Interference-aware SaaS User Allocation Game for Edge Computing

Guangming Cui, Qiang He, *Senior Member, IEEE,*
Xiaoyu Xia, Phu Lai, Feifei Chen, Tao Gu, and Yun Yang, *Senior Member, IEEE*

Abstract—Edge Computing, extending cloud computing, has emerged as a prospective computing paradigm. It allows a SaaS (Software-as-a-Service) vendor to allocate its users to nearby edge servers to minimize network latency and energy consumption on their devices. From the SaaS vendor's perspective, a cost-effective SaaS user allocation (SUA) aims to allocate maximum SaaS users on minimum edge servers. However, the allocation of excessive SaaS users to an edge server may result in severe interference and consequently impact SaaS users' data rates. In this paper, we formally model this problem and prove that finding the optimal solution to this problem is NP-hard. Thus, we propose ISUAGame, a game-theoretic approach that formulates the interference-aware SUA (ISUA) problem as a potential game. We analyze the game and show that it admits a Nash equilibrium. Then, we design a novel decentralized algorithm for finding a Nash equilibrium in the game as a solution to the ISUA problem. The performance of this algorithm is theoretically analyzed and experimentally evaluated. The results show that the ISUA problem can be solved effectively and efficiently.

Index Terms—SaaS user allocation, Interference, Data rate, Game theory, Edge computing, Nash Equilibrium, Potential game.

1 Introduction

Over the past decades, the prevalence of mobile devices, including mobile phones, wearable devices, tablets, etc., has increased significantly. Ericsson's Mobility Report [1] predicts that there will be around 32 billion mobile devices by 2023. It is expected that mobile devices will be the dominant computing devices for accessing many computation-intensive applications in the future [2], such as interactive gaming [3], face recognition [4], natural language processing [5]. Such complex applications are usually resource-hungry and demand intensive computation, high energy consumption and low latency. However, due to their physical size constraint, mobile devices are generally constrained by computation power and battery [6]. In the past several years, with the development of wireless communication technology, such as Wi-Fi and 4G, cloud computing has been employed as the main approach for tackling this issue. However, the cloud computing paradigm often cannot fulfill the stringent requirements of latency-sensitive applications due to the often unpredictable network latency and expensive bandwidth [7]. It is an evident weakness of the cloud computing paradigm because humans are acutely sensitive to delays which are very difficult to reduce at the wide area network scale [8].

In recent years, edge computing (EC), a key 5G enabler technology, has emerged as a new computing paradigm. It extends cloud computing by pushing resources from the cloud to the edge of the cloud. Computation tasks can be offloaded from mobile devices to edge servers endowed with cloud-like resources instead of the centralized cloud [8], [9], [10], [11], [12]. The EC paradigm paves the way for SaaS (software-as-a-service) applications that require low latency [10]. Extensive research has been conducted in the past few years on computation offloading in the EC environment [8], [10], [11], [12], [13], [14]. While a lot of existing research focus on latency reduction and/or energy savings from the network providers' or mobile devices' perspectives, this paper studies a critical problem from the SaaS vendor's perspective. The EC paradigm allows SaaS vendors such as Youtube and Uber to hire resources, e.g., bandwidth, cpu and memory, on edge servers from the network provider, e.g., AT&T, Telstra [15], [16]. Their SaaS applications can then be deployed on those edge servers to serve nearby users. Usually, an edge server covers a specific geographical area so that the SaaS users within its coverage can connect to it via wireless access [17]. Thus, edge servers are deployed in a distributed fashion - usually near cellular base stations [17], [18] - so that they can cover different geographical areas. The coverage areas of adjacent edge servers usually intersect to avoid blank areas that are not covered by any edge servers. A SaaS user located in the intersection area can connect to one of the nearby edge servers (i.e., *proximity constraint*) that has sufficient resources (i.e., *resource constraint*) such as bandwidth, cpu and memory. Given the number of SaaS users in a particular area, there are many ways to allocate them to the edge servers covering that area. An inappropriate allocation might result in a large number of unallocated SaaS users. Thus, from the SaaS vendor's perspective, a straightforward and important objective is to maximize the number of its SaaS users that can be allocated to edge servers in that area. Following the pay-as-you-go pricing model, the SaaS vendor pays for the resources on the edge servers hired from the network providers. Thus, the SaaS vendor's other important objective is to minimize the overall system cost for serving the SaaS users. This problem

- *G. Cui, Q. He, P. Lai and Y. Yang are with School of Software and Electrical Engineering, Swinburne University of Technology, Australia. E-mail: {gcui, qhe, tlai, yyang}@swin.edu.au.*
- *X. Xia and F. Chen are with School of Information Technology, Deakin University, Australia. E-mail: {xiaoyu.xia, feifei.chen}@deakin.edu.au.*
- *T. Gu is with the School of Science, RMIT University, Australia. E-mail: tao.gu@rmit.edu.au.*

Manuscript received April xx, 20xx; revised August xx, 20xx.

was first studied in [19] and is referred to as the *SaaS User Allocation* (SUA) problem in this paper.

A representative exemplar EC application is cloud gaming [20]. Most modern games are resource-hungry. By offloading computation to remote cloud servers, cloud gaming platforms, e.g., Google Stadia[1], Sony PlayStation Now[2] and Nvidia Shield[3], offer an energy-efficient gaming solution to thin-client devices like mobile phones and tablets [21]. For example, in a specific area shown in Fig. 1, there are four edge servers $s_1, ..., s_4$ that are capable of communicating with players' devices on multiple channels, and seven players $u_1, ..., u_7$. SaaS users outside the coverage area of an edge server cannot be allocated to that edge server due to the proximity constraint. For example, u_3 can be allocated to s_2, but not to s_1, s_3 or s_4. Furthermore, the SaaS vendor must consider the resource constraints, e.g., bandwidth, cpu, memory, etc. Compared with cloud servers powered by mega-scale data centers, edge servers are limited in their resources, which are usually shared by multiple SaaS vendors. Thus, an edge server must have adequate resources available to accommodate the SaaS users allocated to that edge server. In Fig. 1, the available resources of an edge server and players' resource needs, both unitized for simplicity, that is $\langle 1,1,1,1 \rangle$ for each SaaS user, are represented by vectors $\langle$bandwidth, cpu, memory, storage$\rangle$. According to [19], under the resource constraints, u_2, u_5, u_6 and u_7 cannot be all allocated to s_3 because their aggregate resource needs, i.e., $\langle 4, 4, 4, 4 \rangle$, exceed the available resources of s_3, i.e., $\langle 4, 3, 2, 4 \rangle$. While u_1, u_2 and u_5 can be allocated to edge server s_1, because $\langle 3, 3, 3, 3 \rangle \le \langle 3, 3, 4, 5 \rangle$. As discussed above, the SaaS vendor must pay for the hired resources based on a specific pricing model. Modeling resources in a unitized manner, the models and approach proposed in this paper are independent of specific pricing models. Given a SaaS's specific resource requirements and a specific pricing model, the cost of hiring resources on edge servers to serve SaaS users can be calculated. For example, assuming that an online game requires an Intel Xeon 1-core CPU and 4GB memory for each player, the corresponding Amazon EC2 instance(s) or Google Computing Engine instance(s) can be identified and the total cost for hiring the virtual machine instances can be calculated.

When multiple SaaS users wirelessly communicate with the same edge server at the same time, communication interference occurs, which lowers their data rates [3], [8], [22], [23]. As a result, their received data rates are usually lower than the theoretical results. For example, when u_1, u_2 and u_5 are allocated to s_1, the data rates actually received by each SaaS user might be lower than the minimum requirement. This essentially violates the resource constraints. The approach proposed in [19] attempts to allocate the most SaaS users to the least edge servers. Given the SUA problem presented in Fig. 1, the optimal solution found by the approach proposed in [19] is to allocate u_1, u_2 and u_5 to s_1, u_3 and u_7 to s_2, u_4 and u_6 to s_4. However, under the impact of wireless communication interference, 5 out of the 7 SaaS users, i.e., u_1 ,u_2, u_3 u_5 and u_7, cannot be allocated because their actual received width is lower than the minimum requirement. Thus, the SUA problem must be solved with the consideration of the wireless communication interference. It is referred to as the ISUA (interference-aware SUA) problem in this paper.

An edge server usually has several wireless channels to accommodate SaaS users. Thus, there are many solutions that fulfill all the proximity and resource constraints with wireless communication interference. Finding the optimal one that allocates the most SaaS users to the least edge servers is not trivial. The sharing of wireless channels further complicates this allocation problem because it impacts the server channel utilization and consequently the overall system cost under the pay-as-you-go pricing model.

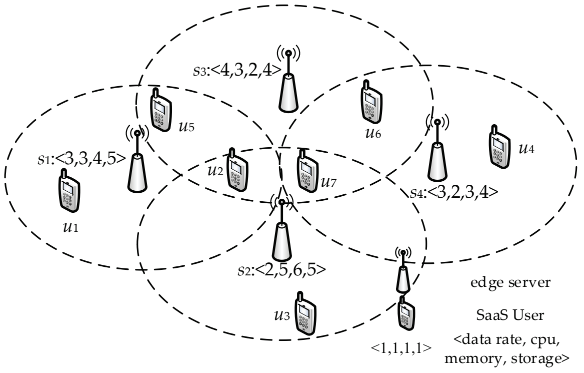


Fig. 1. An example ISUA scenario.

