



Cost-Effective User Allocation in Mobile Edge Computing

Presented in fulfillment of the requirements of the degree of Doctor of Philosophy

by

Phu Lai

*Department of Computing Technologies
School of Science, Computing, and Engineering Technologies
Swinburne University of Technology
Melbourne, Australia*

August 2021

Abstract

Mobile edge computing (MEC) aims to tackle one of the most challenging obstacles in cloud computing – high and unpredictable latency. By deploying edge servers at cellular base stations, mobile network operators can offer computing resources at the network edge within users' close geographic proximity. Mobile application vendors can rent these computing resources to host their applications and serve their users with low latency. Apart from the many challenges arisen with the development of MEC that have been discovered and studied, we have identified a new problem – edge user allocation (EUA). In the EUA problem, an app vendor needs to allocate its users to edge servers to achieve certain optimization objectives such as minimizing the computing resource cost, maximizing the number of users allocated to edge servers, or improving quality of service (QoS), etc. In this thesis, we focus on the formulation of cost-effective solutions to edge user allocation.

First, we establish the a core foundation for the EUA problem by modeling it as a variable-sized vector bin packing problem, which aims to maximize the number of users allocated to edge servers while minimizing the number of edge servers required to serve the allocated users. We propose an optimal approach to find optimal solutions to this problem using the Lexicographic Goal Programming technique. Due to its NP-hardness, we propose a heuristic to efficiently solve this NP-hard problem in large-scale scenarios.

Secondly, based on the foundation established above, where the users' quality of service is fixed, we now consider the scenario where an app vendor can flexibly adjust its users' QoS to maximize user satisfaction measured by the quality of experience (QoE). We propose an optimal approach to find optimal solutions and a heuristic to efficiently solve this QoS-aware EUA problem. The experimental results show a limitation of the proposed heuristic (low QoE when the number of users is very high). Thus, we attempt to tackle this limitation by formulating this problem as a potential game then solve it with a decentralized game-theoretical approach that is guaranteed to reach a Nash equilibrium as the final solution.

In the previous two research problems, users are stationary/static, which could make our work imprac-

tical in a dynamic scenario where users randomly come and go over time. This leads us to the third research problem. Here, we consider a time-slotted system. A user can be allocated to either an edge server or the remote cloud. If the assigned edge server is full, the user will be placed in a waiting list waiting for service, which incurs an allocation delay cost. We aim to allocate users to maximize the time-average system benefit measured by the number of allocated users, allocation delay cost, and cloud latency cost. To solve this problem, we propose an online algorithm based on the Lyapunov optimization framework, which is very effective in optimizing a metric in the long term while stabilizing the system.

Finally, having realized that wireless communication is an integral part of mobile edge computing and should not be neglected, we attempt to solve the user allocation problem while considering the key characteristics in wireless communication, namely multi-channel, achievable data rate, and non-orthogonal multiple access (NOMA), etc. NOMA is an emergent 5G/6G multiple-access technique that allows multiple users to share the same wireless channel by multiplexing users in the power domain, enabling the massive connectivity demanded by 5G/6G. We aim to allocate users to proper channels in edge servers to minimize the system cost, which is measured by computing resource cost and transmit power cost. To solve this problem, we formulate it as an ordinal potential game then find a Nash equilibrium with a decentralized game-theoretical approach. Since this is a static scenario where users are stationary, we extend this problem by investigating a scenario where users come and go over time. We aim to minimize the long-term system cost while stabilizing users' overall data rate.

In all four research problems above, a series of rigorous experiments and theoretical analyses have been conducted against a number of baseline and state-of-the-art approaches to demonstrate the effectiveness and efficiency of the proposed approaches.

Acknowledgments

During my PhD candidature, I have received tremendous support from many people, for which I am extremely grateful. They are the main reasons for my smooth journey, even during the COVID-19 pandemic. My study would not have been the same without their support. I acknowledge the financial support from Swinburne University of Technology and Australian Research Council Discovery Project grants DP170101932, DP180100212, DP200102491, and Laureate Fellowship FL190100035.

First and foremost, I would like to express my gratitude to my principle supervisor, Associate Prof. Qiang He, on both professional and personal levels. His enormous support, guidance, encouragement, patience, and expertise are what have made me the academic I am today. I would also like to thank my associate supervisors, Prof. Yun Yang, Prof. John Grundy, Associate Prof. Mohamed Abdelrazek, and Prof. John Hosking for their great support and contribution to my research. I really could not have asked for more from this amazing supervision panel. Next, I would like to thank my research team members for their generous support, especially Guangming Cui and Xiaoyu Xia.

I would also like to thank my parents, my grandparents, my aunty, my brother, and my cousins for their trust and understanding. Jay, Angel, and Tofoo the cats are also of great emotional support.

Phu Lai,

August 2021

Declaration

In accordance with the Swinburne University Higher Degree by Research Committee regulations, I, Phu Lai, hereby declare that the examinable outcome:

1. contains no material that has been accepted for the award to the candidate of any other degree or diploma, except where prior permission to do so has been received by the ADRD, and with due reference made about this in the text of the examinable outcome;
2. to the best of the candidate's knowledge contains no material previously published or written by another person except where due reference is made in the text of the examinable outcome, and with permission received to republish the work in the thesis; and
3. where the work is based on joint research or publications, discloses the relative contributions of the respective creators or authors using the Authorship Declaration Form.

I warrant that I have obtained, where necessary, permission from the copyright owners to use any third party copyright material reproduced in the thesis (such as artwork, images, unpublished documents), or to use any of my own published work (such as journal articles) in which the copyright is held by another party (such as publisher, co-author).

Phu Lai,

August 2021

Contents

- Abstract** **i**

- Acknowledgments** **iii**

- Declaration** **iv**

- List of Publications** **viii**

- 1 Introduction** **1**
 - 1.1 Research Background and Motivation 1
 - 1.2 Motivating Scenarios 3
 - 1.2.1 Scenario 1 3
 - 1.2.2 Scenario 2 4
 - 1.2.3 Scenario 3 5
 - 1.2.4 Scenario 4 6
 - 1.3 Key Research Problems 8

1.4	Literature Review	9
1.4.1	User Allocation in general MEC Systems	11
1.4.2	User Allocation in NOMA-based MEC Systems	13
1.5	Overview of the Publications (Chapters) in this Thesis	15
1.6	Thesis Organization	16
2	Cost-Effective User Allocation in Mobile Edge Computing Systems	17
3	Quality of Service-Aware User Allocation in Mobile Edge Computing Systems	35
3.1	An Integer Programming-based Approach and A Greedy Heuristic Approach	35
3.2	An Iterative Heuristic Approach	52
3.3	A Game-Theoretical Approach	66
4	Dynamic User Allocation in Stochastic Mobile Edge Computing Systems	76
5	Cost-Effective User Allocation in NOMA-based Mobile Edge Computing Systems	90
5.1	A Static Scenario	91
5.2	A Dynamic Scenario	111
6	Conclusions, Limitations, and Future Work	121
6.1	Key Contributions and Conclusions	121
6.2	Limitations and Future Work	123
	References	124

