Edge User Allocation with Dynamic Quality of Service

Phu Lai[1], Qiang He[1(✉)], Guangming Cui[1], Xiaoyu Xia[2], Mohamed Abdelrazek[2], Feifei Chen[2], John Hosking[4], John Grundy[3], and Yun Yang[1]

[1] Swinburne University of Technology, Hawthorn, Australia
{tlai,qhe,gcui,yyang}@swin.edu.au
[2] Deakin University, Burwood, Australia
{xiaoyu.xia,mohamed.abdelrazek,feifei.chen}@deakin.edu.au
[3] Monash University, Clayton, Australia
john.grundy@monash.edu
[4] The University of Auckland, Auckland, New Zealand
j.hosking@auckland.ac.nz

Abstract. In edge computing, edge servers are placed in close proximity to end-users. App vendors can deploy their services on edge servers to reduce network latency experienced by their app users. The edge user allocation (EUA) problem challenges service providers with the objective to maximize the number of allocated app users with hired computing resources on edge servers while ensuring their fixed quality of service (QoS), e.g., the amount of computing resources allocated to an app user. In this paper, we take a step forward to consider dynamic QoS levels for app users, which generalizes but further complicates the EUA problem, turning it into a dynamic QoS EUA problem. This enables flexible levels of quality of experience (QoE) for app users. We propose an optimal approach for finding a solution that maximizes app users' overall QoE. We also propose a heuristic approach for quickly finding sub-optimal solutions to large-scale instances of the dynamic QoS EUA problem. Experiments are conducted on a real-world dataset to demonstrate the effectiveness and efficiency of our approaches against a baseline approach and the state of the art.

Keywords: Resource allocation · Edge computing · Quality of Service · Quality of Experience · User allocation

1 Introduction

Mobile and Internet-of-Things (IoT) devices, including mobile phones, wearables, sensors, etc., have become extremely popular in modern society [4]. The rapid growth of those devices have increased the variety and sophistication of software applications and services such as facial recognition [21], interactive gaming [6], real-time, large-scale warehouse management [7], etc. Those applications usually require intensive processing power and high energy consumption. Due to the limited computing capabilities and battery power of mobile and IoT devices,

S. Yangui et al. (Eds.): ICSOC 2019, LNCS 11895, pp. 86–101, 2019.
https://doi.org/10.1007/978-3-030-33702-5_8

a lot of computing tasks are offloaded to app vendors' servers in the cloud. However, as the number of connected devices is skyrocketing with the continuously increasing network traffic and computational workloads, app vendors are facing the challenge of maintaining a low-latency connection to their users.

Edge computing – sometimes often referred to as *fog computing* – has been introduced to address the latency issue that often occurs in the cloud computing environment [3]. A usual edge computing deployment scenario involves numerous edge servers deployed in a distributed manner, normally near cellular base stations [16]. This network architecture significantly reduces end-to-end latency thanks to the close proximity of edge servers to end-users. The coverage areas of nearby edge servers usually partially overlap to avoid non-serviceable areas – the areas in which users cannot offload tasks to any edge server. A user located in the overlapping area can connect to one of the edge servers covering them (*proximity constraint*) that has sufficient computing resources (*resource constraint*) such as CPU, storage, bandwidth, or memory. Compared to a cloud data-center server, a typical edge server has very limited computing resources, hence the need for an effective and efficient resource allocation strategy.

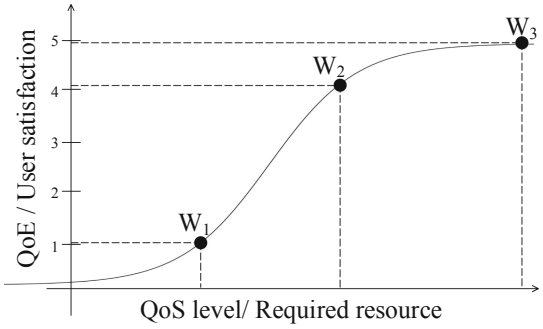


Fig. 1. Quality of Experience - Quality of Service correlation

Naturally, edge computing is immensely dynamic and heterogeneous. Users using the same service have various computing needs and thus require different levels of quality of service (QoS), or computational requirements, ranging from low to high. Tasks with high complexity, e.g. high-definition graphic rendering, eventually consume more computing resources in an edge server. A user's satisfaction, or quality of experience (QoE), varies along with different levels of QoS. Many researchers have found that there is a quantitative correlation between QoS and QoE, as visualized in Fig. 1 [2,8,15]. At one point, e.g. W_3, the user satisfaction tends to converge so that the QoE remains virtually unchanged at the highest level regardless of how high the QoS level is.