The SUA problem has been proven to be NP-hard [19]. The ISUA problem is further complicated by wireless communication interference. In real-world EC applications, the problem scale can be much larger than the example in Fig. 1 in terms of the number of SaaS users and/or edge servers. Thus, finding the optimal solution to a large-scale ISUA problem is intractable. SaaS vendors are in urgent need of an approach that can help them allocate their SaaS users effectively and efficiently.

In this paper, we introduce ISUAGame, a game-theoretic approach for finding a solution to the ISUA problem. Game theory has been widely used in the field of edge computing [3], [8]. It is a powerful tool for the design of decentralized mechanisms [24]. ISUAGame eases the burden of centralized optimization by modeling ISUA as a game where SaaS users' allocation decisions are made individually to achieve a collectively satisfactory solution. In addition, ISUAGame pursues SaaS users' individual (and often differentiated) interests by acting in their interest to find a nearby edge server that can serve each SaaS user. ISUAGame then employs a decentralized algorithm to achieve the Nash equilibrium of the game as the solution to the ISUA problem. The main contributions of this paper are as follows:

- We first model the ISUA problem as a constraint optimization problem and prove that it is NP-hard to find the centralized optimal solution.
- To solve the ISUA problem effectively and efficiently, we formulate it as a game which takes into account SaaS users' benefits, wireless communication interference and the overall system cost. The aim of the game is to maximize the number of allocated SaaS users and minimize the overall system cost while

1. https://www.stadia.com/
2. https://www.playstation.com/en-gb/explore/playstation-now/
3. https://www.nvidia.com/en-us/shield/

fulfilling all proximity constraints and resource constraints.

- We analyze the ISUA game and prove that it is a potential game that admits a Nash equilibrium.
- We propose a decentralized algorithm for finding the Nash equilibrium in the ISUA game, and analyze the performance of the algorithm theoretically and experimentally for effectiveness and efficiency.

The rest of this paper is organized as follows. We state the system model in Section 2. Section 3 formulates the ISUA game and presents the decentralized algorithm for finding the Nash equilibrium in the game. Section 4 evaluates the proposed decentralized algorithm theoretically and experimentally. Section 5 reviews the related work. Finally, Section 6 concludes this paper and points out the future work.

2 System Model

Given n SaaS users $U = \{u_1, ..., u_n\}$ and m edge servers $S = \{s_1, ..., s_m\}$ in a particular area, ISUA aims to find a cost-effective way to assign the SaaS users to appropriate wireless channels on the edge servers. Let us assume that there are k_j wireless channels for each edge server s_j, $j \in \{1, ..., m\}$, denoted as $C_j = \{c_j^1, ..., c_j^{k_j}\}$. Each SaaS user, with a set of computation resource requirements, can be allocated to an edge server that has adequate available resources to accommodate this SaaS user. The resource requirements of a SaaS user u_i, $i \in \{1, ..., n\}$, is denoted by $\omega_i = (\omega_i^d)$, where $d \in D =$\{bandwidth, cpu, memory, storage, ...\}. The available resources of an edge server s_j, $j \in \{1, ..., m\}$ is denoted as $\tau_j = (\tau_j^d)$, where $d \in D$. An *allocation decision* is made for each user u_i:

Definition 1 (Allocation Decision). An allocation decision for SaaS user u_i specifies that u_i is allocated to which communication channel of which edge server. It is denoted by $a_i \in \{(0,0) \bigcup (j,k)\}$, where $j \in \{1, ..., m\}$ and $k \in \{1, ..., k_j\}$ indicate that u_i is allocated to the kth wireless channel of s_j, i.e., c_j^k, or not to any edge servers, i.e., $a_i = (0,0)$.

Based on Definition 1, the allocation strategy for all SaaS users is defined as follows:

Definition 2 (Allocation Strategy). An allocation strategy is the combination of all SaaS users' allocation decisions, indicated by $\mathbf{a} = (a_1, ..., a_n)$.

Similar to many studies of edge computing, we investigate the ISUA problem in quasi-static scenarios where the edge servers' available resources remain unchanged during the allocation. More dynamic scenarios will be investigated in our future work. In this section, we introduce the wireless communication model, user benefit model, system cost model and optimization model. The notations adopted in the paper are summarized in Appendix A.

2.1 Wireless Communication Model

The allocation of excessive SaaS users to an edge server may lead to severe wireless communication interference. This can cause network congestion and reduce SaaS users' actual data rates [10]. Similar to [8], [10], [22], [23], we use $g_{i,j}^k$ to denote the up-link channel gain between SaaS user u_i and edge server s_j on its wireless channel c_j^k, which captures the effect of path-loss, shadowing and antenna gain [23]. SaaS user u_i's transmission power is denoted by p_i, which is determined by the wireless base station hosting the edge server according to some power control algorithms [8], in such as [25] and [26]. According to [22], [23], the transmission power of SaaS user u_i is the same when allocated to different edge servers because these edge servers are mostly provided by the same network provider. When multiple SaaS users can communicate with the same edge server on different wireless channels, the up-link inter-channel interference can be reduced significantly. However, these SaaS users still suffer from intra-channel interference [23]. Under these circumstances, the Signal-to-Interference-plus-Noise Ratio (SINR) for SaaS user u_i's wireless communication with edge server s_j on the kth wireless channel (c_j^k) can be calculated as follows [8], [22], [23]:

$$\gamma_{i,j}^k = \frac{p_i g_{i,j}^k}{\varpi_0 + \sum_{u_l \in U \backslash \{u_i\}: a_l = a_i} p_l \cdot g_{l,j}^k} \tag{1}$$

where ϖ_0 is the background noise variance of complex white Gaussian channel noise [22]. Since each SaaS user can be allocated to only one wireless channel, the actual achievable data rate for SaaS user u_i's communication with edge server s_j on channel c_j^k can be calculated as follows:

$$R_{i,j}^k = \mathcal{W}_j^k \cdot \log_2(1 + \gamma_{i,j}^k) \tag{2}$$

where $\mathcal{W}_j^k$ is the channel bandwidth of the kth wireless channel of edge server s_j. For a mobile network, there is a upper bound for data rate [27], [28]. Under this constraint, the maximum achievable data rate is denoted as R_{max}. Accordingly, the range of achievable data rate is $R_{i,j}^k \in [R_0, R_{max})$, where R_0 is the lowest data rate required for accessing the SaaS application on the edge server, which is domain-specifically determined by the SaaS vendor. In this research, we do not consider the physical differences between edge servers, e.g. their transmission power, antennas, etc. Thus, to simplify the discussion, we assume $\mathcal{W}_j^k$ and $g_{i,j}^k$ are the same for different edge servers, as they are both decided by the physical nature of the edge server and do not impact the ISUA game. In more general cases, $\mathcal{W}_j^k$ and $g_{i,j}^k$ can be different for different edge servers. The performance of ISUAGame in such cases will be experimentally evaluated in Section 4. SaaS user u_i's achievable data rate under wireless communication interference can then be calculated as follows:

$$R_{i,j}^k = \mathcal{W} \cdot \log_2(1 + \frac{g \cdot p_i}{\varpi_0 + \sum_{u_l \in U \backslash \{u_i\}: a_l = a_i} g \cdot p_l}) \tag{3}$$

2.2 User Benefit Model

As each edge server has a specific wireless coverage, a SaaS user u_i can be allocated to an edge server s_j only if it is covered by s_j:

$$u_i \in cov(s_j), \text{ for } u_i \in U, \ s_j \in S \tag{4}$$

where $cov(s_j)$ is the wireless coverage of s_j. If u_i is covered by s_j, s_j is referred to as u_i's *nearby edge server*. The set of u_i's nearby edge servers is denoted by $N(u_i)$.

A SaaS user can obtain benefits from the ISUA by saving its device's resources in multiple dimensions, such as data rate, cpu, memory, storage, etc. Based on the wireless communication model, given an allocation strategy $\mathbf{a} = \{a_1, ..., a_n\}$, the overall benefit of u_i with $a_i \in \mathbf{a}$ is calculated by aggregating its savings in those dimensions:

$$B_{\mathbf{a}}(a_i) = \begin{cases} \lambda_i^0 R_{i,s_{a_i}}^{c_{a_i}} + \sum_{d \in D'} \lambda_i^d \omega_i^d, & a_i \neq (0,0) \\ 0, & a_i = (0,0) \end{cases} \quad (5)$$

where $D' = D \backslash \{\text{data rate}\}$ represent the set of all resource dimensions except for data rate. When $a_i \neq (0,0)$, $a_i = (j,k)$ indicates that u_i is allocated to s_j on channel c_j^k. Here, we use s_{a_i} to represent edge server s_j in allocation decision $a_i = (j,k)$, and c_{a_i} to represent channel c_j^k on s_j in $a_i = (j,k)$. λ_i^d (λ_i^0 for data rate) is the weight that indicates u_i's priority for the dth resource dimension, as SaaS users might have different priorities for savings in those resource dimensions. For instance, a SaaS user that favors high-quality video streaming usually has a higher priority for data rate over other resources. Weight λ_i^d can be specified by u_i or calculated based on the differences between u_i's resource needs and the resources available on u_i's device. For some applications, it is possible that SaaS users have different resource needs. For example, a cloud game player that demands higher resolution and frame rate in the game requires a higher data rate, more cpus and more memory. However, it is not the case for most other applications, e.g., face recognition and natural language processing. The users of such an application usually have the same resource needs. Thus, for those applications, there is $\omega_i^d = \omega_{i'}^d, \forall u_i, u_{i'} \in U$.