A Authorship Statements

131

List of Publications

Below is a list of published and accepted publications arisen from the research conducted as part of this study:

1. **Phu Lai**, Qiang He, Guangming Cui, Feifei Chen, John Grundy, Mohamed Abdelrazek, John Hosking, Yun Yang, Cost-Effective User Allocation in 5G NOMA-based Mobile Edge Computing Systems, IEEE Transactions on Mobile Computing, 2021. DOI: 10.1109/TMC.2021.3077470
2. **Phu Lai**, Qiang He, Xiaoyu Xia, Feifei Chen, Mohamed Abdelrazek, John Grundy, John Hosking, Yun Yang, Dynamic User Allocation in Stochastic Mobile Edge Computing Systems, IEEE Transactions on Services Computing, 2021. DOI: 10.1109/TSC.2021.3063148
3. **Phu Lai**, Qiang He, John Grundy, Feifei Chen, Mohamed Abdelrazek, John Hosking, Yun Yang, Cost-Effective App User Allocation in an Edge Computing Environment, IEEE Transactions on Cloud Computing, 2020. DOI: 10.1109/TCC.2020.3001570.
4. **Phu Lai**, Qiang He, Guangming Cui, Xiaoyu Xia, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, Yun Yang, QoE-aware User Allocation in Edge Computing Systems with Dynamic QoS, Future Generation Computer Systems, Vol 112, pp. 684-694, 2020.
5. **Phu Lai**, Qiang He, Guangming Cui, Feifei Chen, Mohamed Abdelrazek, John Grundy, John Hosking and Yun Yang, Quality of Experience-Aware User Allocation in Edge Computing Systems: A Potential Game, 40th IEEE International Conference on Distributed Computing Systems, pp. 223-233, Singapore, 2020.
6. **Phu Lai**, Qiang He, Guangming Cui, Xiaoyu Xia, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang, Edge User Allocation with Dynamic Quality of Service,

17th International Conference on Service-Oriented Computing, pp. 86-101, Toulouse, France, 2019.

7. **Phu Lai**, Qiang He, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang, Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing, 16th International Conference on Service-Oriented Computing, pp. 230-245, Hangzhou, China, 2018. **Best Research Paper Award.**

Below is a list of submitted publications, awaiting decisions:

1. **Phu Lai**, Qiang He, Feifei Chen, John Grundy, Mohamed Abdelrazek, John Hosking, Yun Yang, Online User Allocation in Dynamic NOMA-based Mobile Edge Computing Systems, submitted to IEEE International Conference on Computer Communications (INFOCOM), under review.

Introduction

This thesis investigates the edge user allocation (EUA) problem in mobile edge computing (MEC) – an emergent distributed computing paradigm that brings computation power to the network edge, closer to where it is required, to improve end-to-end latency for end-users. We tackle the EUA problem from an app vendor’s perspective, aiming to help them minimize several types of associated costs when running their applications and services on edge servers by systematically allocating users to proper edge servers.

This chapter provides an overview of this thesis. We introduce the background, motivation, key research problems, our major contributions, and the outline of this thesis.

1.1 Research Background and Motivation

Recently, we are observing a rapid growth in mobile and IoT device subscriptions, including smartphones, wearables, environmental sensors, self-driving vehicles, etc. This comes with a rich variety and sophistication of applications and services, such as critical response systems, vital monitoring systems, facial recognition, interactive VR/AR gaming, ultra-low latency streaming, and so on. They usually require intensive processing power and large energy capacity, which are not available on thin clients such as mobile or IoT devices. Traditionally, heavy computation tasks are offloaded to app vendors’ servers in the cloud. Nevertheless, maintaining a low-latency connection to users is a major challenge for app vendors due to the skyrocketing number of connected devices, the increasing network traffic and computational load, plus the long distance between users and the cloud, all of which severely defeat the purpose of time-sensitive applications and services embracing 5G/6G.

Network latency impacts application performance, quality of service (QoS), and user quality of experience (QoE) significantly. This is one of the main reasons why edge computing, sometimes referred to as fog computing [1], has emerged to tackle the challenge of high network latency. Mobile edge computing [2] takes advantage of the highly distributed cellular base station environment. In an MEC system, numerous edge servers, which provide both processing power [3, 4] and storage [5, 6], are deployed at base stations [7]. App vendors like Uber or Youtube can deploy their apps on edge servers, which are in closer geographic proximity to their users than the cloud, to remarkably reduce the latency in accessing those apps [8, 9].

In an MEC system, edge servers are densely distributed. The coverage areas of adjacent edge servers¹ usually partially overlap to avoid non-service areas [10, 11] – the areas in which users cannot be served by any edge server. A user located in an overlapping area will be allocated to one of the edge servers covering them as long as that edge server has sufficient computing and communication resources, e.g. CPU, RAM, storage, or bandwidth, to serve the user. In a real-world MEC system, there could be numerous of such overlapping areas due to the dense distribution of base stations (up to 50 per km² [12]).

The problem of allocating users to suitable edge servers in an MEC system is referred to as an EUA problem. Compared to a cloud server, a typical edge server comes with constrained computing, storage and communication resources due to its size limit [13, 14]. Thus, an ineffective user-to-edge-server allocation may exhaust edge server resources rather quickly, leaving no available resources to serve more users. Not only do app vendors have to consider the edge server capacity, but they also need to take into account several factors such as user satisfaction, achievable data rate, edge server operating costs, etc. Without considering those factors would negatively affect both users and app vendors. On the user side, users would be less likely to receive a satisfactory service while using the applications or services provided by the app vendors; for example, low data rate, poor QoS/QoE level, or even unable to use the services. On the app vendor side, failure to satisfy their users would lead to serious financial consequences and reputational damages. Furthermore, app vendors must properly manage the rented resources on edge servers to fully utilize their investments and avoid incurring too much extra business cost.

Unlike the computation offloading problem which challenges the edge infrastructure providers, e.g., AT&T, Vodafone, or T-Mobile, the EUA problem challenges the app vendors who rent computing

¹We speak interchangeably of edge servers and base stations. For the sake of consistency, we will hereafter try to use the term "edge server" instead of "base station". In situations where the communication/networking aspects are discussed, "base station" will be used.

resources to serve their own users. The rapid growth of mobile subscriptions promoted by 4G and the forthcoming 5G/6G, which is predicted to reach 9 billions in 2025 [15], has put a great burden on the existing infrastructure and made this NP-hard problem even more complicated. The EUA problem has been gaining a lot of attention [16–34] in recent years since we first introduced it in [11]. In this thesis, we tackle it from different angles in increasingly sophisticated scenarios, accompanied by extensive experimental and theoretical evaluation.