Consider a typical game streaming service for example, gaming video frames are rendered on the game vendor's servers then streamed to player's devices. For the majority of players, there is no perceptible difference between 1080p and 1440p video resolution 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 [17]. Servicing a 1440p or UHD video certainly consumes more resources (bandwidth, processing power), which might be unnecessary since most players are likely to be satisfied with 1080p in those cases. 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 app vendor can lower the QoS requirements of high demanding users, potentially without any remarkable downgrade in their QoE, in order to better service users experiencing low QoS levels. This way, app vendors can maximize users' overall satisfaction measured by their overall QoE. In this context, our research aims at allocating app users to edge servers so that their overall QoE is maximized.

We refer to the above problem as a *dynamic QoS edge user allocation* (EUA) problem. Despite being critical in edge computing, this problem has not been extensively studied. Our main contributions are as follows:

- We define and model the dynamic QoS EUA problem, and prove its $\mathcal{NP}$-hardness.
- We propose an optimal approach based on integer linear programming (ILP) for solving the dynamic QoS EUA and develop a heuristic approach for finding sub-optimal solutions to large-scale instances of the problem efficiently.
- Extensive evaluations based on a real-world dataset are carried out to demonstrate the effectiveness and efficiency of our approaches against a baseline approach and the state of the art.

The remainder of the paper is organized as follows. Section 2 provides a motivating example for this research. Section 3.1 defines the dynamic QoS problem and proves that it is $\mathcal{NP}$-hard. We then propose an optimal approach based on ILP and an efficient sub-optimal heuristic approach in Sect. 4. Section 5 evaluates the proposed approaches. Section 6 reviews the related work. Finally, we conclude the paper in Sect. 7.

2 Motivating Example

Using the game streaming example in Sect. 1, let us consider a simple scenario shown in Fig. 2. There are ten players $u_1, ..., u_{10}$, and four edge server $s_1, ..., s_4$. Each edge server has a particular amount of different types of available resources ready to fulfill users' requests. A server's resource capacity or player's resource demand are denoted as a vector $\langle CPU, RAM, storage, bandwidth \rangle$. The game vendor can allocate its users to nearby edge servers and assign a QoS level to each of them. In this example, there are three QoS levels for the game vendor to choose from, namely W_1, W_2 and W_3 (Fig. 1), which consume $\langle 1, 2, 1, 2 \rangle$, $\langle 2, 3, 3, 4 \rangle$, and $\langle 5, 7, 6, 6 \rangle$ units of $\langle CPU, RAM, storage, bandwidth \rangle$, respectively. Players' corresponding QoE, measured based on Eq. 3, are 1.6, 4.09, and 4.99, respectively. If the server's available resources are not limited then all players will be able to enjoy the highest QoS level. However, a typical edge server has relatively limited

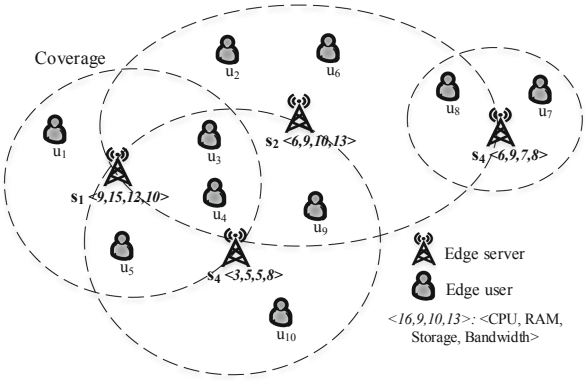


Fig. 2. Dynamic QoS EUA example scenario

resources so not everyone will be assigned W_3. The game provider needs to find a player - server - QoS allocation so that the overall user satisfaction, i.e. QoE, is maximized.