2.3 System Cost Model

As discussed in Section 1, one of the SaaS vendor's optimization objectives is to minimize the overall cost incurred by hiring the resources on edge servers for serving the SaaS users. Given an allocation decision a_i for SaaS user u_i, $i \in \{1, ..., n\}$ and an allocation strategy $\mathbf{a} = \{a_1, ..., a_n\}$, based on the wireless communication model, the system cost incurred by a_i is calculated as follows:

$$Z_{\mathbf{a}}(a_i) = \begin{cases} \lambda_i^0 R_{i,s_{a_i}}^{c_{a_i}} + \sum_{d \in D'} \lambda_i^d \omega_i^d, & a_i \neq (0,0) \\ \beta(\lambda_i^0 R_{max} + \sum_{d \in D'} \lambda_i^d \omega_i^d), & a_i = (0,0) \end{cases} \quad (6)$$

where R_{max} is the theoretical maximum data rate that can achieved by u_i on an edge server and $\beta \in [1, +\infty)$ is the weight that indicates the SaaS vendors' priority for its optimization objective to maximize the number of allocated SaaS users. For example, a SaaS vendor that is keen to serve as many SaaS users as possible at a high cost can specify a large β. As discussed in Section 2.2, for a SaaS vendor's any two different SaaS users u_i and $u_{i'}$, $(\forall u_i, u_{i'} \in U)$, there is $\omega_i^d = \omega_{i'}^d, \forall d \in D$. As a result, the system cost incurred in resource dimensions D', denoted by $\delta = \sum_{d \in D'} \lambda_i^d \omega_i^d$, is a constant and (6) can be converted to:

$$Z_{\mathbf{a}}(a_i) = \begin{cases} \lambda_i^0 R_{i,s_{a_i}}^{c_{a_i}} + \delta, & a_i \neq (0,0) \\ \beta(\lambda_i^0 R_{max} + \delta), & a_i = (0,0) \end{cases} \quad (7)$$

Although δ must be calculated into the overall system cost in real-world applications, it is not impacted by the changes in a_i. It does not impact the update of a_i. Thus, in the following discussion, we ignore δ and focus on the system cost incurred in the data rate dimension. Now, (7) can be re-formulated as follows:

$$Z_{\mathbf{a}}(a_i) = \begin{cases} \lambda_i^0 R_{i,s_{a_i}}^{c_{a_i}}, & a_i \neq (0,0) \\ \beta \lambda_i^0 R_{max}, & a_i = (0,0) \end{cases} \quad (8)$$

As discussed in Section 1, the SaaS vendor has two optimization objectives for ISUA. One is to maximize the number of allocated SaaS users, and the other is to minimize the system cost for accommodating those SaaS users. SaaS users not allocated to any edge servers (i.e., $a_i = (0,0)$) will not be able to access the application on edge servers. This incurs losses for the SaaS vendor and thus is included as part of the system cost. Without this part, the minimization of the overall system cost will be equivalent to the minimization of only the cost for hiring the resources on edge servers. Driven by this objective, the SaaS vendor will allocate none of its SaaS users to any of the edge servers because it will minimize the overall system cost to zero. Thus, the overall system cost must include both the cost for hiring the resources on edge servers and the cost for not being able to serve the unallocated users on edge servers. Accordingly, given an allocation strategy $\mathbf{a} = \{a_1, ..., a_n\}$, the SaaS vendor's objective to minimize the overall system cost is expressed as follows:

$$Z_{\mathbf{a}} = \min \sum_{u_i \in U} Z_{\mathbf{a}}(a_i) \quad (9)$$

2.4 Optimization Model

The ISUA problem can be modeled as a constrained optimization problem (COP) that consists of a finite set of variables $X = \{x_1, ..., x_n\}$, with a domain D_i $(i = 1, ..., n)$ listing the possible values for each variable x_i in X, and a set of constraints τ over X. A solution to a COP is an assignment of a value to each variable x_i in X from D_i such that all constraints in τ are fulfilled. Given a set of SaaS users $U = \{u_1, ..., u_n\}$ and a set of edge servers $S = \{s_1, ..., s_m\}$, the COP model for an ISUA problem is formally expressed as follows:

$$\max I_{\{a_i \neq (0,0)\}} \quad (10)$$

$$\min \sum_{u_i \in U} Z_{\mathbf{a}}(a_i) \quad (11)$$

subject to:

$$a_i \in \{0,0\} \cup \{j | u_i \in cov(s_j)\}, \forall u_i \in U \quad (12)$$

$$\begin{cases} \sum_{u_i \in U : a_i = (j,k)} \omega_i^d \leq \tau_j^d, \forall d \in D' \\ \sum_{u_i \in U : a_i = (j,k)} R_{i,j}^k \leq \tau_j^d, d = data\ rate \end{cases} \quad (13)$$

where $I_{\{condition\}}$ is an indicator function that returns 1 if *condition* is true, and 0 otherwise. Objective (10) maximizes the number of allocated SaaS users. It calculates the number of edge servers needed. Objective (11) minimizes the overall system cost. The coverage constraints (12) ensure that every SaaS user $u_i \in U$ can be allocated to its nearby edge server. Resource constraints (13) ensure that the total requirements of the SaaS users allocated to an edge server s_j for the kth resource must not exceed the corresponding resource available on s_j. The solution to this problem is an allocation strategy $\mathbf{a} = (a_1, a_2, ..., a_n)$ that achieves objectives (10) and

(11), and in the meantime fulfills constraints (12) and (13). How to trade off both optimization objectives is specific to SaaS vendors. For example, objective (11) can be prioritized over objective (10) with the Lexicographic Goal Programming technique for a SaaS vendor with a low budget.

The ISUA problem is in fact the generalization of the classic bin packing (BP) problem [19]. In a classic BP problem, we are given an infinite number of bins $S = \{s_1, ..., s_m\}$ with a resource $\tau \in \mathbb{N}$, a set of n items $U = \{u_1, ..., u_n\}$ sized w_i ($0 < w_i \leq \tau$). The objective is to pack all the items in U into the fewest bins possible such that the total size of the items in each bin must not exceed the size of the bin: $\sum_{u_i \in U(s_j)} \omega_i \leq \tau, \forall s_j \in S$, where $U(s_j)$ is the set of items in $s_j \in S$. In the ISUA problem, edge servers are regarded as bins with different capacities. This is because edge servers might have different hardware specifications and host different applications for different numbers of users. SaaS users are regarded as items. Edge servers' available resources and SaaS users' resource needs are usually multi-dimensional, e.g., bandwidth, cpu, memory and storage etc., which are represented as a d-dimensional vector. Thus, the ISUA problem can be modeled as a variable size vector BP problem. The classic BP problem is a special case of the variable size vector BP problem with $d = 1$ and the same size for all the bins. Because the classic BP problem is NP-hard [29], the variable size vector BP problem is NP-hard. Thus, when the scale of the ISUA problem is large in terms of the number of SaaS users and/or the number of edge servers, finding the optimal solution is intractable. The SaaS vendor needs an efficient approach for finding an effective solution to the large-scale ISUA problem.

3 Game Formulation and Algorithm Mechanism

To solve the ISUA problem, this section presents ISUAGame, our game-theoretic approach for solving the ISUA problem. There are three main reasons for the adoption of a game-theoretic approach. First, SaaS users may have differentiated requirements for resources. Game theory has been successfully employed in many fields as a powerful tool for analyzing the interactions among multiple players pursuing their individual interests. Second, in the EC environment, it can be employed to devise incentive compatible mechanisms for finding collectively satisfactory solutions to the ISUA problem such that no SaaS users have the incentive to deviate unilaterally. Third, by making local allocation decisions for SaaS users individually, ISUAGame solves the ISUA problem in a decentralized manner. It can lift the burden of finding the centralized optimal solution. In this way, it scales with the size of the ISUA problem. The solutions to large-scale ISUA problems can be found quickly. This caters to SaaS users' as well as SaaS vendors' needs for low latency in the EC environment.

