEUA achieves a higher total QoE than other approaches (Figure 3a). Especially in the first four steps (100, 200, 300, and 400 users), QoEUA is almost as good as Optimal. This occurs in those scenarios because the available resources are redundant and therefore almost all the users receive the highest QoS level. However, as the number of users continues to increase while the amount of available resources is fixed, the average computing resources for each user become more scarce, making QoEUA start to downgrade. As we can see, from 500 users onward, the total QoE achieved by QoEUA slowly decreases and starts being outperformed by ICSOC18, TPDS20, and Random at some point. Still, the differences in the total QoE between QoEUA and those approaches are very marginal, even at 1,000 users. Despite being outperformed in terms of the total QoE in some cases, QoEUA is able to allocate significantly more users to edge servers than all other approaches (Figure 3b). As we keep increasing the number of users to be allocated, QoEUA allocates approximately 20% more users compared to other approaches on average. Given 1,000 users, QoEUA can allocate almost 80% of them while the second-best approach can only allocate roughly 60% of them. For more experimental results on the percentage of users allocated with different QoS levels, please refer to Appendix B.

In experiment Sets #2 and #3, we vary the number of edge servers available to serve users (Figure 4) and edge server capacity (Figure 5). Increasing those two parameters consequently increases the redundancy of computing resources available to serve users. As a result, we can observe the same trend in those two experiment sets, where the total QoE and the percentage of users allocated increase with the increase in the number of edge servers and the edge servers' capacities. This pattern can be observed under all experimental settings and for all approaches. In terms of the total QoE of all users (Figures 4a and 5a), QoEUA's performance is very close to Optimal, just slightly lower. In the meantime, QoEUA manages to allocate the most number of users, considerably



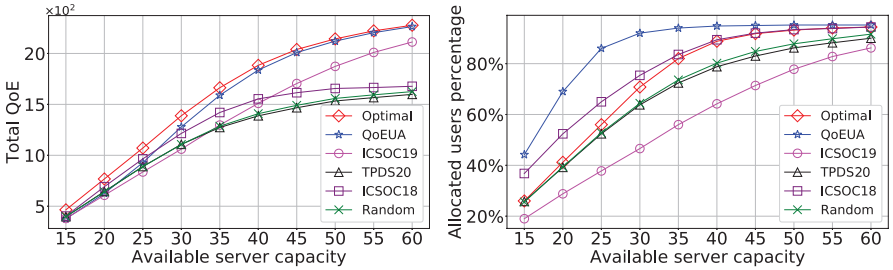
(a) Number of users vs. Total QoE (b) Number of users vs. Percentage of users allocated

Figure 3: Experiment Set #1 results



(a) Number of edge servers vs. Total QoE (b) Number of edge servers vs. Percentage of users allocated

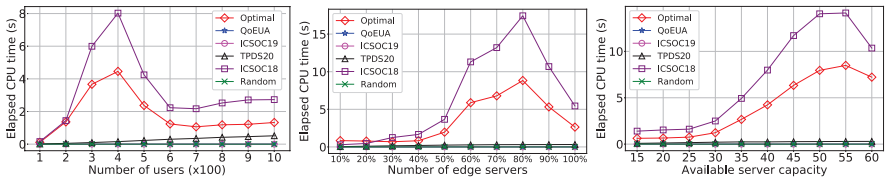
Figure 4: Experiment Set #2 results



(a) Edge server capacity vs. Total QoE (b) Edge server capacity vs. Percentage of users allocated

Figure 5: Experiment Set #3 results

greater than all other approaches under any experimental settings (Figures 4b and 5b). Clearly, QoEUA significantly outperforms the baseline and state-of-the-art approaches in experiment Sets #2 and #3. For more experimental results on the percentage of users allocated with different QoS levels, please



(a) Number of users vs. Execution CPU time (b) Number of edge servers vs. Execution CPU time (c) Edge server capacity vs. Execution CPU time

Figure 6: Execution CPU time in experiment Sets #1, #2, and #3

refer to Appendix B.