Let us assume server s_2 has already reached its maximum capacity and cannot serve anymore players. As a result, player u_8 needs to be allocated to server s_4 along with player u_7. If player u_8 is assigned the highest QoS level W_3, the remaining resources on server s_4 will suffice to serve player u_7 with QoS level W_1. The resulting total QoE of those two players is 1.6+4.99 = 6.59. However, we can see that the released resources from the downgrade from W_3 to W_2 allows an upgrade from W_1 to W_2. If players u_7 and u_8 both receive QoS level W_1, players' overall QoE is $4.09 + 4.09 = 8.18$, greater than the previous solution.

The scale of the dynamic QoS EUA problem in the real-world scenarios can of course be significantly larger than this example. Therefore, it is not always possible to find an optimal solution in a timely manner, hence the need for an efficient yet effective approach for finding a near-optimal solution to this problem efficiently.

3 Problem Formulation

3.1 Problem Definition

This section defines the dynamic QoS EUA problem. Table 1 summarizes the notations and definitions used in 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 servers accessible for the user have reached their maximum resource capacity. If a user cannot be allocated to any edge servers, or is not positioned within the coverage of any edge servers, they will be directly connected to the app vendor's central cloud server.

Table 1. Key notations

Notation	Description
$\mathcal{S} = \{s_1, s_2, ..., s_m\}$	finite set of edge server s_j, where $j = 1, 2, ..., m$
$\mathcal{D} = \{CPU, RAM, storage, bandwidth\}$	a set of computing resource dimension
$c_j = \langle c_j^1, c_j^2, ..., c_j^d \rangle$	$d-$dimensional vector with each dimension c_j^k being a resource type, such as CPU or storage, representing the available resources of an edge server s_j, $k \in \mathcal{D}$
$\mathcal{U} = \{u_1, u_2, ..., u_n\}$	finite set of user u_i, where $i = 1, 2, ..., n$
$\mathcal{W} = \{W_1, W_2, ..., W_q\}$	a set of predefined resource level W_l, where $l = 1, 2, ..., q$. A higher resource level requires more resource than a lower one $W_l < W_{l+1}$. We will also refer to a resource level as a QoS level.
$w_i = \langle w_i^1, w_i^2, ..., w_i^d \rangle$	$d-$dimensional vector representing the resource amount demanded by user u_i. Each vector component w_i^k is a resource type, $k \in \mathcal{D}$. Each user can be assigned a resource level $w_i \in W$
$\mathcal{U}(s_j)$	set of users allocated to server s_j, $\mathcal{U}(s_j) \subseteq \mathcal{U}$
$\mathcal{S}(u_i)$	set of user u_i's candidate servers – edge servers that cover user u_i, $\mathcal{S}(u_i) \subseteq \mathcal{S}$
s_{u_i}	edge server assigned to serve user u_i, $s_{u_i} \in \mathcal{S}$
$cov(s_j)$	coverage radius of server s_j

A user u_i can only be allocated to an edge server s_j if they are located within s_j's coverage area $cov(s_j)$. We denote $\mathcal{S}_{u_i}$ as the set of all user u_i's candidate edge servers – those that cover user u_i. Take Fig. 2 for example, users u_3 and u_4 can be served by servers s_1, s_2, or s_3. Server s_1 can serve users u_1, u_3, u_4, and u_5 as long as it has adequate 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, they will be assigned a specific amount of computing resources $w_i = (w_i^d)$, where each dimension $d \in \mathcal{D}$ represents a type of resource, e.g. CPU, RAM, storage, or bandwidth. w_i is selected from a predetermined set $\mathcal{W}$ of q resource levels, ranging from low to high. Each of those resource levels corresponds to a QoS level. The total resources assigned to all users allocated to an edge server must not exceed the available resources on that edge server. The available computing resources on an edge server $s_j, s_j \in \mathcal{S}$ are denoted as $c_j = (c_j^d), d \in \mathcal{D}$. In Fig. 2, users u_1, u_3, u_4, and u_5 cannot all receive QoS level W_3 on server s_1 because the total required resources would be $\langle 20, 28, 24, 24 \rangle$, exceeding server s_1's available resources $\langle 9, 15, 12, 10 \rangle$.