3.1 Game Formulation and Property

In the ISUA game, SaaS users, as players, make the decisions on which channel of which edge servers they are allocated to, producing $a_i \in \{(0,0)\} \bigcup \{(j,k) | \forall u_i \in cov(s_j), k \in C_j = \{c_j^1, ..., c_j^{k_j}\}\}$ for each u_i. Let $\mathbf{a}_{-i} = (a_1, ..., a_{i-1}, a_{i+1}, ..., a_n)$ denote all SaaS users' allocation decisions except u_i. Given other SaaS users' allocation decisions $\mathbf{a}_{-i}$, u_i would like to make an allocation decision a_i to maximize its benefit in terms of its savings on multi-dimensional resources:

$$\max_{a_i \in \{(0,0)\} \bigcup \{(j,k) | s_j \in N(u_i), c_j^k \in C_j\}} B_{\mathbf{a}}(a_i) \quad (14)$$

Base on (14), the ISUA problem can be formulated as a game $\chi = (U, \{\mathcal{A}_i\}_{u_i \in U}, \{\mathcal{B}_i\}_{u_i \in U})$, where U is the set of SaaS users, $\mathcal{A}_i$ is u_i's finite set of allocation decisions and $\mathcal{B}_i$ is the function that calculates the benefit produced by $a_i \in \mathcal{A}_i$. In this game, there might be conflicts among SaaS users. For example, the allocation of some SaaS users to a particular edge server might prevent some other SaaS users from being allocated to the same edge server or any edge servers. In Fig. 1, if u_2 and u_7 are allocated to s_2, u_3 cannot be allocated to any edge servers. If u_2 is willing to select s_1 or s_3, and u_7 is willing to select s_3 or s_4, u_3 can be allocated to s_2. However, the decisions of u_2 and u_7 might result in conflicts with other SaaS users. To study how such conflicts can be resolved, we investigate whether the game admits at least one Nash equilibrium [30]:

Definition 3 (Nash Equilibrium). An allocation strategy $\mathbf{a}^* = (a_1^*, a_2^*, ..., a_n^*)$ is a Nash equilibrium if no SaaS user can further increase its individual benefit by unilaterally changing its allocation decision, i.e.,

$$B_{\mathbf{a}^*_{-i}}(a_i^*) \geq B_{\mathbf{a}^*_{-i}}(a_i), \forall a_i \in \mathcal{A}_i, u_i \in U \quad (15)$$

Whether the ISUA game admits at least one Nash equilibrium is of significant importance in this research. In a Nash equilibrium, every SaaS user's allocation decision is its best response to the choices of the $n-1$ other SaaS users.

Property 1. In a Nash equilibrium $\mathbf{a}^*$ of the ISUA game, the allocation decision a_i^* of each user u_i ($u_i \in U$) must be the best choice in $\mathcal{A}_i$ in response to $\mathbf{a}_{-n}$.

Property 1 ensures that, if a Nash equilibrium can be achieved, ISUA game can use it as a self-enforcing solution to the ISUA problem. A self-enforcing solution, once agreed upon by all the SaaS users, does not need centralized enforcement, because it is in every SaaS user's self-interest to stick to the agreement [31]. Based on Property 1, we investigate the existence of at least one Nash equilibrium in the ISUA game. **The key is to prove that the ISUA game is a potential game** [32], which is defined as follows:

Definition 4 (Potential Game). A game is a potential game if, for a potential function $\phi(\mathbf{a})$, there is

$$B_{\mathbf{a}_{-i}}(a_i) < B_{\mathbf{a}_{-i}}(a_i') \Rightarrow \phi(a_i, \mathbf{a}_{-i}) < \phi(a_i', \mathbf{a}_{-i}) \quad (16)$$

for any $u_i \in U$, $a_i, a_i' \in \mathcal{A}_i$ and $\mathbf{a}_{-i} \in \prod_{l \neq i} \mathcal{A}_l$.

Based on Definition 3, the Nash equilibrium in an ISUA game can be interpreted in a similar way [33]. A decision strategy $\mathbf{a}^*$ is a Nash equilibrium if, for all SaaS users $u_i \in U$, there is $B_{\mathbf{a}^*_{-i}}(a_i^*) = \max_{a_i \in \mathcal{A}_i} B_{\mathbf{a}^*_{-i}}(a_i)$. Thus, in a potential game, there is always at least one Nash equilibrium which can be found by finding the local optima of the potential function, as proven by Marden et al. in [34]. This property of potential games has also be leveraged by Chen et al. in [3], [8].

To prove that the ISUA game is a potential game, we first introduce and prove a property of the formulated ISUA game.

Lemma 1. Given an allocation strategy $\mathbf{a} = \{a_1, ..., a_n\}$, a SaaS user u_i can be allocated to channel c_j^k if its received wireless communication interference $\mu_{i,j}^k(\mathbf{a}) \triangleq \sum_{u_l \in U \setminus \{u_i\}: a_l = a_i} g \cdot p_l$ fulfills $\mu_{i,j}^k(\mathbf{a}) \leq T_i$, with the threshold

$$T_i = \frac{g \cdot p_i}{2^{\frac{R_0}{W}} - 1} - \varpi_0$$

and $\mu_{i,j}^{k,d}(\mathbf{a}) \triangleq \sum_{u_l \in U \setminus \{u_i\}: a_l = a_i} \omega_l$ for $d \in D'$ satisfies $\mu_{i,j}^{k,d}(\mathbf{a}) \leq T_i^d$, with the threshold

$$T_i^d = \tau_{a_i}^d - \omega_i^d$$

The proof of Lemma 1 can be found in Appendix B. As discussed in Section 2.3, there is $\omega_i^d = \omega_{i'}^d$ and $\tau_{a_i}^d$, $d \in D'$, is the available resources of edge server s_{a_i}, T_i^d $(i = 1, ..., n)$ are usually the same for SaaS users allocated to the same edge server. In this way, when a SaaS user u_i attempts to be allocated to a particular edge server, the corresponding T_i^d can be calculated and verified easily. Thus, in the following analysis of the potential game, we mainly focus on the wireless interference dimension in D. According to Lemma 1, when SaaS user u_i's received interference $\mu_i(\mathbf{a})$ on channel c_j^k is adequately low, it is beneficial for u_i to be allocated to edge server s_j. Otherwise, u_i is not allocated to any edge servers. Based on Lemma 1, with the potential function below, the ISUA game is indeed a potential game.

$$\begin{aligned}\phi(a_i, \mathbf{a}_{-i}) = &-\frac{1}{2} \sum_{u_i \in U} \sum_{u_l \neq u_i} g p_i \cdot g p_l I_{\{a_i = a_l\}} I_{\{a_i \neq (0,0)\}} \\ &- \sum_{u_i \in U} g p_i T_i I_{\{a_i = (0,0)\}}\end{aligned} \tag{17}$$

Now, we prove that the ISUA game is a potential game with Eq. (17) as the potential function.

Theorem 1 (Potential ISUA Game). The ISUA game is a potential game with $\phi(a_i, \mathbf{a}_{-i})$ as the potential function.

The proof of Theorem 1 can be found in Appendix C.

A potential game admits at least one Nash equilibrium [34]. An important property of potential games is the *Finite Improvement Property*. This property indicates that a Nash equilibrium of a potential game can be reached via a process with a finite number of iterations [32]. The Finite Improvement Property ensures that this process will eventually complete and find a Nash equilibrium. Based on this property, we design a decentralized allocation mechanism for finding the Nash equilibrium in the ISUA game, which will be presented and analyzed in Section 3.2 next.

3.2 Mechanism Design and Convergence Analysis

Given $U = \{u_1, ., ., , u_n\}$ and $S = \{s_1, ..., s_m\}$, ISUAGame employs an iterative process to find a decision strategy that reaches the Nash equilibrium as the solution to the ISUA problem. Algorithm 1 presents the pseudo code. This process starts with an initial allocation strategy (Lines 1-3). Then, based on Eq. (9), the system cost on each edge server $s_j \in S$ incurred by decision strategy $\mathbf{a}$ in the current iteration t $(t = 1, 2...)$, denoted by $\mathbf{a}(t)$, is calculated with Eq. (18) (Line 5):

$$\rho_{\mathbf{a}(t)}(s_j) \triangleq \sum_{u_i \in U: s_{a_i} = s_j} \lambda_i^0 R_{i,j}^k \tag{18}$$

Next, leveraging the Finite Improvement Property, one SaaS user u_i that can improve its benefit updates its current allocation decision a_i to a better one, denoted by a_i', in each iteration. Specifically, each user $u_i \in U$ first calculates the updated system cost on each $s_j \in N(u_i)$ incurred by $\mathbf{a'}(t)$ updated from $\mathbf{a}(t)$ with the change from a_i to a_i' (Lines 7-9). There are three cases, as indicated by Eq. (19).

$$\rho_{\mathbf{a'}(t)}(s_j) = \begin{cases} \rho_{\mathbf{a}(t)}(s_j) - \lambda_i^0 R_{i,s_{a_i}}^{c_{a_i}}, & for\ s_j = s_{a_i} \\ \rho_{\mathbf{a}(t)}(s_j) + \lambda_i^0 R_{i,s_{a_i'}}^{c_{a_i'}}, & for\ s_j = s_{a_i'} \\ \rho_{\mathbf{a}(t)}(s_j), & otherwise \end{cases} \tag{19}$$

Next, u_i finds its optimal allocation decision a_i' that incurs the lowest total system cost across $N(u_i)$. If $a_i' \neq a_i$, it sends a_i' to the other SaaS users requesting to contend for the decision update opportunity (Line 10). If there are multiple such allocation decisions, it randomly selects one to be sent. In each iteration, one SaaS user is randomly selected as the winner and its allocation decision is updated. The winner can be selected in a centralized or a decentralized manner. The former requires centralized control [8] and the latter requires messaging synchronizing [3]. In either way, the calculation in each iteration of the process (Lines 5-17) is performed by individual SaaS users in parallel. The SaaS users who did not win the contest do not update their allocation decisions in the current iteration. This process iterates until no SaaS users request to update their allocation decisions. The allocation decisions for all the SaaS users constitute the final decision strategy as the solution to the ISUA problem.