1.2 Motivating Scenarios

A representative application that can utilize the benefits of MEC is online virtual-reality (VR) gaming. This type of applications require a significant amount of compute power for graphics rendering, which is not available on players' mobile devices or any other thin clients such as IoT sensors, drones, etc. Conventionally, mobile devices could offload heavy computation tasks to centralized cloud servers. A major drawback of this model is the high network latency due to the long distance between the players and the remote cloud servers. MEC addresses this issue by pushing computing power closer to players (to be referred to as "users" hereafter for the sake of consistency with the rest of the thesis). The game vendor (to be referred to as "app vendor" hereafter for the sake of consistency with the rest of the thesis) needs to allocate its users to edge servers, which host applications to process the users' tasks. Next, we introduce several examples of the EUA problem in different scenarios. Those motivating scenarios inspire the four research problems studied in this thesis, which are discussed in Section 1.3.

1.2.1 Scenario 1

Fig. 1.1 illustrates a small example of an MEC system with 3 base stations, each has an edge server, and 8 users. An edge server can only serve users that are located within the cell coverage area of the base station hosting the edge server. For example, edge server 1 would not be able to serve any users other than users 1, 2, and 3. This is referred to as the *proximity constraint* throughout this thesis. Each edge server has a limited amount of computing resources, denoted by a vector $\langle \text{CPU, RAM, storage, bandwidth} \rangle$. Edge servers have to dedicate different amounts of computing resources to accommodate different users. For example, serving a user who selects a higher graphics setting (e.g., 4K resolution, spatial anti-aliasing, etc.) consumes more resources than serving a user with a lower setting (e.g., 720p resolution, etc.). The total resource requirements of all users allocated to an edge server must not exceed the available computing resources of that edge server. This is referred to as the *capacity*

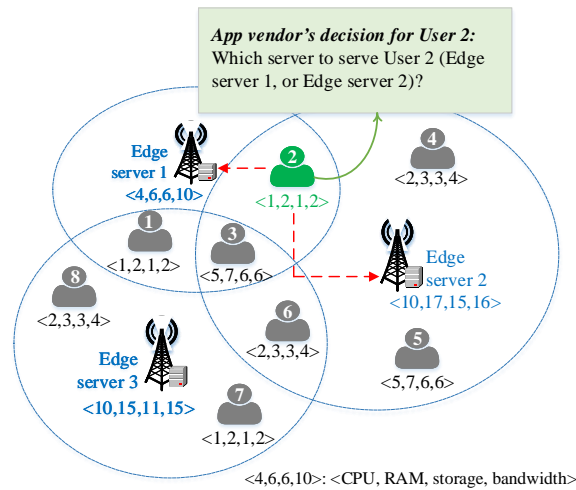


Figure 1.1: An example of an MEC system (Scenario 1)

constraint. As illustrated in Fig. 1.1, there are five users located within the coverage of edge server 3, whose aggregated resource requirement is $\langle 11, 17, 14, 16 \rangle$, which exceeds the available resources of edge server s3 ($\langle 10, 15, 11, 15 \rangle$). Thus, the app vendor needs to allocate one or some of those users to other edge servers. **In order to save the cost of operating edge servers, we aim to minimize the number of edge servers needed to be used and maximize the number of allocated users.**

A possible allocation is to allocate users 1 and 2 to server 1, users 4, 5, and 6 to server 2, and users 3, 7, and 8 to server 3. Neither proximity nor resource constraints is violated in this way. However, this might not be the best solution. A better solution would be to allocate users 2, 4, 5, and 6 to server 2, and users 1, 3, 7, and 8 to server 3. This allocation does not require edge server 1 at all.

1.2.2 Scenario 2

In Scenario 1, each user is assigned a fixed QoS level (video resolution in this example). Each QoS level corresponds to an amount of computing resources required by an edge server (represented by a multi-dimensional vector $\langle \text{CPU, RAM, storage, bandwidth} \rangle$). In the example in Fig. 1.1, we can see that there are three possible QoS levels, namely $\langle 1, 2, 1, 2 \rangle$ (720p, assigned to users 1, 2, and 7), $\langle 2, 3, 3, 4 \rangle$ (1080p, assigned to users 4, 6, and 8), and $\langle 5, 7, 6, 6 \rangle$ (1440p, assigned to users 3 and 5). Now, in Scenario 2 (Fig. 1.2), we consider situations where the QoS level of a user can be flexibly adjusted. Each user can be assigned any of the three possible QoS levels, depending on the app vendor's decision. Different QoS levels results in different user QoE levels. Users with higher video resolutions would have a better experience (higher QoE) than users with lower video resolutions. Beside picking a QoS level for each user, the app vendor also needs to determine an edge server to serve the user. **In**

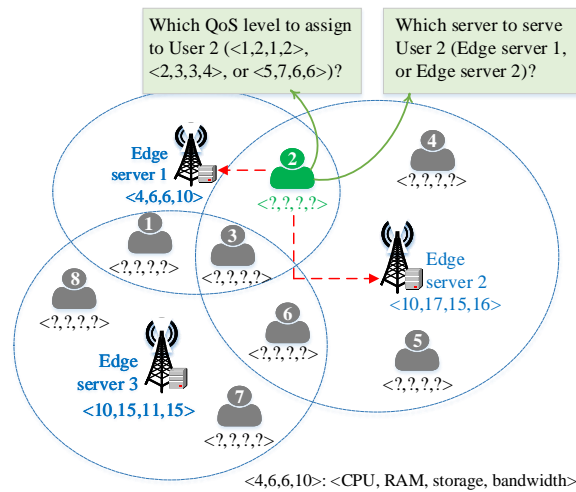


Figure 1.2: An example of an MEC system (Scenario 2)

this scenario, we aim to maximize the accumulative QoE of allocated users.

To achieve this objective, one would think of assigning the highest QoS level to every user. This, however, can be very expensive and will exhaust edge servers' resources quickly, leaving no resources for other users. Note that the correlation between QoS and QoE is not linear, which further complicates the EUA problem. For the majority of users, there is no perceptible difference between 1080p and 1440p video resolutions on their mobile devices, or even between 1080p and UHD from a distance farther than 1.5x the screen height regardless of the screen size [35]. Servicing a high definition video quality like 1440p or UHD certainly consumes more resources (bandwidth, CPU, and GPU), which might be unnecessary since most users on their mobile devices are likely to be satisfied with 1080p. Instead, those resources can be utilized to serve users who are currently unhappy with the service, e.g. those experiencing poor 240p or 360p graphics, or those not able to use the app at all due to all nearby edge servers being overloaded. Therefore, the QoS levels of some users can be lowered, potentially without causing any noticeable QoE downgrade, in order to better service users experiencing low QoS levels. In this way, the users' overall satisfaction can be maximized.