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

Each user u_i's assigned resource w_i corresponds to a QoS level that results in a different QoE level. As stated in [2,8,15], 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 Fig. 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 works model the correlation between QoE and QoS using the sigmoid function [10,12,20]. In this research, we use a logistic function (Eq. 3), a generalized version of the sigmoid function, to model the QoS - QoE correlation. This gives us more control over the QoE model, including 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 $x_i = \frac{\sum_{k \in \mathcal{D}} w_i^k}{|\mathcal{D}|}$. We let $E_i = 0$ if user u_i is unallocated.

Our objective is to find a user-server assignment $\{u_1, ..., u_n\} \longrightarrow \{s_1, ..., s_m\}$ with their individual QoS levels $\{w_1, ..., w_n\}$ in order to maximize the overall QoE of all users:

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

3.2 Problem Hardness

We can prove that the dynamic QoS EUA problem defined above is $\mathcal{NP}$-hard by proving that its associated decision version is $\mathcal{NP}$-complete. The decision version of dynamic QoS EUA is defined as follows:

Given a set of demand workload $\mathcal{L} = \{w_1, w_2, ..., w_n\}$ 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.

Theorem 1. *The dynamic QoS EUA problem is $\mathcal{NP}$.*

Proof. 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 or equal than its available resource. Dynamic QoS EUA is thus in $\mathcal{NP}$ class.

Theorem 2. PARTITION $\leq_p$ *dynamic QoS EUA. Therefore, dynamic QoS EUA is* $\mathcal{NP}$*-hard.*

Proof. We can prove that the dynamic QoS EUA problem is $\mathcal{NP}$-hard by reducing the PARTITION problem, which is $\mathcal{NP}$-complete [9], to a specialization of the dynamic QoS EUA decision problem.

Definition 1 *(*PARTITION*).* *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 dynamic QoS 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 dynamic QoS 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 $\mathcal{NP}$, the general decision problem of dynamic QoS EUA is thus $\mathcal{NP}$-complete. Since the optimization problem is at least as hard as the decision problem, the dynamic QoS EUA problem is $\mathcal{NP}$-hard, which completes the proof.

4 Our Approach

We first formulate the dynamic QoS EUA problem as an integer linear programming (ILP) problem to find its optimal solutions. 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 dynamic QoS problem must achieve the greatest QoE over all users while satisfying a number of constraints. The ILP model of the dynamic QoS problem can be formulated as follows:

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

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

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

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

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

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 allocated users. In (5), the QoE level E_l can be pre-calculated based on the predefined set $\mathcal{W}$ of QoS levels $W_l, \forall l \in \{1, ..., q\}$. 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. *Resource constraint* (7) makes sure that the aggregate resource demands of all users allocated to an edge server must not exceed the remaining resources 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[1], or Gurobi[2], an optimal solution to the dynamic QoS EUA problem can be found.

4.2 Heuristic Approach

However, due to the exponential complexity of the problem, computing an optimal solution will be extremely inefficient for large-scale scenarios. This is demonstrated in our experimental results presented in Sect. 5. Approximate methods have been proven to be a prevalent technique when dealing with this type of intractable problems. In this section, we propose an effective and efficient heuristic approach for finding sub-optimal solutions to the dynamic QoS problem.

Heuristic 1. GREEDY

```
1: procedure AllocateEdgeUsers(S,U)
2:     for each u_i ∈ U do
3:         S_{u_i} ← {s_j ∈ S | u_i ∈ cov(s_j)};
4:         if S_{u_i} ≠ ∅ then
5:             s_{u_i} ← argmax_{s_j ∈ {0} ∪ S_{u_i}} {s_j : c_j ≥ W_1};
6:             w_i ← argmax_{W_l ∈ {0} ∪ W} {W_l : W_l ≤ c_j};
7:         end if
8:     end for
9: end procedure
```

The heuristic approach allocates every user $u_i \in \mathcal{U}$ one by one (line 2). For each user u_i, we obtain the set $\mathcal{S}_{u_i}$ of all candidate edge servers that cover that user (line 3). If the set $\mathcal{S}_{u_i}$ is not empty, or user u_i is covered by one or more edge

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

servers, user u_i will then be allocated to the server that has the most remaining resources among all candidate servers (line 5) so that the server will be most likely to have enough resources to accommodate other users. In the meantime, user u_i is assigned the highest QoS level that can be accommodated by the selected edge server (line 6).

The running time of this greedy heuristic consists of: (1) iterating through all n users, which costs $\mathcal{O}(n)$, and (2) sorting a maximum of m candidate 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 $\mathcal{O}(nm \log m)$.