Algorithm 1 Decentralized ISUA Allocation

```
1:  Initialization:
2:  each $u_i$ chooses the allocation decision $a_i = (0,0)$
3:  End of initialization
4:  **repeat**
5:      compute the system cost on $s_j \in S$ incurred by $\mathbf{a}(t)$: $\rho_{\mathbf{a}(t)}(s_j)$
6:      **for** each $u_i \in U$ **do**
7:          **for** each server $s_j \in N(u_i)$ **do**
8:              calculate the system cost $\rho_{\mathbf{a'}(t)}(s_j)$ under $\mathbf{a'}(t)$ updated from $\mathbf{a}(t)$ with $a_i' = j$
9:          **end for**
10:         find the allocation decision $a_i'$ that incurs the lowest total system cost across $N(u_i)$
11:         **if** $a_i' \neq a_i$ **then**
12:             send $a_i'$ to contend for decision update opportunity
13:             **if** wins the opportunity **then**
14:                 update its allocation decision with $a_i'$
15:             **end if**
16:         **end if**
17:     **end for**
18: **until** no more needs for decision
```

The Finite Improvement Property of the potential game ensures that, after a finite number of iterations, the allocation process will complete and reach a Nash equilibrium. Let F be the total number of iterations, $Q_i \triangleq gp_i$, $Q_{min} \triangleq \min(Q_i)$, $Q_{max} \triangleq \max(Q_i)$, $T_{min} \triangleq \min T_i$, $T_{max} \triangleq \max(T_i)$ $(i = 1, ..., n, j = 1, ..., m$ and $k = 1, ..., c_j)$, Theorem 2 holds for the quantification of F.

Theorem 2. When T_i and Q_i are non-negative integers for any $u_i \in U$, the maximum convergence time of ISUAGame, measured by the maximum number of iterations, is $n^2 Q_{max}^2 / 2Q_{min} + nQ_{max} \cdot T_{min}/Q_{min}$, that is, $F \leq n^2 Q_{max}^2 / 2Q_{min} + nQ_{max} \cdot T_{min}/Q_{min}$.

The proof of Theorem 2 can be found in Appendix D. Theorem 2 shows that ISUAGame converges within a quadratic time. Here, we consider the case where Q_i and T_i are non-negative integers for ease of exposition. For a more general case where Q_i and T_i can be real numbers, the experimental results demonstrated in Section 4.2 show that ISUAGame converges rapidly and its convergence time scales with the number of SaaS users (n) and the number of edge servers (m).

4 Performance Evaluation

In this section, we first theoretically and then experimentally evaluate the performance of ISUAGame in achieving the SaaS vendor's two optimization objectives.

4.1 Theoretical Analysis

As discussed in Section 3.2, the SaaS users make their allocation decisions in parallel in each iteration of the allocation process. The winner of the contest for the decision update opportunity in each iteration may be determined in a non-deterministic manner, e.g., via random selection. This leads to the possibility of multiple Nash equilibria in the ISUA game. Thus, the performance of ISUAGame is dependent of the Price of Anarchy (PoA) of the decentralized ISUA allocation mechanism, which is measured by the ratio between the utility of the worst Nash equilibrium and the centralized optimal solution [35]. In the ISUA game, the utility of a Nash equilibrium is measured by the number of allocated SaaS users and the overall system cost.

4.1.1 PoA in the Number of Allocated SaaS users

Let us denote the set of decision strategies that reach different Nash equilibria in the ISUA game with χ and the centralized optimal decision strategy with $\mathbf{a}^* = (a_1^*, a_2^*, ..., a_n^*)$. Given a decision strategy $\mathbf{a} \in \chi$, let $poa_{user}(\mathbf{a})$ be the PoA measured by the ratio between the number of allocated SaaS users with $\mathbf{a}$ and that with $\mathbf{a}^*$, $poa_{user}(\mathbf{a})$ is calculated as follows:

$$poa_{user}(\mathbf{a}) = \frac{\min\limits_{\mathbf{a}\in\chi} \sum\limits_{s_j\in S} num_{s_j}(\mathbf{a})}{\sum\limits_{s_j\in S} num_{s_j}(\mathbf{a}^*)} = \frac{\min\limits_{\mathbf{a}\in\chi} \sum\limits_{u_i\in U} I_{\{a_i>0\}}}{\sum\limits_{u_i\in U} I_{\{a_i^*>0\}}} \quad (20)$$

where $num_{s_j}(\mathbf{a})$ and $num_{s_j}(\mathbf{a}^*)$ are the numbers of allocated SaaS users with $\mathbf{a}$ and $\mathbf{a}^*$ respectively.

Theorem 3 (PoA in the Number of Allocated SaaS users). Given any decision strategy $\mathbf{a} \in \chi$ that achieves a Nash equilibrium in the ISUA game and the centralized optimal decision strategy $\mathbf{a}^*$, the PoA of ISUAGame, calculated with Eq. (20) fulfills:

$$1 \geq poa_{user}(\mathbf{a}) \geq \frac{\lfloor T_{min}/Q_{max} \rfloor}{\lfloor T_{max}/Q_{min} \rfloor + 1} \quad (21)$$

The proof of Theorem 3 can be found in Appendix E.

4.1.2 PoA in Overall System Cost

Given a decision strategy $\mathbf{a} \in \chi$, let $poa_{cost}(\mathbf{a})$ be the PoA measured by the ratio between the overall system cost incurred by $\mathbf{a}$ and that by $\mathbf{a}^*$ and $poa_{cost}(\mathbf{a})$ can be calculated as follows:

$$poa_{cost}(\mathrm{a}) = \frac{\min\limits_{\mathbf{a}\in\chi} \sum\limits_{u_i\in U} Z_i(\mathbf{a})}{\sum\limits_{u_i\in U} Z_i(\mathbf{a}^*)} \quad (22)$$

As discussed in Section 3.2, there are two components in the overall system cost $Z_{\mathbf{a}}$, incurred by hiring resources on edge servers and failures to allocate SaaS users to any edge servers. Let us use $Z_i^{edge}(\mathbf{a})$ and $Z_i^{local}(\mathbf{a})$ to represent the former and the latter respectively. Given Q_i, Q_{max} and Q_{min} defined in Section 3.2, let us define $Z_i^{edge}(\mathbf{a}) \triangleq \lambda_i^0 R_{i,j}^k$, $Z_{i,min}^{edge}(\mathbf{a}) \triangleq \lambda_i^0 R_0$ and $Z_{i,max}^{edge}(\mathbf{a}) \triangleq \lambda_i^0 R_{max}$, where $Q_{min} \leq Q_i \leq Q_{max}$, and $Z_i^{local}(\mathbf{a}) \triangleq \lambda_i^0 R_{max}$. Based on these definitions, we have Theorem 4.

Theorem 4 (PoA in Overall System Cost). Given a decision strategy $\mathbf{a} \in \chi$ that achieves a Nash equilibrium in the ISUA game and the centralized optimal decision strategy $\mathbf{a}^*$, the PoA of ISUAGame fulfills:

$$\begin{aligned} 1 &\leq poa_{cost}(\mathbf{a}) \\ &\leq \frac{\sum\limits_{u_i\in U} \lambda_i^0 R_{max}}{\sum\limits_{u_i\in U} \lambda_i^0 R_{max} \cdot I_{\{a_i=(0,0)\}} + \lambda_i^0 R_0 \cdot I_{\{a_i\neq(0,0)\}})} \end{aligned} \quad (23)$$

The proof of Theorem 4 can be found in Appendix F.