2) Efficiency: Figure 6 illustrates the efficiency of all the approaches in the study, measured by the elapsed CPU time. The execution time of Optimal and ICSOC18 follow a similar pattern in all three experiment sets. As the experimental parameters increase from the starting point to a threshold somewhere in the middle – 400 users in Set #1, 80% of the number of edge servers in Set #2, and 55 average server resource capacity in Set #3 – the time rapidly rises until it reaches a cap of around a hefty 8 seconds (for Optimal), which is unacceptable for real-time, latency-sensitive applications. This is expected since the QoE-aware EUA problem is $\mathcal{NP}$-hard. The rationale for this lies in the complexity of the problem. Adding up more users, edge servers, and available resources generates more possible options and solutions for Optimal to select from. After passing those thresholds, the execution time decreases then tends to converge. This happens in experiment Set #1 (Figure 6a) since when the number of users exceeds 400, a newly generated user tends to be positioned at the exact location of an existing user. Thus, the IBM CPLEX solver's decision to be made for the new user can be based on the decision made for the existing user. As a result, we can see the elapsed CPU time of Optimal is almost symmetrical around that threshold (400 users) then roughly stabilizes at around 1 second, which is still very slow for real-time applications. In experiment Sets #1 and #2, the execution time decreases because after the experimental parameters exceed the above-mentioned thresholds, the available computing resources gradually become more redundant so that more users can obtain the highest QoS level without having to compete with each other. This generates fewer possible options for Optimal, thus takes less time to complete.

As the complexity of the problem increases by adding more users, edge servers, and edge server capacity, the execution time of other approaches also increases gradually. In all experiment sets, ICSOC19 takes 0.5-1 millisecond to solve the allocation problem. QoEUA is an iterative algorithm. The number of iterations it takes to complete is an important indicator of its efficiency. In the experiments, QoEUA requires 2-4 iterations, or 1-3 milliseconds of CPU time, which is acceptable for real-time applications and services.

5.4. Threats to Validity

Threat to construct validity. The main threat to construct validity lies in the bias in our experimental design. To minimize any potential bias, we conduct experiments with different varying parameters that would directly affect the experimental results, including the number of edge servers, the number of users, and available computing resources (edge server capacity). The result of each experiment set is the average of 100 executions, each with a different user distribution, to neutralize special cases such as over-dense or over-sparse user distributions.

Threat to external validity. A threat to external validity is the generalizability of our findings in other specific domains. We mitigate this threat by experimenting with different numbers of users and edge servers in the same geographical area to simulate various distributions and density levels of users and edge servers, which might be observed in different real-world scenarios. Furthermore, we employ a generic QoS-QoE model in this work to improve the generalizability.

Threat to internal validity. A threat to internal validity is whether an experimental condition makes a difference or not. To minimize this, we fix the other experimental parameters at a neutral value while changing a parameter. For more sophisticated scenarios where two or more parameters change simultaneously, the results can easily be predicted in general based on the obtained results as we mentioned in Section 5.3.

Threat to conclusion validity. The lack of statistical tests is the biggest threat to our conclusion validity. This has been mitigated by a comprehensive series of experiments that cover different scenarios varying in both size and complexity. For each set of experiments, the result is averaged over 100 runs of the experiment.

6. Related Work

Edge computing comes with many new unique characteristics, namely location awareness, wide-spread geographical distribution, mobility, a substantial number of nodes, the predominant role of wireless access, the strong presence of streaming and real-time applications, and heterogeneity. Those characteristics allow edge computing to deliver a very broad range of new services and applications at the edge of the network, further extending the existing cloud computing architecture.

QoE management and QoE-aware resource allocation have long been a challenge since the cloud computing era and before that [30]. Su et al. [31] propose a game-theoretic framework for resource allocation among media cloud, brokers and mobile social users that aims at maximizing user's QoE and media cloud's profit. While having some similarity to our work, e.g. the brokers can be seen as edge servers, there are several fundamental architectural differences. The broker in their work acts as a proxy for transferring computation tasks between mobile users and the cloud, whereas our edge server is where the tasks are processed. [32] investigates the cost-QoE trade-off in the virtual machine provi-

sioning problem in a centralized cloud, specific to the video streaming domain. In the aforementioned works, QoE is measured by the processing, playback, and downloading rate. We consider a more general scenario where QoE is measured based on different required amounts of computing resources, which we call QoS levels.