5 Experimental Evaluation

In this section, we evaluate the proposed approaches by an experimental study. All the experiments were conducted on a Windows machine equipped with Intel Core i5-7400T processor(4 CPUs, 2.4 GHz) and 8 GB RAM. The ILP model in Sect. 4.1 was solved with IBM ILOG CPLEX Optimizer.

5.1 Baseline Approaches

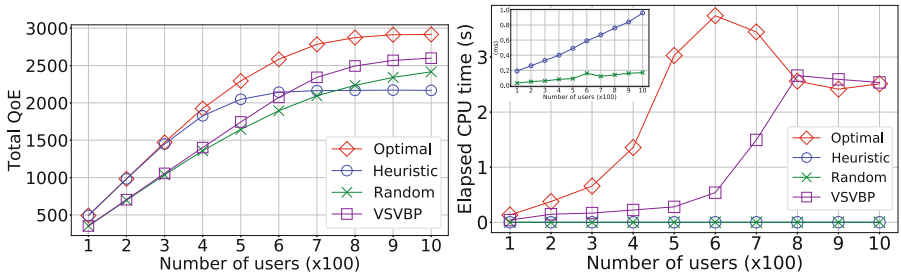
Our optimal approach and sub-optimal heuristic approach are compared to two other approaches, namely a random baseline, and a state-of-the-art approach for solving the EUA problem:

- *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 the server can achieve is W_2, the user will be randomly assigned either W_1 or W_2.
- *VSVBP*: [18] models the EUA problem as a variable sized vector bin packing (VSVBP) problem and proposes an approach that maximizes the number of allocated users while minimizing the number of edge servers needs to be used. Since VSVBP does not consider dynamic QoS, we randomly preset users' QoS levels, i.e., resource demands.

5.2 Experiment Settings

Our experiments were conducted on the widely-used EUA dataset [18], which includes data of base stations and end-users within the Melbourne central business district area in Australia. In order to simulate different dynamic QoS EUA scenarios, we vary the following three parameters:

- Number of end-users: We randomly select $100, 200, ..., 1,000$ users. Each experiment is repeated 100 times to obtain 100 different user distributions so that extreme cases, such as overly sparse or dense distributions, are neutralized.



(a) Total QoE (b) Elapsed CPU time

Fig. 3. Experiment set #1 results

- Number of edge servers: Say the users selected above are covered by m servers, we then assume $10\%, 20\%, ..., 100\%$ of those m servers are available to accommodate those users.
- Server's available resources: The server's available computing resources is generated following a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where $\sigma = 1$ and the average resource capacity of each server $\mu = 5, 10, 15, ...50$ in each dimension $d \in \mathcal{D}$.

Table 2 summarizes the settings of our three sets of experiments. The possible QoS level, for each user is preset to $\mathcal{W} = \{\langle 1, 2, 1, 2\rangle, \langle 2, 3, 3, 4\rangle, \langle 5, 7, 6, 6\rangle\}$. For the QoE model, we set $L = 5, \alpha = 1.5$, and $\beta = 2$. We employ two metrics to evaluate our approaches: (1) overall QoE achieved over all users for effectiveness evaluation, and (2) execution time (CPU time) for efficiency evaluation.

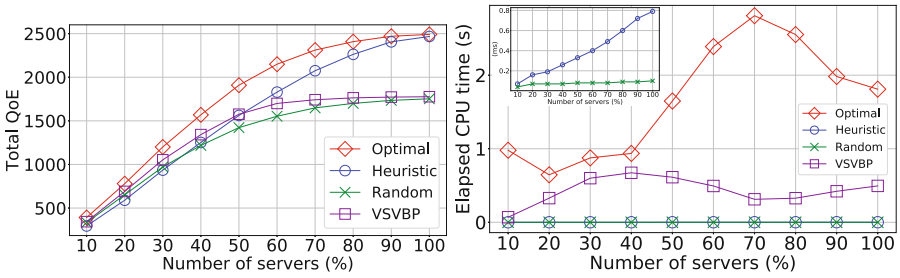
Table 2. Experiment Settings

	Number of users	Number of servers	Server's available resources
Set #1	$100, 200, ..., 1000$	70%	35
Set #2	500	$10\%, 20\%, ..., 100\%$	35
Set #3	500	70%	$5, 10, 15, ..., 50$

5.3 Experimental Results and Discussion

Figures 3, 4, and 5 depict the experimental results of three experiment sets 1, 2, and 3, respectively.