1.2.3 Scenario 3

In the previous two scenarios, the temporal dimension is not considered; the app vendor only needs to allocate a fixed and known set of users. Here, we consider a scenario where users randomly come and go over time (Fig. 1.3). The operational timeline of this MEC system is represented by a series of time slots. In each time slot, there is a random number of existing users leave the system and a random of new users. The app vendor has no knowledge of future user arrivals and departures. As computing

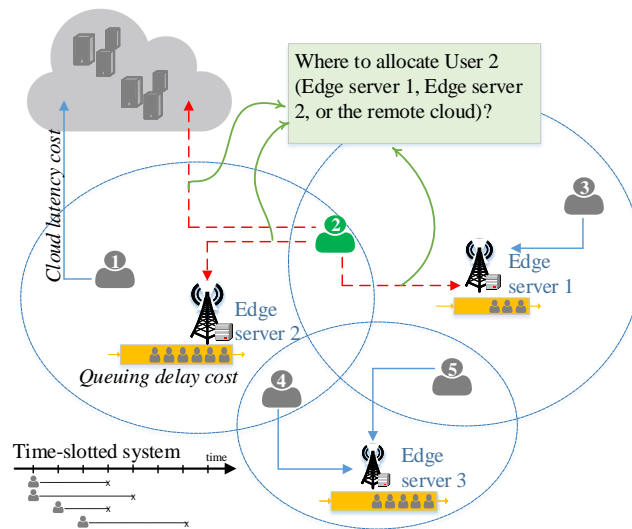


Figure 1.3: An example of an MEC system (Scenario 3)

resources on edge servers are limited, edge servers might be exhausted and cannot serve new users. In this case, there are two options for each new user: (1) be allocated to an edge server and wait to be served once one or more existing users leave and released the occupied computing resources, and (2) be allocated to the remote cloud server to be served straight away as cloud servers typically have abundant resources. The first option incur a queuing delay cost as the user has to wait for a certain amount of time until being served. The second option incurs a cloud latency cost as the remote cloud server is further away from the user than an edge server. We define the throughput benefit as the benefit obtained from allocating users to edge servers. **In this scenario, our objective is to maximize the system cost, which is measured by the throughput benefit and the costs of queuing delay and cloud latency.**

1.2.4 Scenario 4

The communication aspect of MEC has not been incorporated in the previous three scenarios. Communication is an important component of MEC, along with computation, and thus must not be neglected. Here in Scenario 4, we start to incorporate several features of a typical wireless communication network. First, we consider uplink transmissions rather than downlink transmissions because uplink is important for applications that transmits a great amount of data to users. Taking an online VR gaming application for instance, a user would request an edge server to render game graphics, which cannot be done on thin clients like mobile devices. The rendered graphics will then be sent back to the user.

Secondly, we incorporate non-orthogonal multiple access (NOMA) – an emergent multi-access technique that allows multiple users to share wireless channels by multiplexing users in the power domain,

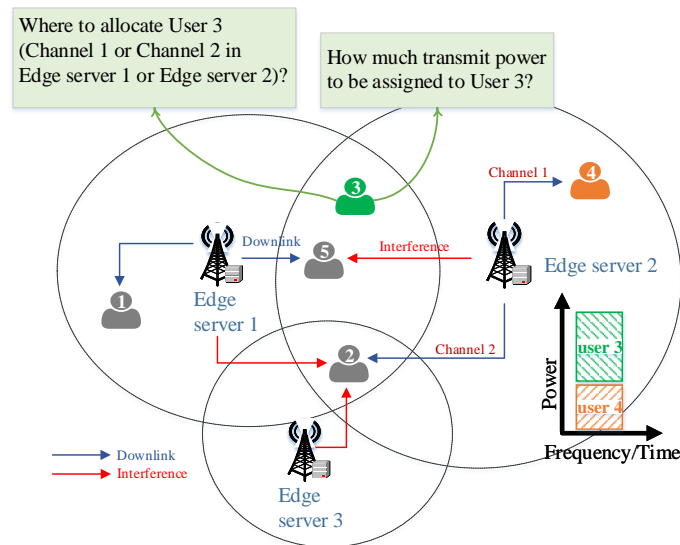


Figure 1.4: An example of an MEC system (Scenario 4)

enabling the massive connectivity demanded by the upcoming 5G/6G era. A base station, which hosts an edge server, can serve users via multiple communication channels. In the example in Fig. 1.4, each base station has two channels, Channel 1 and Channel 2. A user could experience intra-cell interference (caused by users sharing the same channel in the same base station) and inter-cell interference (caused by users sharing the same channel in neighbor base stations). In NOMA, the transmit power allocated to a user is dependent on the channel condition of that user. For example, if user 3 is allocated to channel 1 in base station 2, user 3 would experience the intra-cell interference caused by user 4, and inter-cell interference caused by those allocated to channel 1 in base station 1. User 4 has better channel condition than user 3 because of being closer to base station 2 and experience no inter-cell interference. Therefore, user 3 should be allocated less transmit power than user 3 so that every user can achieve a target data rate.

From the example above, we can see that the choices of base stations (edge servers), channels, and transmit powers for all users are all intertwined and must be jointly considered. **In this scenario we aim to minimize the total amount of transmit power allocated to all users.** We consider two sub-scenarios: (1) a static scenario, where the number of users is fixed and time is not considered – similar to Scenarios 1 and 2, and (2) a dynamic scenario, where users randomly come and go over time – similar to Scenario 3. In the dynamic scenario, the allocation delay cost (equivalent of the queuing delay cost in Scenario 3) must also be minimized.

1.3 Key Research Problems

There are numerous kinds of applications and services in an MEC system, all with a number of unique and specific characteristics in different scenarios that require investigation. The four motivating scenarios identified in Section 1.2 correspond to the following four research problems:

- **Research Problem 1** (Addressing aims of Scenario 1 – Section 1.2.1): The MEC system being investigated is quasi-static, where users are static, e.g., traffic cameras, smart sensors, or mobile users who do not move much, etc. The QoS level of each user is fixed. The capacity and proximity constraints must not be violated. An app vendor needs to allocate as many users to as fewest edge servers as possible to minimize the cost of renting resources on edge servers.
- **Research Problem 2** (Addressing aims of Scenario 2 – Section 1.2.2): The MEC system being investigated is quasi-static, similar to Research Problem 1. The QoS level of a user can now be flexibly adjusted. For example, for a video-streaming service, the app vendor can determine the resolution of the videos being streamed to each user such as 480p, 720p, or 1080p, etc. Each QoS level is associated with a level of user satisfaction (QoE). An app vendor needs to properly allocate users to edge servers and select a QoS level for each user so that the total user satisfaction is maximized while not violating the capacity and proximity constraints.
- **Research Problem 3** (Addressing aims of Scenario 3 – Section 1.2.3): The MEC system being investigated is now dynamic, where users come and go randomly over time. An app vendor does not know how many and when users would come or go. A user can be allocated to either an edge server or the remote cloud. If the user is allocated to an edge server whose resources are exhausted, the user would have to wait in a queue until existing users leave and free up the occupied resources – this incurs a queuing delay cost. If allocated to the cloud, the user would not have to wait in a queue as the cloud has an ample amount of resources. However, this incurs a latency cost since the cloud is much further away from users compared to edge servers. The app vendor needs to decide where to allocate its users so that the system benefit is maximized. The system benefit is measured by the throughput benefit (calculated based on the number of users allocated to edge servers) minus the queuing delay and latency costs.
- **Research Problem 4** (Addressing aims of Scenario 4 – Section 1.2.4): Wireless communication is an integral part of MEC as users are to be connected to base stations, which host edge servers, through wireless communication. Several key characteristics in wireless communication are incorporated, namely multi-channel, interference, achievable data rate, and NOMA. An app vendor

needs to determine a proper channel in a suitable edge server and a sufficient amount of transmit power for each user to minimize the system cost, while satisfying several constraints such as proximity, capacity, user data rate, and base station transmit power constraints. We consider the following two scenarios:

- **Static scenario:** In this scenario, users are static (similar to Research Problems 1 and 2), the system cost consists of transmit power costs.
- **Dynamic scenario:** In this scenario, users come and go randomly over time (similar to Research Problem 3). Users allocated to an exhausted edge server will be placed in a waiting list. This incurs an allocation delay cost, which needs to be minimized along with the transmit power cost.