QoE-oriented architecture and resource allocation have started gaining attention in the edge computing area as well. [33] proposes a novel architecture in mobile cloud computing that integrates resource-intensive computing with mobile applications. Their goal is to provide a new generation of personalized, QoE-aware services. [16] and [17] tackle the application placement in edge computing environments. They measure a user's QoE based on three levels (low, medium, and high) of access rate, required computing resources, and processing time. The problem we are addressing, i.e., user allocation, can be seen as the phase after their application placement phase. [34] focuses on the computation offloading scheduling problem in mobile clouds from a networking perspective, where energy and latency must be considered in most situations. They propose a QoE-aware optimal and near-optimal scheduling scheme applied in time-slotted scenarios that take into account the trade-off between mobile energy consumption and latency.

Apart from the aforementioned literature, there are a number of studies of user allocation in edge computing [19, 9, 1, 10]. [19] addresses the user mobility where users may move from one place to another, which requires reallocating users across edge servers. By contrast, we study a quasi-static scenario. Furthermore, they do not consider the dynamics of users' QoS and just measure the capacity of an edge server by a fixed number of users that can be allocated to it. If they had considered the dynamic QoS, the capacity of an edge server would have had to be measured by the available amount of computing resources. The authors of [9] and [10] study a quasi-static scenario but lack the consideration of dynamically adjustable QoS levels and QoE, which are important for applications and services where human plays a prominent role. Therefore, the approaches proposed in those papers are not suitable for solving the QoE-aware EUA problem as demonstrated in Section 5.3. The heuristic approach proposed in [1] is most relevant to our work. Nevertheless, it is very ineffective in resource-scarce scenarios, which are very common in edge computing. Our proposed heuristic (QoEUA) demonstrates a significant improvement compared to the heuristic introduced in [1].

7. Conclusion and Future Work

App users' quality of experience (QoE) is of great importance for app vendors where user satisfaction is taken seriously. Despite being significant, there is very limited work in edge computing considering this aspect. Therefore, we have identified and formally formulated the QoE-aware edge user allocation problem with the goal of maximizing users' overall QoE as the first step of tackling QoE-oriented problems in edge computing. Having been proven to be $\mathcal{NP}$-hard and also experimentally illustrated, the optimal approach is not efficient once the problem scales up. In our previous work [1], we have proposed a heuristic

to deal with the high complexity of the problem. However, that approach is not suitable in resource-scarce scenarios. This has led to the development of a more effective heuristic in this paper. We also conduct extensive experiments on a real-world dataset to evaluate the effectiveness and efficiency of our new approach against several baseline and state-of-the-art approaches.

Given this foundation of the problem, we have identified a number of possible directions for future work with respect to QoE such as QoE-aware user allocation in time-varying situations, user's mobility, service migration, service recommendation, just to name a few. In addition, the QoE model used in this paper is a generic model and might not be very suitable in some other specific domains such as video streaming, web browsing, cloud gaming, and so on. Therefore, a finer-grained, domain-specific QoE model with various types of costs or network conditions could be studied next. Furthermore, QoEUA showed a slight performance degradation when the number of users is large. A new approach could be investigated to deal with this situation.

Acknowledgments. This work was supported by the Australian Research Council Discovery Project [grant numbers DP170101932, DP180100212]; and the Laureate Fellowship [grant number FL190100035].