4.2 Experimental Evaluation

In this section, we evaluate the performance of ISUAGame with a set of small-scale experiments and a set of large-scale ones. In the first set of experiments (set #1), we compare the effectiveness of ISUAGame in achieving the SaaS vendor's two optimization objectives, i.e., Eqs. (10) and (11), measured by the number of allocated SaaS users ($I \triangleq I_{\{a_i\neq(0,0)\}}$) and the overall system cost ($Z \triangleq Z_{\mathbf{a}}$), against the *Optimal* approach, and two baseline approaches originated from [19], i.e., *Random* and *Greedy*. Given a set of SaaS users and a set of edge servers, *Random* randomly allocates each SaaS user to one of its nearby edge servers that has adequate available resources and *Greedy* allocates a SaaS user to its nearby edge server with the most available resources. The Optimal approach solves the optimization model presented in Section 2.4 with IBM's CPLEX Optimizer v12.2 and allocate SaaS users based on the solution. In the second set of experiments (set #2), we compare ISUAGame against only *Random* and *Greedy*, because the scales of the ISUA problems in set #2 are too large for Optimal to find a solution within a reasonable amount of time. To evaluate the efficiency of ISUAGame, we present and discuss its time consumption, i.e., the time taken to find a solution. We also analyze its convergence time, i.e., the number of iterations taken to reach a Nash equilibrium, which is an important metric for evaluating the efficiency of game-theoretical approaches [3], [8], [36]. In each iteration of ISUAGame, the decisions are made for individual SaaS users simultaneously. Thus, the one that takes the most time is calculated into the time consumption in that iteration. All the experiments are conducted on a Windows machine equipped with Intel Core i5-7400T processor (4 cpus, 2.4GHz) and 8GB RAM.

Experimental Data. The experiments are conducted on the locations of real-world base stations and SaaS users within Metropolitan Melbourne in Australia, which has a total area of over 9,000 km^2. Australian Communications and Media Authority (ACMA) publishes the radio-comms license dataset that contains the geographical location of all cellular base stations in Australia, which are used as the locations of edge servers because edge servers are usually deployed at base stations [17]. The Asia Pacific Network Information Centre (APNIC) provides all IP address blocks allocated to Australia. We use an IP lookup service[4] to convert the obtained IP addresses into geographical locations to simulate SaaS users' locations. Since IP addresses in the last octet often have identical geographical addresses returned by the IP lookup service, SaaS users are uniformly distributed around each of the obtained geographical locations. The coverage of each edge server is randomly set based on the density of SaaS users within its coverage areas. In areas with high, medium and low user densities, the coverage radius of an edge server is set between 450 and 750 meters, 2,000 and 3,000 meters, 7,000 and 8,000 meters respectively. The dataset used in the experiments is publicly available[5] for the reproduction of our experimental results.

Experimental Settings. To comprehensively evaluate ISUAGame, we have simulated various ISUA scenarios by changing three parameters in the experiments: 1) the number of SaaS users; 2) the number of edge servers; and 3) the available resources on edge servers. The available resources on each edge server are randomly generated following normal distributions. The details are shown in Table 1, where the last column indicates the average resource in each dimension on each edge server. Each edge server is simulated in the same way as [3], [8], [37], with $C_j = 5, s_j \in S$, channel bandwidth $\mathcal{W} = 5$MHz, transmission power $p = 1000$mWatts and the background noise $\varpi_0 = -100$dBm. Based on the wireless communication model for urban cellular radio environment [37], which is also employed in [8], we set the channel gain $g_{i,j} = l_{i,j}^{\alpha}$, where $l_{i,j}$ is the distance between SaaS user u_i and the edge server s_j, and $\alpha = 5$ is the path loss factor. In terms of the threshold of data rate, the minimum data rate R_0 is 1 unit of the minimum required data rate and the maximum data rate R_{max} is 5 units of the minimum data rate. Each experiment is repeated for 100 times and the results are averaged.

Effectiveness. Through comparison with *Optimal*, *Random* and *Greedy*, Fig.s 2 - 4 show the effectiveness of ISUAGame in experiment set #1 and the impacts of three parameters, i.e., the number of SaaS users (n), the number of edge servers (m) and the available resources on the edge servers (c). Overall, *Optimal* allocates the most SaaS users at the lowest overall system cost. Second to *Optimal*, ISUAGame outperforms *Random* and *Greedy* significantly, by 21.98% and 15.25% respectively. Compared to *Optimal*, the performance loss of ISUAGame in both the number of allocated SaaS users and the overall system cost is less than 15% in all cases. On average across all cases in this set of experiments, ISUAGame only allocates 6.62% fewer SaaS users at 9.82% higher system cost than *Optimal*. This demonstrates the high performance of ISUAGame in maximizing the number of allocated SaaS users and minimizing the overall system cost. Fig. 2(b) shows that the increase in n results in a higher overall system cost. This is because more resources are needed, which results in more unallocated SaaS users. Fig. 3(a) shows that, when m increases, more edge servers are able to accommodate more SaaS users. This increases the cost for hiring resources but decreases the cost caused by failures to allocate SaaS users more significantly. Thus, the overall system cost decreases as m increases, as illustrated in Fig. 3(b). In experiment set #1.3, the increase in edge servers' available resources impacts the effectiveness of the comparing approaches in a way similar to the increase in m. As a result, Fig. 4 shows increasing trends for the number of allocated SaaS users and decreasing trends for the overall system cost in a way very similar to Fig. 3. Especially, in Fig. 4(b), the overall system cost first experiences a dramatic decrease and then a slight increase as τ increases. The decrease is caused by a larger number of allocated SaaS users. As τ continues to increase, each SaaS user can be assigned a data rate higher than its minimum requirement on average, which leads to the increase in the overall system cost.

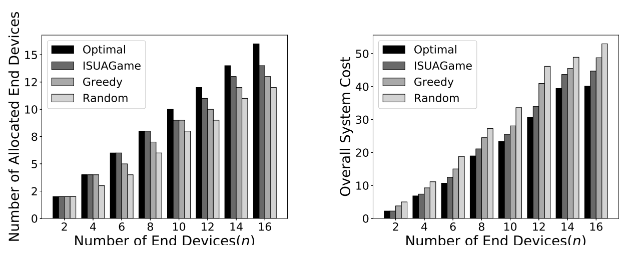
TABLE 1
Experimental Settings

		n	m	c
Small	Set #1.1	2,4,...,16	5	5
	Set #1.2	10	1,...,8	5
	Set #1.3	10	5	1,...,8
Large	Set #2.1	$2^7, ..., 2^{14}$	2^8	60
	Set #2.2	2^{12}	$2^3, ..., 2^{10}$	60
	Set #2.3	2^{12}	2^8	10,20,...,80

Fig.s 5 - 7 show the results in experiment set #2. In general, ISUAGame outperforms *Greedy* and *Random* with different margins in all cases. Overall, the average advantages of ISUAGame over *Greedy* and *Random* are 15.61% and 16.86% in terms of the allocated SaaS users, and 6.76% and 6.27% in terms of the overall system cost. As shown in Fig. 5(a), ISUAGame is capable of allocating much more SaaS users than *Greedy* and *Random*, by 21.39% and 19.05% specifically. Similarly, the advantages of ISUAGame over *Greedy* and *Random* are 10.22% and 9.69% on average when the number of edge servers increases from 8 to 1,024, as demonstrated by Fig. 6(a). As shown in Fig. 7(a), ISUAGame again outperforms *Greedy* and *Random* significantly, by 15.22% and 21.84% respectively. Fig. 7(b) shows that the overall system cost experiences a rapid decrease at $\tau = 20$ and then a gradual increase when τ continues to increase. This phenomenon and the underlying reason are similar to Fig. 4(b).

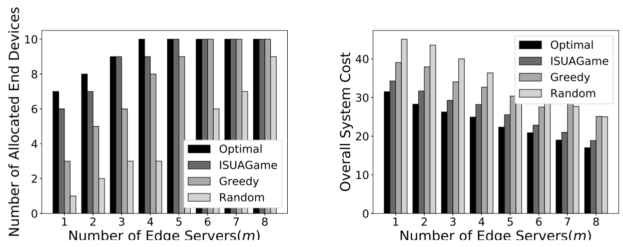
Efficiency. Fig. 8 shows the times taken by *Optimal*,

4. http://ip-api.com/
5. https://github.com/swinedge/eua-dataset



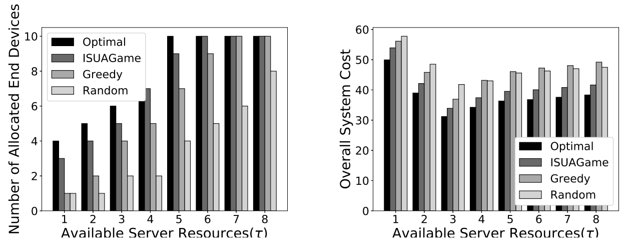
(a) $I_{\{a_i \neq (0,0)\}}/n$ vs. n (b) $Z_{\mathbf{a}}$ vs. n

Fig. 2. Effectiveness vs. Number of SaaS users (Set #1.1)



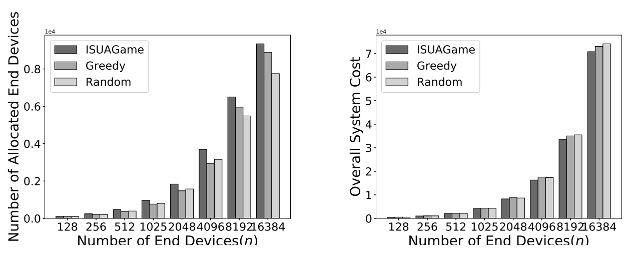
(a) $I_{\{a_i \neq (0,0)\}}/n$ vs. m (b) $Z_{\mathbf{a}}$ vs. m

Fig. 3. Effectiveness vs. Number of Edge Servers (Set #1.2)