Our research started off by a comprehensive literature review of existing research problems in MEC. Once we had identified a gap in MEC research – edge user allocation, we commenced from a basic research problem (Research Problem 1) to establish a foundation for the EUA problem. From there, our research problems and corresponding solutions have kept evolving with more sophisticated scenarios.

The research methods employed to solve each research problem are mathematical modelling and experimental observation. Specifically, we have modelled each research problem as an optimization problem using integer or mixed-integer programming. To solve each optimization problem, we find and adopt a suitable mathematical tool, e.g., heuristic, game theory, Lyapunov optimization, etc. Our proposed approaches are then evaluated both theoretically and experimentally. They are also compared against a number of baseline and state-of-the-art approaches, if applicable.

1.4 Literature Review

In this chapter, we systematically review and summarize key related works in relation to the four research problems. The chapter is organized as follows. In Section 1.4.1, we review the studies related to the user allocation problem in general MEC systems. Section 1.4.1.1 reviews existing works that study cost-effective user allocation strategies (Research Problems 1 and 3). Section 1.4.1.2 reviews existing works that take into account user QoS and QoE (Research Problem 2). In Section 1.4.2, we review relevant user allocation methods proposed by the works that incorporate wireless communication aspects in general in MEC systems (Section 1.4.2.1), or NOMA scheme in particular in pure cellular systems (Section 1.4.2.2) (Research Problem 4).

MEC is a natural extension of cloud computing with regard to the network topology and infrastructure deployment, where the architecture is more geographically distributed compared to cloud computing. Compared to the traditional cloud computing paradigm, MEC possesses numerous unique properties, including wide-spread geographic distribution, a sizeable number of nodes, location awareness, the predominant role of wireless access, and a strong presence of streaming and real-time applications, enabling MEC to deliver a new generation of services and applications at the edge of the network. This new architecture pushes cloud computing resources closer to users. Barcelona in Spain is one of the first cities implementing MEC with many applications, including power monitoring in public spaces, event-based video streaming, traffic analysis, and connectivity on-demand [36]. There are more than 3,000 edge servers deployed across the city serving thousands of IoT devices. The sheer number of edge servers and end-devices, with the horizontal scaling nature of MEC, leads to the need for effective and efficient solutions to many different research problems faced by app vendors, including the user allocation problem.

Computation offloading [4, 10, 37] is an important research track in the field of MEC that shares some similarities with the EUA problem. Nevertheless, those two problems are differentiated by several essential characteristics. In the computation offloading problem, a user generates a series of computation tasks, which can be partly executed on its local device and edge servers (partial offloading), or completely on edge servers or remote clouds (full offloading) [38]. A computation task usually has a single-dimensional resource requirement (CPU cycles) [4, 10, 39]. On the other hand, in the user allocation problem, an app vendor needs to dedicate multiple types of resources to serve a user on an edge server [24, 40, 41]. In some works [4, 10], users are assumed to be pre-allocated to edge servers before proceeding to the task offloading phase. Furthermore, a user in the user allocation problem must be allocated to a server, either an edge server or a cloud server, instead of being able to partially offload its computation tasks, or share computation tasks among edge servers.

Cloud task scheduling is also a research topic that shares some similarities with the EUA problem. Nevertheless, from the app user's perspective, the virtual machines or clouds in the same data center share the same reachability while an app user can only access edge servers that cover it. This proximity constraint is one of the unique characteristics in the MEC system, which has been widely acknowledged in many other studies of edge computing [10, 16, 24, 29].

1.4.1 User Allocation in general MEC Systems

1.4.1.1 Cost-Effective User Allocation in MEC Systems

In an MEC system, an app vendor can deploy its applications and services on edge servers to serve their users with minimum latency. From an app vendor's perspective, the user allocation problem is the problem of how to allocate its users to proper edge servers so that its optimization objectives are satisfied. We first introduced the user allocation problem in our previous work [11]. The authors of [42] study the users-to-cloudlets allocation and cloudlet placement problems in wireless metropolitan networks. In their scenario, users are connected to access points, which might or might not have cloudlets, using a density-based clustering algorithm. The ideology of their proposed algorithm is to let cloudlets select users; whereas our ideology is to let users select edge server. Predictably, their approach would fully use all the edge servers, which is contradictory to our objective in Research Problem 1 (Chapter 2) – minimizing the number of required edge servers. The user allocation problem in [43] is more of a base-stations-to-edge-clouds allocation problem as opposed to our users-to-base-stations (edge servers) allocation problem. The authors of [44] study the service placement problem that takes into account user mobility. The users-to-edge-servers allocation is assumed to be already automatically handled by the edge infrastructure provider. The authors of [45] also tackle the problem of resource allocation for MEC, which allocates edge servers' computing resources to multiple competing services owned by different app vendors. We take the next step and address the user allocation problem, which allocates the edge server computing resources owned by a single app vendor to its users.

In [46, 47], the authors assume that each small geographical area will be covered by a single edge server, or each server covers a region exclusively with other servers. This is unlikely to happen in any practical MEC situations, where different edge servers' coverage areas might partially overlap to avoid non-service areas [10, 29]. The authors of [16] consider scenarios where users can move from one place to another, which require reallocating users among edge servers. Their user allocation approach aims at maximizing the number of allocated users while minimizing the number of reallocations. This does not help minimize the number of required servers as required by Research Problem 1. The authors of [24] propose a game-theoretic user allocation approach that minimizes system costs, measured by the amount of computing resources needed to serve users and the penalty of having unallocated users. This approach also indirectly minimizes the number of required edge servers so we have compared it with our approach in Chapter 2. In [19], the authors follow our work and attempt to solve Research Problem 1 by using a fruit fly optimization-based algorithm. In [48], a one-user-one-VM (virtual machine) scenario is addressed. This might be impractical in a MEC system since launching a VM is a time-consuming

process, which defeats the edge computing's purpose of ensuring a low-latency connection.