Appendix A. Proof of Theorem 1

We can prove that the QoE-aware EUA problem defined in Section 3 is $\mathcal{NP}$-hard by proving that its associated decision version is $\mathcal{NP}$-complete. The decision version of QoE-aware EUA is defined as follows:

Given a set of required resources $\mathcal{L} = \{w_1, w_2, ..., w_n\}$ and a set of server resource capacity $\mathcal{C} = \{c_1, c_2, ..., c_m\}$; for each positive number Q determine whether there exists a partition of $\mathcal{L}' \subseteq \mathcal{L}$ into $\mathcal{C}' \subseteq \mathcal{C}$ with aggregate QoE greater than Q, such that each subset of $\mathcal{L}'$ sums to at most $c_j, \forall c_j \in \mathcal{C}'$, and the constraint (1) is satisfied. By repeatedly answering the decision problem, with all feasible combination of $w_i \in \mathcal{W}, \forall i \in \{1, ..., n\}$, it is possible to find the allocation that produces the maximum overall QoE.

First, we show that the QoE-aware EUA problem is $\mathcal{NP}$. Given a solution with m servers and n users, we can easily verify its validity in polynomial time $\mathcal{O}(mn)$ – ensuring each user is allocated to at most one server, and each server meets the condition of having its users' total workload less than or equal to its available resource. QoE-aware EUA is thus in $\mathcal{NP}$ class.

Then, we can prove that the QoE-aware EUA problem is $\mathcal{NP}$-hard by reducing the Partition problem, which is $\mathcal{NP}$-complete [35], to a special case of the QoE-aware EUA decision problem. The Partition problem can be defined as follows: Given a finite sequence of non-negative integers $\mathcal{X} = (x_1, x_2, ..., x_n)$, determine whether there exists a subset $\mathcal{S} \subseteq \{1, ..., n\}$ such that $\sum_{i \in \mathcal{S}} x_i = \sum_{j \notin \mathcal{S}} x_j$.

Each user u_i can be either unallocated to any edge server, or allocated to an edge server with an assigned QoS level $w_i \in \mathcal{W}$. For any instance $\mathcal{X} = (x_1, x_2, ..., x_n)$ of Partition, construct the following instance of the QoE-aware problem: there are n users, where each user u_i has two 2-dimensional QoS level options, $\langle x_i, 0\rangle$ and $\langle 0, x_i\rangle$; and a number of identical servers whose size

is $\langle C, C\rangle$, where $C = \frac{\sum_{i=1}^{n} x_i}{2}$. Assume that all users can be served by any of those servers. Note that $\langle x_i, 0\rangle \equiv \langle 0, x_i\rangle \equiv w_i$. Clearly, there is a solution to QoE-aware EUA that allocates n users to two servers *if and only if* there is a solution to the PARTITION problem. Because this special case is $\mathcal{NP}$-hard, and being $\mathcal{NP}$ as shown above, the general decision problem of QoE-aware EUA is thus $\mathcal{NP}$-complete. Since the optimization problem is at least as hard as the decision problem, the QoE-aware EUA problem is $\mathcal{NP}$-hard, which completes the proof.

Appendix B. Extra experimental results

In this appendix, we present the experimental results on the percentage of users allocated with different QoS levels in three experiment sets (Figures B.7, B.8, and B.9). As set in Section 5.2, there are three possible QoS levels $\mathcal{W} = \{W_1, W_2, W_3\}$, where W_3 is the most demanding QoS level, i.e., requiring the highest amount of computing resources among the three QoS levels, and W_1 is the least demanding one. We only present the results produced by Optimal, QoEUA, and ICSOC19 since they are the most relevant approaches which consider dynamic QoS level. The other approaches just randomly pre-assign QoS levels to users before the allocation process.

Figure B.7 depicts the percentage of unallocated users and users assigned with QoS levels W_1, W_2, and W_3 among all users in experiment Set #1. When the number of users to be allocated is low at 100, there are sufficient resources to accommodate almost all users at the highest QoS level. As the number of users increases, Optimal tends to allocate most users with W_2 since it is the most "economical" option, i.e., highest QoE per unit of resources. Although QoEUA does not tend to pick the most economical option, it is able to allocate more users than Optimal (also refer to Figure 3b). Greedy, being a greedy approach, allocates most users with the highest QoS level.