(1) Effectiveness: Figures 3, 4, and 5(a) demonstrate the effectiveness of all approaches in experiment sets 1, 2, and 3, measured by the overall QoE of all users in the experiment. In general, Optimal, being the optimal approach, obviously outperforms other approaches across all experiment sets and parameters. The performance of Heuristic largely depends on the computing resource availability, which will be analyzed in the following section.



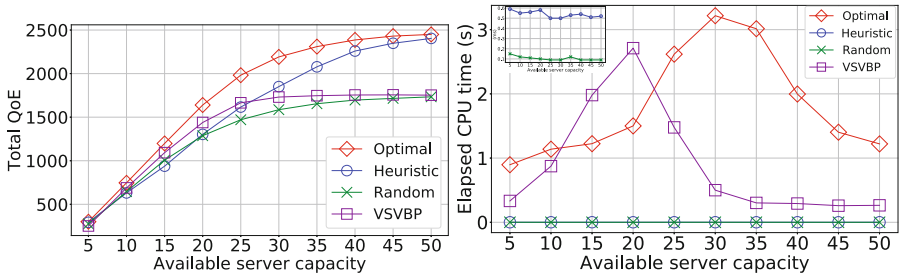
(a) Total QoE (b) Elapsed CPU time

Fig. 4. Experiment set #2 results

In experiment set 1 (Fig. 3(a)), we vary the number of users starting from 100 and ending at 1,000 in steps in 100 users. From 100 to 600 users, Heuristic results in higher total QoE than Random and VSVBP. Especially in the first three steps (100, 200, and 300 users), Heuristic achieves a QoE almost as high as Optimal. This occurs in those scenarios because the available resource is redundant and therefore almost all users receive the highest QoS level. However, as the number of users continues to increase while the amount of available resources is fixed, the computing resource for each user becomes more scarce, making Heuristic no longer suitable in these situations. In fact, from 700 users onwards, Heuristic starts being outperformed by Random and VSVBP. Due to being a greedy heuristic, Heuristic always tries to exhaust the edge servers' resources by allocating the highest possible QoS level to users, which is not an effective use of resource. For example, one user can achieve a QoE of 4.99 if assigned the highest QoS level W_3, which consumes a resource amount of $\langle 5, 7, 6, 6 \rangle$. That resource suffices to serve two users with QoS levels W_1 and W_2, resulting in an overall QoE of $1.6 + 4.09 = 5.69 > 4.99$. Since a user's QoS level is randomly assigned by Random and VSVBP, these two methods are able to user resource more effectively than Heuristic in those specific scenarios.

A similar trend can be observed in experiment sets 2 and 3. In resource-scarce situations, i.e. number of servers ranging from 10%–40% (Fig. 4(a)), and server's available resources ranging from 5–25 (Fig. 5(a)), Heuristic shows a nearly similar performance to Random and VSVBP (slightly worse in a few cases) for the same reason discussed previously. In those situations, the performance difference between Heuristic and Random/VSVBP is not as significant as seen in experiment set 1 (Fig. 3(a)). Nevertheless, the difference might be greater if the resources are more limited, e.g. 1,000 users in both experiment sets 2 and 3, an average server resource capacity of 20 in set 2, and 50% number of servers in set 3.

As discussed above, while being suitable for resource-redundant scenarios, Heuristic has not been proven to be superior when computing resources are limited. This calls for a more effective approach to solve the dynamic QoS problem under resource-scarce circumstances.



(a) Total QoE (b) Elapsed CPU time

Fig. 5. Experiment set #3 results

(2) Efficiency: Figures 3, 4, and 5(b) illustrate the efficiency of all approaches in the study, measured by the elapsed CPU time. The execution time of Optimal follows a similar pattern in all three experiment sets. As the experimental parameters increase from the starting point to a point somewhere in the middle – 600 users in set 1, 70% number of servers in set 2, and 30 average server resource capacity in set 3 – the time quickly increases until it reaches a cap of around a hefty 3 seconds due to being $\mathcal{NP}$-hard. The rationale for this is that the complexity of the problem increases as we keep adding up more users, servers, and available resource, generating more possible options and solutions for Optimal to select from. After passing that mid-point, the time gradually decreases at a slower rate then tends to converge. We notice that this convergence is a reflection of the convergence of the total QoE produced by Optimal in each corresponding experiment set. After the experimental parameters passing the point mentioned above, the available resource steadily becomes more redundant so that more users can obtain the highest QoS level without competing with each others, generating less possible options for Optimal, hence running faster.