In [16, 23], the authors study a dynamic scenario where users come and go overtime – similar to the scenario in our Research Problem 3 (Chapter 4). They, however, do not consider the user queuing system like we do. Thus, the user queuing delay cost is not incorporated into the system cost in those studies. This is also the case for the vast majority of existing researches on the EUA problem. In [49], the authors assume that an edge server can serve a user outside its coverage via an intermediary server when that intermediary server is being overloaded in terms of computation capability (but still has sufficient communication capability to handle the user). In real-world scenarios, this assumption is not necessarily realistic. Even if it is, it is only applicable to scenarios without many users. In most real-world scenarios, due to edge servers' constrained computing resources, it is unlikely for them to serve their neighbor servers' users. Most of the time, they will be busy serving their own users (those that are covered by the edge server itself). This is why we consider a queuing system in Chapter 4. In this chapter (and Chapter 5), we employ the Lyapunov optimization framework, which has been proven to be very effective and widely applied when dealing with a time-slotted scenario [4, 44, 50–52]. Previous works that employ the standard Lyapunov optimization framework usually assume that each job or task can be completely executed within the duration of a time slot [4, 10, 52]. We apply the Lyapunov optimization in a more general situation where a user session can last longer than one time slot, which could be a few seconds or minutes in a highly stochastic MEC environment.

1.4.1.2 QoS-Aware User Allocation in MEC Systems

QoE management and QoE-aware resource allocation have long been a challenge even before the cloud computing era [53]. The authors of [54] develop a framework for resource allocation among media cloud, brokers and mobile social users that maximizes media cloud's profit and users' QoE. While the concept of brokers is similar to edge servers, there are some essential differences. In their work, the broker is simply a middleware for transferring tasks between the cloud and mobile users. Edge servers, on the other hand, are capable of processing computation tasks. The authors of [55] study the trade-off between QoE and system costs of virtual machine provisioning in a centralized video-streaming cloud environment. In the aforementioned works, QoE is measured by the processing, playback, or downloading rate. In Chapter 3 (Research Problem 2), we consider a more general scenario where QoE is measured based on QoS levels, which are represented by the required amount of computing resource.

Relevant QoE-focused problems have started gaining attraction in the field of MEC as well. The authors of [56] introduce an architecture that integrates resource-intensive computing with mobile applications

while leveraging MEC. They aim to provide a new line of personalized and QoE-aware applications. The authors of [57] and [40] solve the application placement problem in the MEC system. In their works, a user's QoE is estimated based on three levels (low, medium, and high) of access rate, processing time, and required resources. The user allocation problem that we are dealing with can be regarded as the next step after application placement. The authors of [58] address the QoE-aware computation offloading scheduling problem in mobile clouds from a networking perspective, where the energy consumption of mobile devices and latency must be considered. In contrast, in this thesis, the user allocation problem is tackled from the app vendor's perspective, who tries to allocate its own users rather than dealing with low-level computation tasks.

Besides the aforementioned literature, there are a number of works on user allocation in edge computing [11, 16, 24, 26, 29]. They, however, do not consider QoE or adjustable QoS like we do in Research Problem 2, which is a major limitation that severely lowers their performance in the scenario investigated here. [28] tackles the QoE-aware EUA problem. Nevertheless, it assumes that a user can be served by multiple edge servers, whereas a user in our problem can only be served by one edge server. In [18], the authors continue our Research Problem 2 and consider a more detailed QoE model, which incorporates the distance between users and edge servers. The authors of [17] aim to increase user QoE in scenarios where a user's request can be decomposed into multiple tasks to be processed by different edge servers.

1.4.2 User Allocation in NOMA-based MEC Systems

1.4.2.1 Wireless Communication-Aware User Allocation in MEC Systems

The vast majority of existing works on the EUA problem [11, 16, 21, 24, 28–31] do not take into account various aspects of wireless communication such as multi-channel, inter-cell and intra-cell interference, transmit power, achievable transmission data rate, etc. As wireless communication is the backbone of MEC, the aforementioned characteristics should not be neglected in user allocation, which might hold back the performance of MEC systems. In [25–27], the authors incorporate the communication aspect of an MEC system that features multiple wireless channels with interference. Nevertheless, they only consider intra-cell interference, whereas we also consider inter-cell interference, which could be very severe in a dense real-world 5G/6G networks (up to 50 per km² [12]), impacting users' data rates significantly and thus must not be neglected. Allocating users without considering the communication aspect fully can be highly uneconomical since app vendors can now access and leverage network data such as received signal, received power, throughput, neighbor cells, QoS, etc. [59]. Furthermore, none

of the existing works on EUA considers NOMA, in which interference and power control play an important role and should not be overlooked as demonstrated in our experiments in Chapter 5.

Realizing the benefits provided by NOMA, researchers are beginning to study several MEC problems under NOMA settings [60–62]. However, most of them focus on the computing offloading problem, which is an important problem that shares some similarities with the edge user allocation problem. Nevertheless, they are distinguished by several essential characteristics as discussed in Section 1.4.1.1.

1.4.2.2 User Allocation in NOMA-based Cellular Networks

The user allocation problem (also often referred to as the user association problem) is a mature and well-researched problem in conventional cellular networks [63]. However, the introduction of NOMA has sparked a new wave of research on this problem. The authors of [64] aim to maximize the sum-throughput in both downlink and uplink NOMA systems. They also demonstrate that NOMA can achieve a significant throughput gain over the traditional orthogonal multiple access (OMA) scheme. Inter-cell interference is not yet considered in [64] though. The authors of [65] confirm that carefully pairing users into channels in NOMA can offer a larger sum rate than OMA. Nevertheless, this work is limited to only two users per channel. The authors of [66, 67] also impose a limit on the number of users on a channel. The authors of [68] try to maximize the sum data rate and spectral efficiency in a multi-cell but single-channel system. The authors of [69, 70] aim to minimize the power consumption and maximize the spectral efficiency in a multi-channel single-cell NOMA scenario. If applied to a multi-cell NOMA scenario, the authors would also have to consider the inter-cell interference. The authors of [71] attempt to increase the energy efficiency in a multi-cell multi-channel NOMA system by allocating users based on a ranking of channel conditions. The authors of [72] aim to increase users' QoE by maximizing users' data rates, which will consequently result in energy inefficiency. The authors of [73] also aim to maximize the sum-rate. Whereas in most real-world scenarios, users only require a specific data rate level to ensure the QoE. From the review of existing studies, we can see that all the existing approaches will need to be remodeled or redefined when adapting to a new environment like MEC as they completely neglect the computing side of MEC (the scarcity and heterogeneity of computing resources on edge servers). Many of them even make unrealistic assumptions, e.g., a limit on the number of users on each channel [65–67], and single-cell [69, 74] or single-channel [68, 75] systems. These assumptions reduce the problem complexity but unnecessarily impede the prospects of NOMA and render their approaches impractical in real-world NOMA-based MEC systems. Our approach eliminates this limitation by jointly considering both communication and computation aspects

in an MEC system.

1.5 Overview of the Publications (Chapters) in this Thesis

As this is a *PhD by Publication*, the body of this thesis consists of eight papers stemmed from the research conducted as part of this study. They collectively solve the four research problems identified in Section 1.3. This section provides an overview of the contributions of each paper. Table 1.1 summaries those papers.