Figures B.8 and B.9 illustrate the percentage of unallocated users and users assigned with QoS levels W_1, W_2, and W_3 among all users in experiment Sets #2 and #3. Their results follow the same pattern since the two varying experimental parameters, i.e., number of edge servers and available server capacity, both directly affect the amount of available resources which can be used to serve users. When the amount of available resources is low, Optimal assigns W_2, which is the most economical option, to most users. With QoEUA, most users get W_1. As the amount of available resources increases, Optimal and QoEUA can then start assigning higher QoS levels to the users.

References

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

[2] P. Cerwall, A. Lundvall, P. Jonsson, S. Carson, R. Möller, P. Jonsson, S. Carson, P. Lindberg, K. Öhman, I. Sorlie, R. Queirós, F. Muller, L. En-

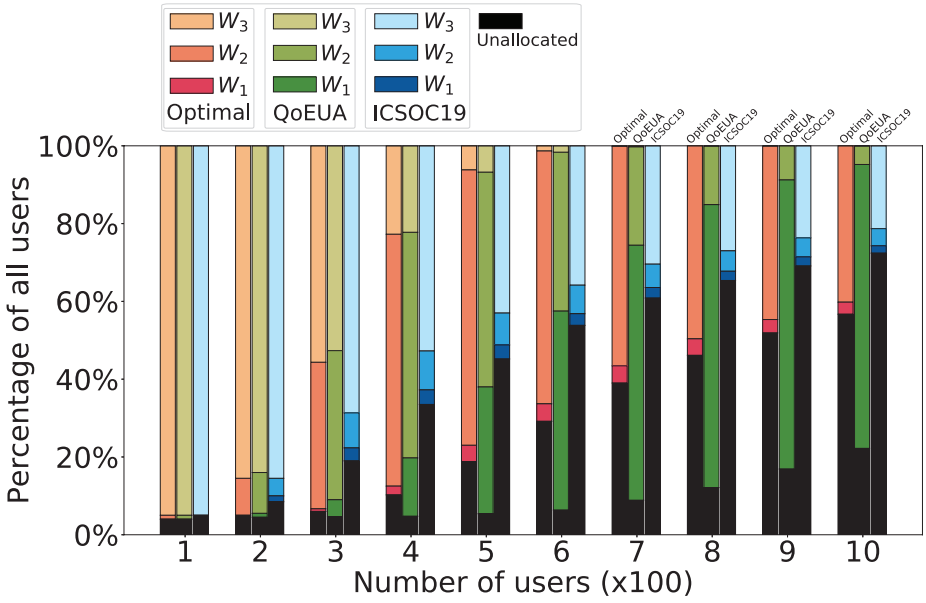


Figure B.7: Proportion of users by QoS levels in experiment Set #1

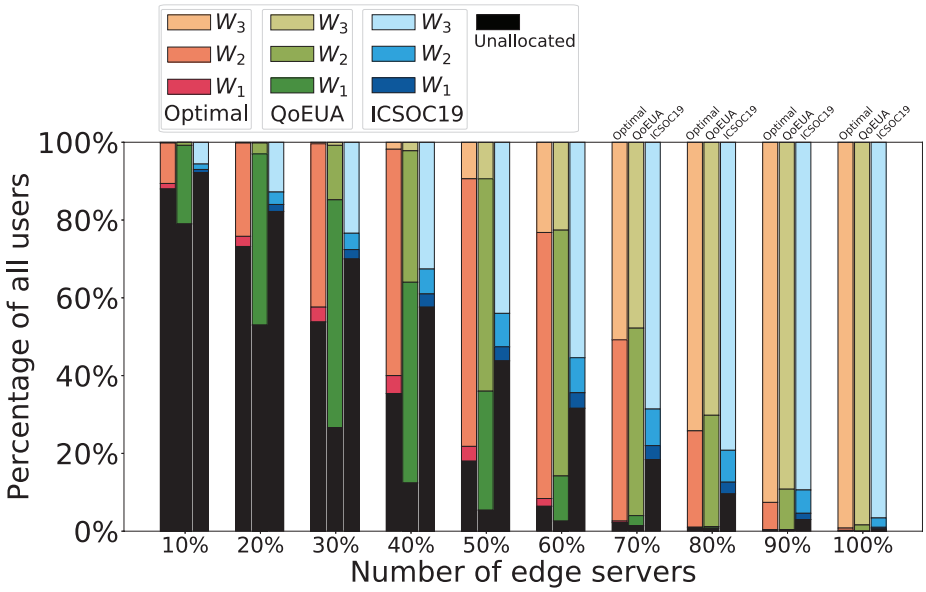


Figure B.8: Proportion of users by QoS levels in experiment Set #2

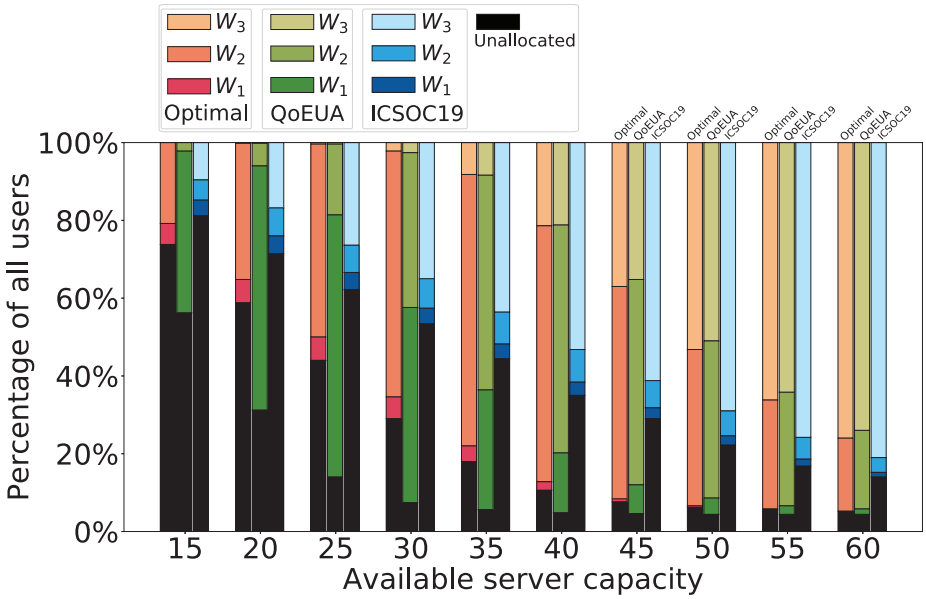


Figure B.9: Proportion of users by QoS levels in experiment Set #3

glund, M. Arvedson, A. Carlsson, Ericsson mobility report, Ericsson, Stockholm (2018).
URL `www.ericsson.com/en/mobility-report/reports/november-2018`