In experiment sets 1 and 2, the execution time of Heuristic grows gradually up to just 1 milliseconds. However, it does not grow in experiment set 3 and instead stabilizes around 0.5–0.6 ms. This is because the available resource does not impact the complexity of Heuristic, which runs in $\mathcal{O}(nm \log m)$.

5.4 Threats to Validity

Threat to Construct Validity. The main threat to the construct validity lies in the bias in our experimental design. To minimize the potential bias, we conducted experiments with different changing parameters that would have direct impact on the experimental results, including the number of servers, the number of users, and available resources. The result of each experiment set is the average of 100 executions, each with a different user distribution, to eliminate the bias caused by special cases such as over-dense or over-sparse user distributions.

Threat to External Validity. A threat to the 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 that might be observed in different real-world scenarios.

Threat to Internal Validity. A threat to the 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 Sect. 5.3.

Threat to Conclusion Validity. The lack of statistical tests is the biggest threat to our conclusion validity. This has been compensated for by comprehensive 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

Cisco [3] coined the fog computing, or edge computing, paradigm in 2012 to overcome one major drawback of cloud computing – latency. Edge computing comes with many new unique characteristics, namely location awareness, wide-spread geographical distribution, mobility, substantial number of nodes, predominant role of wireless access, strong presence of streaming and real-time applications, and heterogeneity. Those characteristics allows edge computing to deliver a very broad range of new services and applications at the edge of 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 [13]. Su et al. [22] 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 is just a proxy for transferring tasks between mobile users and the cloud, whereas our edge server is where the tasks are processed. In addition, the price for using/hiring the broker/media cloud's resource seems to vary from time to time, broker to broker in their work. We target a scenario where there is no price difference within a single service provider. [11] investigates the cost - QoE trade-off in virtual machine provisioning problem in a centralized cloud, specific to video streaming domain. QoE is measured by the processing, playback, or downloading rate in those work.

QoE-focused architecture and resource allocation have started gaining attraction in edge computing area as well. [5] proposes a novel architecture that integrates resource-intensive computing with mobile application while leveraging mobile cloud computing. Their goal is to provide a new breed of personalized, QoE-aware services. [1,19] tackle the application placement in edge computing environments. They measure user's QoE based on three levels (low, medium, and

high) of access rate, required resources, and processing time. The problem we are addressing, user allocation, can be seen as the step after application placement. [14] focuses on computation offloading scheduling problem in mobile clouds from a networking perspective, where energy and latency must be considered in most cases. They propose a QoE-aware optimal and near-optimal scheduling scheme applied in time-slotted scenarios that takes into account the trade-off between user's mobile energy consumption and latency.

Apart from the aforementioned literature, there are a number of work on computation offloading or virtual machine placement problem. However, they do not consider QoE, which is important in an edge computing environment where human plays a prominent role. Here, we seek to provide an empirically grounded foundation for the dynamic QoS/QoE edge user allocation problem, forming a solid basis for further developments.

7 Conclusion

App users' quality-of-experience is of great importance for app vendors where user satisfaction is taken seriously. Despite being significant, there is very limited work considering this aspect in edge computing. Therefore, we have identified and formally formulated the dynamic QoS edge user allocation problem with the goal of maximizing users' overall QoE as the first step of tackling the QoE-aware user allocation problem. Having been proven to be $\mathcal{NP}$-hard and also experimentally illustrated, the optimal approach is not efficient once the problem scales up. We therefore proposed a heuristic approach for solving the problem more efficiently. We have also conducted extensive experiments on real-world dataset to evaluate the effectiveness and efficiency of the proposed approaches against a baseline approach and the state of the art.