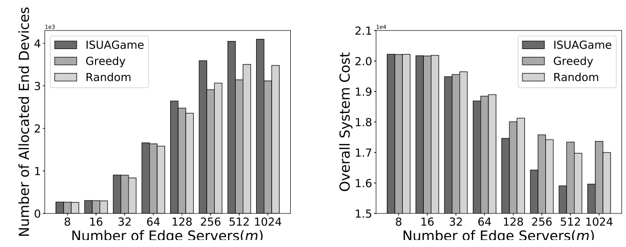
(a) $I_{\{a_i \neq (0,0)\}}/n$ vs. τ (b) $Z_{\mathbf{a}}$ vs. τ

Fig. 4. Effectiveness vs. Server resources (Set #1.3)



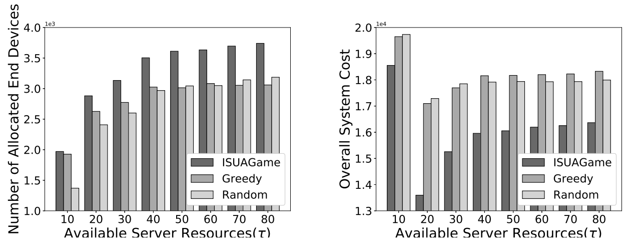
(a) $I_{\{a_i \neq (0,0)\}}/n$ vs. n (b) $Z_{\mathbf{a}}$ vs. n

Fig. 5. Effectiveness vs. Number of SaaS users (Set #2.1)



(a) $I_{\{a_i \neq (0,0)\}}/n$ vs. m (b) $Z_{\mathbf{a}}$ vs. m

Fig. 6. Effectiveness vs. Number of Edge Servers (#2.2)



(a) $I_{\{a_i \neq (0,0)\}}/n$ vs. τ (b) $Z_{\mathbf{a}}$ vs. τ

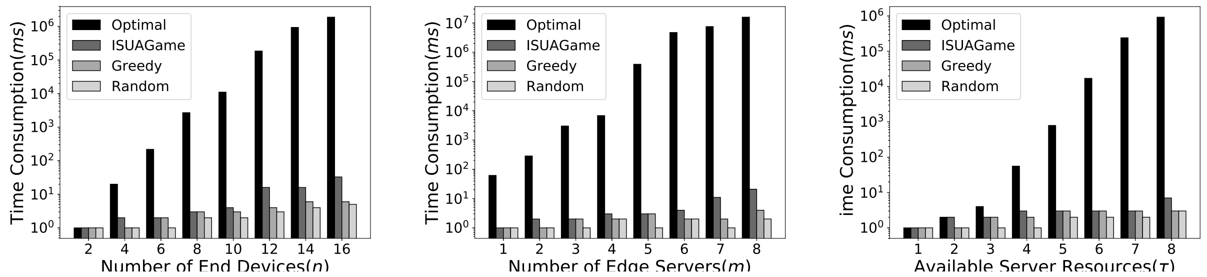
Fig. 7. Effectiveness vs. Available resources (#2.3)

Greedy, *Random* and ISUAGame to find a solution and the impacts of n, m and τ in experiment set #1. Fig. 8 shows that the time consumption of *Optimal* grows exponentially as m increases from 2 to 16, while ISUAGame, *Greedy* and *Random* take almost no time to find a solution. Apparently, it is impractical to employ *Optimal* to solve large-scale ISUA problems. Thus, Fig. 9 and Fig. 10 compares the performance of ISUAGame only against *Greedy* and *Random* in experiment set #2. Fig. 9(a) shows that all approaches can find a solution within 200ms. ISUAGame takes more time than *Greedy* and *Random* in most cases. As demonstrated by Fig. 10(a), when n increases, ISUAGame takes more iterations to reach a Nash equilibrium because in each iteration only one SaaS user can update its allocation decision. Consequently, the time consumption of ISUAGame increases as seen in Fig. 9(a). Fig. 9(a) also shows that the increase in the time consumption of ISUAGame is less significant than *Greedy* and *Random* as n increases. When there is a very large number of SaaS users to accommodate, e.g., 16,384 in the experiment, ISUAGame takes much less time than *Greedy* and *Random* to find a solution. This is because ISUAGame makes the decisions for individual SaaS users simultaneously in each iteration while *Greedy* and *Random* can only make decisions for SaaS users one after another. Fig. 9(b) shows that when the problem scales up in m, ISUAGame takes more time to find a solution. More edge servers provide each SaaS user with more choices on average. It takes more iterations and consequently more time for the SaaS users' decisions to stabilize, as demonstrated in Fig. 10(b). Even so, ISUAGame can still find a solution when there are as many as 1,024 edge servers to consider. Fig. 9(c) shows that the time consumption of ISUAGame increases mildly with the increase in τ. When the available resources on edge servers are tight with $\tau = 10$, ISUAGame takes less time than *Greedy* and *Random* to find a solution. As τ increases, every edge server has more resources to accommodate more SaaS users. This gives each SaaS user more choices and requires more iterations for all the SaaS users to reach a Nash equilibrium, as demonstrated by Fig. 10(c). Consequently, the time consumption of ISUAGame increases. As τ reaches and exceeds 60, the available resources on most edge servers are more than enough to accommodate their nearby SaaS users. Most SaaS users can easily maximize their individual benefits and do not need to update their allocation decisions. The time consumption of ISUAGame does not increase significantly because its convergence time increases only slightly.

The results demonstrated in Fig.s 10 and 9 show that ISUAGame scales with all three parameters, i.e., n, m and τ, which indicates its high efficiency. This is critical to large-scale real-world applications in the EC environment because finding the centralized optimal solutions to large-scale NP-hard ISUA problems is impractical as discussed and proved in Section 2.4.

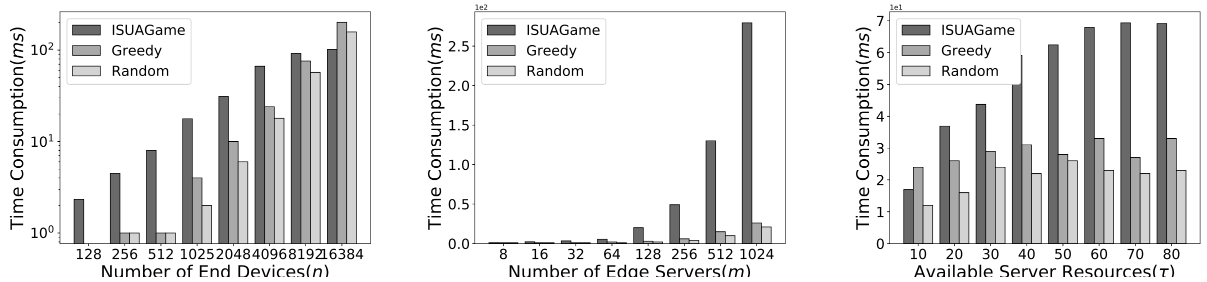
5 Related Work

In 2012, Cisco [38] proposed the edge computing (EC) paradigm, which is sometimes referred to as fog computing. As an extension of cloud computing, the EC paradigm pushes computing resources to the edge of the cloud by



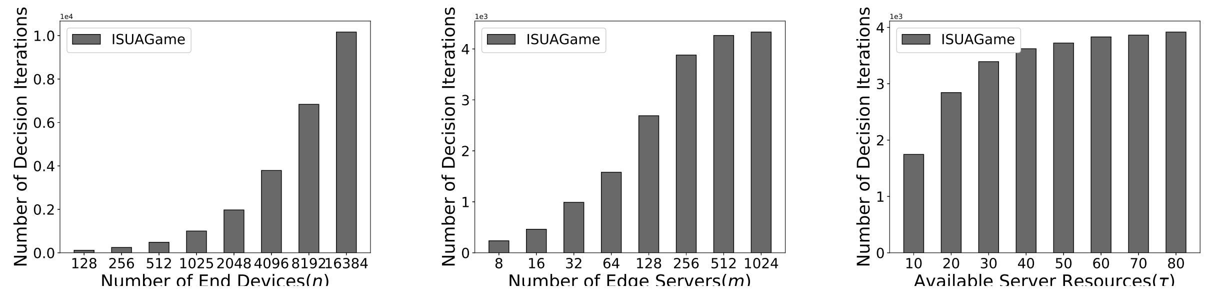
(a) *effi* vs. n (Set #1.1) (b) *effi* vs. m (Set #1.2) (c) *effi* vs. τ (Set #1.3)

Fig. 8. Average Time Consumption (Set #1)



(a) *effi* vs. n (Set #2.1) (b) *effi* vs. m (Set #2.2) (c) *effi* vs. τ (Set #2.3)

Fig. 9. Average Time Consumption (Set #2)



(a) *effi* vs. n (Set #2.1) (b) *effi* vs. m (Set #2.2) (c) *effi* vs. τ (Set #2.3)

Fig. 10. Average Number of Iterations of ISUAGame (Set #2)

distributing edge servers across locations geographically close to SaaS users, e.g., base stations. A lot of researchers have investigated the computation offloading problem in the last few years with a focus on the performance of EC systems, including energy efficiency, the network latency, and system throughput.