[3] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, W. Heinzelman, Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture, in: Proceedings of IEEE Symposium on Computers and Communications, IEEE, 2012, pp. 59–66.

[4] X. Chen, Decentralized computation offloading game for mobile cloud computing, IEEE Transactions on Parallel and Distributed Systems 26 (4) (2015) 974–983.

[5] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, A. Ahmed, Edge computing: A survey, Future Generation Computer Systems 97 (2019) 219–235.

[6] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of Things, in: Proceedings of Workshop on Mobile Cloud Computing, ACM, 2012, pp. 13–16.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, Mobile edge computing—A key technology towards 5G, ETSI white paper 11 (11) (2015) 1–16.

[8] J. Xu, L. Chen, P. Zhou, Joint service caching and task offloading for mobile edge computing in dense networks, in: Proceedings of IEEE Conference on Computer Communications (INFOCOM), IEEE, 2018, pp. 207–215.

[9] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, Y. Yang, A game-theoretical approach for user allocation in edge computing environment, IEEE Transactions on Parallel and Distributed Systems 31 (3) (2020) 515–529.

[10] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, 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.

[11] L. Chen, S. Zhou, J. Xu, Computation peer offloading for energy-constrained mobile edge computing in small-cell networks, IEEE/ACM Transactions on Networking 26 (4) (2018) 1619–1632.