Table 1.1: List of publications resulting from this study

Ref.	Venue	Acronym	Rank ²	Published? ³	Chapter	Research Problems (RP)
[11]	16 th International Conference on Service-Oriented Computing	ICSOC	A	2018	Chapter 3.1	RP1
[30]	IEEE Transactions on Cloud Computing	TCC	Q1	2020	Chapter 3.2	RP1
[29]	17 th International Conference on Service-Oriented Computing	ICSOC	A	2019	Chapter 4.1	RP2
[31]	Future Generation Computer Systems	FGCS	A/Q1	2020	Chapter 4.2	RP2
[32]	40 th International Conference on Distributed Computing Systems	ICDCS	A	2020	Chapter 4.3	RP2
[34]	IEEE Transactions on Services Computing	TSC	A*/Q1	2021	Chapter 5	RP3
[33]	IEEE Transactions on Mobile Computing	TMC	A*/Q1	2021	Chapter 6.1	RP4
[76]	41 th IEEE International Conference on Computer Communications	INFOCOM	A*	In Review	Chapter 6.2	RP4

Paper [11] addresses Research Problem 1, which is inspired by Motivating Scenario 1 (Section 1.2.1). Since this is the first work on a new problem in MEC, this paper has received tremendous attention from the research community, resulting in a series of publications by other researchers addressing the EUA problems. This paper was also selected as the best research paper at ICSOC 2018. In this paper, we propose and formally model a new research problem. We aimed to establish a solid foundation for this EUA problem so we tried to keep it simple and concise instead of covering too many details such

²Measured as of Aug 2021 using CORE Ranking Portal (www.core.edu.au/conference-portal) and Scimago Rankings (www.scimagojr.com)

³Date of publication if applicable

as user mobility, bandwidth, delay requirements, security, etc. In paper [30], we propose an effective and efficient solution to Research Problem 1. This is an extension of [11] so [11] will not be included in this thesis to avoid repetition.

Papers [29, 31, 32] address Research Problem 2, which is inspired by Motivating Scenario 2 (Section 1.2.2). Paper [34] addresses Research Problem 3, which is inspired by Motivating Scenario 3 (Section 1.2.3). Paper [33] addresses the static scenario in Research Problem 4, which is inspired by Motivating Scenario 4 (Section 1.2.4). Paper [76] addresses the dynamic scenario in Research Problem 4, which is inspired by Motivating Scenario 4 (Section 1.2.4). Those papers mathematically formulate the aforementioned research problems and introduce various solutions that employ different techniques. All the proposed solutions are theoretically analysed and experimentally demonstrated to outperform a number of baseline and state-of-the-art approaches.

1.6 Thesis Organization

In this section, we provide an overview of the thesis along with our key contributions, and a literature review of existing work in relation to the four identified research problems. We point out a number of major limitations of the related work that make them impractical to the problems we are studying. Chapters 2, 3, 4, and 5 solve the four research problems discussed in Section 1.3. Those four chapters comprise of eight publications (summarized in Table 1.1) resulting from our study as this is a thesis by publication. In Chapter 6, we conclude this thesis and discuss the future research directions. Appendix A contains the authorship statements for all the publications used as part of this thesis.

Cost-Effective User Allocation in Mobile Edge Computing Systems

In this chapter, we establish a foundation for the EUA problem by addressing the scenario where users are static, e.g., traffic cameras, smart sensors, mobile users who do not move much, etc. An app vendor needs to allocate as many users to as fewest edge servers as possible to maximize user QoS and minimize the cost of renting resources on edge servers. We model the EUA problem as a variable-sized vector bin packing problem, which is an NP-hard problem, and propose an optimal approach to find optimal solutions to this problem using Lexicographic Goal Programming technique. To address the biggest drawback of this optimal approach – computational complexity, we propose an effective and efficient heuristic to solve this problem in large-scale scenarios. We evaluate this heuristic by a series of experiments against the optimal approach, two baseline approaches, two state-of-the-art edge user allocation approaches, and several representative approximation algorithms for solving the VSVBP problem. The proposed heuristic is demonstrated to be highly efficient and more effective than all other non-optimal approaches in comparison, being able to allocate the highest number of users using fewest edge servers in most experimental settings.

This chapter is presented in the form of our published paper [30] as **P. Lai**, Q. He, J. Grundy, F. Chen, M. Abdelrazek, J. Hosking, J. Grundy, and Y. Yang, “Cost-effective app user allocation in an edge computing environment,” *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020, doi: 10.1109/TCC.2020.3001570. ©2021 IEEE. Reprinted, with permission, from IEEE Transactions on Cloud Computing. This journal paper is an extension of our published conference paper [11] **P. Lai**, Q. He, M. Abdelrazek, F. Chen,

J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2018, pp. 230-245. ***Note that several sections in the paper included below have been removed or slightly modified to reduce repeated content that has appeared else where in this thesis. For a full, unedited version, please refer to the original paper itself.***

Cost-Effective App User Allocation in an Edge Computing Environment

Phu Lai, Qiang He, John Grundy, Feifei Chen, Mohamed Abdelrazek, John Hosking, and Yun Yang

Abstract—Edge computing is a new distributed computing paradigm extending the cloud computing paradigm, offering much lower end-to-end latency, as real-time, latency-sensitive applications can now be deployed on edge servers that are much closer to end-users than distant cloud servers. In edge computing, edge user allocation (EUA) is a critical problem for any app vendors, who need to determine which edge servers will serve which users. This is to satisfy application-specific optimization objectives, e.g., maximizing users' overall quality of experience, minimizing system costs, and so on. In this paper, we focus on the cost-effectiveness of user allocation solutions with two optimization objectives. The primary one is to maximize the number of users allocated to edge servers. The secondary one is to minimize the number of required edge servers, which subsequently reduces the operating costs for app vendors. We first model this problem as a bin packing problem and introduce an approach for finding optimal solutions. However, finding optimal solutions to the \mathcal{NP} -hard EUA problem in large-scale scenarios is intractable. Thus, we propose a heuristic to efficiently find sub-optimal solutions to large-scale EUA problems. Extensive experiments conducted on real-world data demonstrate that our heuristic can solve the EUA problem effectively and efficiently, outperforming the state-of-the-art and baseline approaches.

Index Terms—Edge computing, fog computing, user allocation, optimization, resource allocation

1 INTRODUCTION

Several introductory paragraphs have been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

In a mobile edge computing (MEC) environment, a number of edge servers are deployed by edge infrastructure providers like AT&T or Vodafone in a distributed fashion (usually near cellular base stations [2]) so that they can cover different geographical areas. Users within an edge server's coverage can connect to the edge server via LTE, 4G, or 5G [3]. Edge infrastructure providers are responsible for ensuring a low latency connection to users. A user's latency requirement is assumed to be satisfied as long as the user is allocated to an edge server. The coverage areas of adjacent edge servers usually partially overlap to avoid non-service areas – areas that are not covered by any edge server [1], [4]. A user located in the overlapping area can connect to one of the edge servers covering them (*proximity constraint*) that has sufficient computing resources (*capacity constraint*) such as CPU, storage, bandwidth, or memory. Note that from the app vendor's perspective, they need to ensure that the resources hired on an edge server will suffice to accommodate their users allocated to that edge server during the allocation process. How much of the hired resources will be used is dependent on the users at runtime.