Given this foundation of the problem, we have identified a number of possible directions for future work with respect to QoE such as dynamic QoS user allocation in resource-scarce or time-varying situations, user's mobility, service migration, service recommendation, just to name a few. In addition, a finer-grained QoE model with various types of costs or network conditions could be studied next.

Acknowledgments. This research is funded by Australian Research Council Discovery Projects (DP170101932 and DP18010021).

References

1. Aazam, M., St-Hilaire, M., Lung, C.H., Lambadaris, I.: Mefore: QoE based resource estimation at fog to enhance QoS in IoT. In: 2016 23rd International Conference on Telecommunications (ICT), pp. 1–5. IEEE (2016)
2. Alreshoodi, M., Woods, J.: Survey on QoE\QoS correlation models for multimedia services. arXiv preprint arXiv:1306.0221 (2013)

3. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, pp. 13–16. ACM (2012)
4. Cerwall, P., et al.: Ericsson Mobility Report. Ericsson, Stockholm (2018). https://www.ericsson.com/en/mobility-report/reports/november-2018
5. Chen, M., Zhang, Y., Li, Y., Mao, S., Leung, V.C.: EMC: emotion-aware mobile cloud computing in 5G. IEEE Netw. **29**(2), 32–38 (2015)
6. Chen, X.: Decentralized computation offloading game for mobile cloud computing. IEEE Trans. Parallel Distrib. Syst. **26**(4), 974–983 (2015)
7. Ding, B., Chen, L., Chen, D., Yuan, H.: Application of RTLS in warehouse management based on RFID and wi-fi. In: 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1–5. IEEE (2008)
8. Fiedler, M., Hossfeld, T., Tran-Gia, P.: A generic quantitative relationship between quality of experience and quality of service. IEEE Netw. **24**(2), 36–41 (2010)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability, vol. 29. wh freeman, New York (2002)
10. Hande, P., Zhang, S., Chiang, M.: Distributed rate allocation for inelastic flows. IEEE/ACM Trans. Netw. (TON) **15**(6), 1240–1253 (2007)
11. He, J., Wen, Y., Huang, J., Wu, D.: On the cost-QoE tradeoff for cloud-based video streaming under Amazon EC2's pricing models. IEEE Trans. Circuits Syst. Video Technol. **24**(4), 669–680 (2013)
12. Hemmati, M., McCormick, B., Shirmohammadi, S.: QoE-aware bandwidth allocation for video traffic using sigmoidal programming. IEEE MultiMedia **24**(4), 80–90 (2017)
13. Hobfeld, T., Schatz, R., Varela, M., Timmerer, C.: Challenges of QoE management for cloud applications. IEEE Commun. Mag. **50**(4), 28–36 (2012)
14. Hong, S.T., Kim, H.: QoE-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds. In: 2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 1–9. IEEE (2016)
15. Hoßfeld, T., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P., Schatz, R.: Quantification of YouTube QoE via crowdsourcing. In: 2011 IEEE International Symposium on Multimedia, pp. 494–499. IEEE (2011)
16. Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V.: Mobile edge computing–a key technology towards 5G. ETSI White Pap. **11**(11), 1–16 (2015)
17. Lachat, A., Gicquel, J.C., Fournier, J.: How perception of ultra-high definition is modified by viewing distance and screen size. In: Image Quality and System Performance XII, vol. 9396, p. 93960Y. International Society for Optics and Photonics (2015)
18. Lai, P., et al.: Optimal edge user allocation in edge computing with variable sized vector bin packing. In: Pahl, C., Vukovic, M., Yin, J., Yu, Q. (eds.) ICSOC 2018. LNCS, vol. 11236, pp. 230–245. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03596-9_15
19. Mahmud, R., Srirama, S.N., Ramamohanarao, K., Buyya, R.: Quality of experience(QoE)-aware placement of applications in fog computing environments. J. Parallel Distrib. Comput. (2018)

20. Shenker, S.: Fundamental design issues for the future internet. IEEE J. Sel. Areas Commun. **13**(7), 1176–1188 (1995)
21. Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., Heinzelman, W.: Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In: 2012 IEEE Symposium on Computers and Communications (ISCC), pp. 59–66. IEEE (2012)
22. Su, Z., Xu, Q., Fei, M., Dong, M.: Game theoretic resource allocation in media cloud with mobile social users. IEEE Trans. Multimedia **18**(8), 1650–1660 (2016)