You et al. [11] propose a suite of optimal and near-optimal approaches for allocating network resources in EC based on two wireless access protocols, i.e., TDMA and OFDMA. Chen et al. [8] propose a game-theoretic approach - similar to their approach in [3] - that allocates the wireless channels of an edge server for multiple end-users. Neto et al. [39] attempt to improve end-users' energy efficiency during computation offloading by profiling and predicting end-users' task execution times and energy consumption. Instead of end-users' energy efficiency, some researchers investigate the energy efficiency of the edge servers from the network provider's perspective. Wang et al. [12] study computation offloading with the aim to minimize an individual edge server's energy consumption subject to end-users' constraints for latency. Chen et al. [10] also attempt to ensure edge servers' energy efficiency. They propose an online peer computation offloading framework where edge servers can offload computation tasks to each other.

Computation offloading incurs costs for the network provider who deploys, runs and leases the edge servers. Such costs are often considered in research on EC problems. Yao et al. [40] propose an approach to help the network provider deploy edge servers in a cost-effective manner. Yin et al. [18] tackle a similar edge server deployment problem, also with the objective to optimize the performance of EC systems and server deployment costs. Wang et al. [14] model the computation offloading problem as a convex problem and then decompose it so that it can be solved in a distributed manner. In their study, they aim to maximize the network provider's revenue which is collectively determined by the computation offloading decision, resource allocation and content caching strategy.

EC inherits the pay-as-you-go pricing model from cloud computing, which allows SaaS vendors such as Youtube and Uber to hire resources on edge servers from network providers to host their applications and to serve their users.

Thus, the cost incurred for SaaS vendors is critical to the success of EC. Unfortunately, all the above existing works tackle the computation offloading problem from either the end-users' or the network provider's perspectives, none from the SaaS vendor's perspective as to how to cost-effectively serve their SaaS users. This new problem is first studied in [19] with the aim to help the SaaS vendor serve the maximum number of users at the minimum overall system cost. However, the work presented in [19] suffers from two major limitations. First, the overall system cost is simply measured by the number of hired edge servers. Second, the wireless communication interference that may be caused by allocating excessive SaaS users to an individual edge server is not considered. The consequence is that the SaaS users may not actually achieve satisfactory data rates. The wireless communication interference is a critical issue in the EC environment. For example, Chen et al.'s approach considers wireless communication interference from the network provider's perspective while offloading end-users' computation tasks to a single edge server with multiple wireless channels [8]. However, that approach can handle only one single edge server. Guo et al. [22] and Tran et al. [23] take a step forward by tackling the computation offloading problem in an EC environment with multiple edge servers and multiple wireless channels. However, in their study, they assume that all the wireless channels are orthogonal, i.e., one wireless channel can accommodate only one computation task at a time. This oversimplifies the problem by converting the problem of computation offloading across edge servers to across wireless channels.

In this paper, we propose ISUAGame to tackle the ISUA problem with consideration of wireless communication interference. ISUAGame overcomes the limitations of state-of-the-art research by 1) modeling the SaaS vendor's benefit and cost based on multi-dimensional resources; and 2) tackling the ISUA problem from the SaaS vendor's perspective in a realistic EC environment where multiple edge servers are available with non-orthogonal wireless channels.

6 Conclusion and Future Work

In this paper, we proposed a game-theoretical approach, namely ISUAGame, for solving the interference-aware SaaS user Allocation (ISUA) problem from the SaaS vendor's perspective in the edge computing environment. We show that the ISUA problem is NP-hard and formulate it as a game, which is proven to be a potential game and admits a Nash equilibrium. We then designed a decentralized algorithm for finding the Nash equilibrium as the solution to the ISUA problem. Finally, we evaluate the performance of ISUAGame first theoretically and then experimentally on a widely-used real-world dataset.

In our future work, we will investigate the impact to SaaS users' mobility, their dynamic participation as well as departures on the ISUA problem. This will allow ISUAGame to accommodate more sophisticated ISUA scenarios where SaaS users join, move and depart dynamically.

Acknowledgments

This research is funded by Australian Research Council Discovery Projects (DP170101932 and DP180100212).

References

[1] N. Heuveldop, "Ericsson mobility report," *Ericsson, Stockholm*, 2017.

[2] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2019.

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

[4] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *2012 IEEE symposium on Computers and communications (ISCC)*. IEEE, 2012, pp. 59–66.

[5] J. Cohen, "Embedded speech recognition applications in mobile phones: Status, trends, and challenges," in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008*. IEEE, 2008, pp. 5352–5355.

[6] C. Ma and Y. Yang, "A battery-aware scheme for routing in wireless ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 8, pp. 3919–3932, 2011.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

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

[9] N. Fernando, S. W. Loke, and W. Rahayu, "Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 329–343, 2019.

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

[11] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[12] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.

[13] S. E. Mahmoodi, R. Uma, and K. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 301–313, 2019.

[14] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[15] A. Bourdena, C. Mavromoustakis, G. Mastorakis, J. Rodrigues, and C. Dobre, "Using socio-spatial context in mobile cloud offload process for energy conservation in wireless devices," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 392–402, 2019.

[16] J. Barrameda and N. Samaan, "A novel statistical cost model and an algorithm for efficient application offloading to clouds," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 598–611, 2018.

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

[18] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge provisioning with flexible server placement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1031–1045, 2017.

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

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

[21] E. Cuervo, A. Wolman, L. P. Cox, K. Lebeck, A. Razeen, S. Saroiu, and M. Musuvathi, "Kahawai: High-quality mobile gaming using gpu offload," in *13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2015, pp. 121–135.

[22] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, 2018.

[23] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[24] R. B. Myerson, *Game theory*. Harvard university press, 2013.

[25] M. Xiao, N. B. Shroff, and E. K. Chong, "A utility-based power-control scheme in wireless cellular systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 2, pp. 210–221, 2003.

[26] M. Chiang, P. Hande, T. Lan, C. W. Tan *et al.*, "Power control in wireless cellular networks," *Foundations and Trends® in Networking*, vol. 2, no. 4, pp. 381–533, 2008.

[27] T.-S. Kim, H. Lim, and J. C. Hou, "Improving spatial reuse through tuning transmit power, carrier sense threshold, and data rate in multihop wireless networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. ACM, 2006, pp. 366–377.

[28] S. C. Jocke, J. F. Bolus, S. N. Wooters, A. Jurik, A. Weaver, T. Blalock, and B. Calhoun, "A 2.6-μw sub-threshold mixed-signal ecg soc," in *2009 Symposium on VLSI Circuits*. IEEE, 2009, pp. 60–61.

[29] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of npcompleteness (series of books in the mathematical sciences), ed," *Computers and Intractability*, vol. 340, 1979.

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

[31] C. A. Holt and A. E. Roth, "The nash equilibrium: A perspective," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 3999–4002, 2004.

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

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

[34] ——, "Joint strategy fictitious play with inertia for potential games," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 208–220, 2009.

[35] T. Roughgarden, *Selfish routing and the price of anarchy*. MIT press Cambridge, 2005, vol. 174.

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

[37] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*. prentice hall PTR New Jersey, 1996, vol. 2.

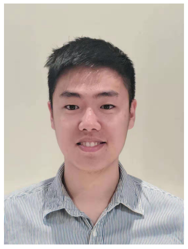
[38] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[39] J. L. D. Neto, S.-y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, "Uloof: A user level online offloading framework for mobile edge computing," *IEEE Transactions on Mobile Computing*, 2018.

[40] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 16, p. e3975, 2017.

Guangming Cui received his Master degree from Anhui University, China, in 2018. He is a PhD candidate at Swinburne University of Technology. His research interests include software engineering, edge computing and service computing.

Qiang He received his first PhD degree from Swinburne University of Technology, Australia, in 2009 and his second PhD degree in computer science and engineering from Huazhong University of Science and Technology, China, in 2010. He is a senior lecturer at Swinburne. His research interests include edge computing, cloud computing, software engineering and service computing. More details about his research can be found at https://sites.google.com/site/heqiang/.



Xiaoyu Xia received his Master degree from The University of Melbourne, Australia in 2015. He is a PhD candidate at Deakin University. His research interests include edge computing, service computing and software engineering.

Phu Lai received his MSc degree in Information Technology in 2017 and is currently working toward a PhD degree at Swinburne University of Technology, Australia. His research interests include software engineering, cloud computing and edge computing.

Feifei Chen received her PhD degree from Swinburne University of Technology, Australia, in 2015. She is a lecturer at Deakin University. Her research interests include software engineering, edge computing, cloud computing and green computing.

Tao Gu Tao Gu received his PhD degree from the National University of Singapore. He is currently an Associate Professor at RMIT University, Australia. His current research interests include mobile computing, ubiquitous computing, wireless sensor networks, sensor data analytics, and Internet of Things.

Yun Yang received his PhD degree from the University of Queensland, Australia, in 1992, in computer science. He is currently a full professor in the School of Software and Electrical Engineering at Swinburne University of Technology, Melbourne, Australia. His research interests include software technologies, cloud computing, workflow systems, and service computing.