[12] M. S. Mushtaq, B. Augustin, 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.

[13] H.-J. Hong, C.-F. Hsu, T.-H. Tsai, C.-Y. Huang, K.-T. Chen, C.-H. Hsu, Enabling adaptive cloud gaming in an open-source cloud gaming platform, IEEE Transactions on Circuits and Systems for Video Technology 25 (12) (2015) 2078–2091.

[14] J. Zheng, Y. Cai, Y. Liu, Y. Xu, B. Duan, 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 13 (12) (2014) 6928–6942.

[15] M. Fiedler, T. Hossfeld, P. Tran-Gia, A generic quantitative relationship between quality of experience and quality of service, IEEE Network 24 (2) (2010) 36–41.

[16] R. Mahmud, S. N. Srirama, K. Ramamohanarao, R. Buyya, Quality of Experience (QoE)-aware placement of applications in fog computing environments, Journal of Parallel and Distributed Computing 132 (2019) 190–203.

[17] M. Aazam, M. St-Hilaire, C.-H. Lung, 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.

[18] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet of Things Journal 5 (1) (2017) 450–465.

[19] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, H. Liu, Y. Qin, 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.

[20] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, R. Schatz, Quantification of YouTube QoE via crowdsourcing, in: Proceedings of IEEE International Symposium on Multimedia, IEEE, 2011, pp. 494–499.

[21] M. Alreshoodi, J. Woods, Survey on QoE\QoS correlation models for multimedia services, CoRR abs/1306.0221 (2013). arXiv:1306.0221.
URL www.arxiv.org/abs/1306.0221

[22] A. Lachat, J.-C. Gicquel, 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.

[23] M. Hemmati, B. McCormick, S. Shirmohammadi, QoE-aware bandwidth allocation for video traffic using sigmoidal programming, IEEE MultiMedia 24 (4) (2017) 80–90.

[24] S. Shenker, Fundamental design issues for the future Internet, IEEE Journal on Selected Areas in Communications 13 (7) (1995) 1176–1188.

[25] P. Hande, S. Zhang, M. Chiang, Distributed rate allocation for inelastic flows, IEEE/ACM Transactions on Networking 15 (6) (2007) 1240–1253.

[26] C. Liang, Y. He, F. R. Yu, N. Zhao, Enhancing video rate adaptation with mobile edge computing and caching in software-defined mobile networks, IEEE Transactions on Wireless Communications 17 (10) (2018) 7013–7026.

[27] S. Daryl, Mobile video requires performance and measurement standards, Huawei Global Mobile Broadband Forum (2015).
URL www.huawei.com/minisite/hwmbbf15/img/mvp_online.pdf

[28] L. Wang, L. Jiao, J. Li, J. Gedeon, M. Mühlhäuser, MOERA: Mobility-agnostic online resource allocation for edge computing, IEEE Transactions on Mobile Computing 18 (8) (2018) 1843–1856.

[29] W. H. Chin, Z. Fan, R. Haines, Emerging technologies and research challenges for 5G wireless networks, IEEE Wireless Communications 21 (2) (2014) 106–112.

[30] T. Hobfeld, R. Schatz, M. Varela, C. Timmerer, Challenges of QoE management for cloud applications, IEEE Communications Magazine 50 (4) (2012) 28–36.

[31] Z. Su, Q. Xu, M. Fei, M. Dong, Game theoretic resource allocation in media cloud with mobile social users, IEEE Transactions on Multimedia 18 (8) (2016) 1650–1660.

[32] J. He, Y. Wen, J. Huang, 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 24 (4) (2013) 669–680.

[33] M. Chen, Y. Zhang, Y. Li, S. Mao, V. C. Leung, EMC: Emotion-aware mobile cloud computing in 5G, IEEE Network 29 (2) (2015) 32–38.

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

[35] M. R. Garey, D. S. Johnson, Computers and intractability, vol. 29, W. H. Freeman and Company, New York, 2002.