This is an extended and revised version of a preliminary conference paper that was presented in ICSOC 2018 [1].

- P. Lai, Q. He, and Y. Yang are with the School of Software and Electrical Engineering, Swinburne University of Technology, 3122, Australia. E-mail: tlai, qhe, yyang@swin.edu.au.
- J. Grundy is with the School of Information Technology, Monash University, 3168, Australia. E-mail: john.grundy@monash.edu.
- F. Chen and M. Abdelrazek are with the School of Information Technology, Deakin University, 3125, Australia. E-mail: mohamed.abdelrazek, feifei.chen@deakin.edu.au.
- J. Hosking is with the School of Science, University of Auckland, Auckland, New Zealand. E-mail: j.hosking@auckland.ac.nz.

Compared to a cloud data center, a typical edge server has very limited computing resources [5], [6], which leads to the need for an effective resource allocation strategy.

Edge servers' capacities, current workloads, coverage, proximity to users, and the number of users to allocate can be obtained or calculated at runtime. Due to the limited resources of the edge servers, an edge server might not be able to serve all the users within its coverage. Since some of the users might also be located in other edge servers' coverage, an app vendor can allocate them to those edge servers to share the workload with the overloaded servers. Due to the aforementioned constraints, there might be some users that cannot be assigned to any edge server. Those users will be connected directly to a remote cloud server, which is not desirable. Therefore, the optimization objective here is to maximize the number of users allocated to edge servers, which ensures the QoS from the app vendor's perspective. In the meantime, the number of edge servers required to serve those users (or the number of active edge servers) needs to be minimized. This will ensure the cost-effectiveness of the allocation by cutting down the app vendor's costs of hiring edge servers to serve their app users [1], [7] and improving the resource utilization on edge servers [7], [8]. In fact, minimizing the number of required servers is one of the key objectives in the server consolidation problem in cloud computing [9], [10]. In this paper, we study quasi-static scenarios where users are relatively static, not roaming across edge servers quickly [4], [7], [11], [12], e.g., surveillance cameras, traffic sensors, mobile, or IoT users who stay in one location.

The above problem is referred to as an *edge user allocation* (EUA) problem [1], [7], [13], [14]. This problem has been modeled as a bin packing problem and proven to be \mathcal{NP} -hard in our previous work [1]. As demonstrated in [1], the EUA problem is extremely computationally ex-

pensive to solve optimally in a large-scale scenario due to the dense distribution and limited computing resources of edge servers. It takes up to 23 seconds to find an optimal solution when there are just around 512 users and 125 edge servers, which is unacceptable for applications or services that require real-time or near real-time decision making. Therefore, we introduce Most Capacity First (MCF), a heuristic approach for efficiently finding a sub-optimal solution to the EUA problem. Note that the EUA problem has a number of variants with different assumptions and objectives, depending on specific applications and services. For example, [13] aims to maximize the number of allocated users and minimize the number of user relocations when dealing with user mobility; [7] aims to maximize the number of allocated users with minimum system costs; and [14] aims to maximize users' overall quality of experience (QoE). The main contributions of this paper include:

- In our previous work [1], we formally model the EUA problem as a variable-sized vector bin packing (VSVBP) problem, which is \mathcal{NP} -hard. We then optimally solve this problem using the Lexicographic Goal Programming (LGP) technique.
- Due to the \mathcal{NP} -hardness of the problem, finding an optimal solution is intractable in a scenario that involves a great number of users and edge servers. To effectively deal with the high complexity, this paper proposes MCF – a heuristic approach for finding a sub-optimal EUA solution efficiently.
- Extensive evaluations are conducted on a real-world dataset to demonstrate the effectiveness and efficiency of the proposed approaches. The results show that our approaches outperform the state-of-the-art and baseline approaches.

The remainder of the paper is organized as follows. Section 2 motivates this research with an example. Section 3 introduces the VSVBP problem, based on which we formulate the EUA problem in Section 4. Section 5 presents our LGP-based and heuristic approaches for solving the EUA problem, which are evaluated in Section 6. Section 7 reviews the related work and Section 8 concludes this paper.

2 MOTIVATING EXAMPLE

A representative application that can leverage the low latency provided by edge computing is large-scale cloud gaming [15] - the fastest growing gaming model [16]. This model has made many online game platforms, such as Hatch¹ and Sony PlayStation Now², more accessible to thin-client mobile players since the resource-expensive game application instances are running on powerful servers in the remote cloud. Let us consider an increasingly popular virtual reality game, which requires a great amount of computing power for graphics rendering. The centralized cloud model allows mobile devices to offload heavy computation tasks to the servers in the cloud. However, this approach often introduces significant network delays because of the long distance between the players and the remote cloud servers.

1. www.hatch.live/
2. www.playstation.com/en-gb/explore/playstation-now/

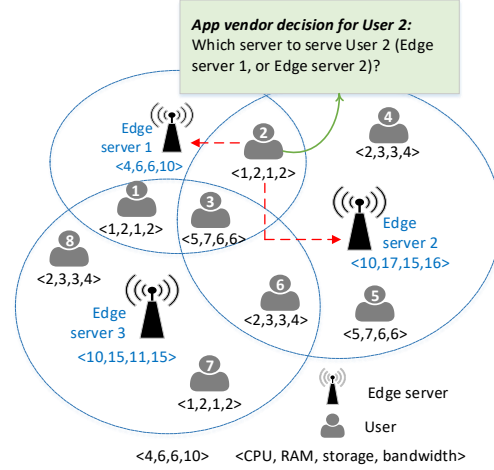


Fig. 1: Example edge computing environment

Therefore, pushing computing power closer to players with edge computing is a promising solution to this problem.

Fig. 1 shows a small example of an MEC environment with 8 players u_1, \dots, u_8 and 3 edge servers, s_1, \dots, s_3 . Each edge server covers a particular geographical area and has a specific amount of computing resources (CPU, RAM, storage, and bandwidth). An edge server can only serve players within its coverage (proximity constraints). For example, player u_1 can be allocated to either edge server s_1 or s_3 only. Edge server s_2 cannot serve player u_1 since player u_1 is outside its coverage area. In an edge computing environment, the computing resources of an edge server, such as CPU, RAM, bandwidth, etc., are usually shared by multiple users. Thus, in addition to the proximity constraints, the game vendor needs to take into account various capacity constraints, i.e., whether the available computing resources on an edge server suffice to serve the players allocated to it. In Fig. 1, each edge server has a limited amount of computing resources, denoted as a vector $(CPU, RAM, storage, bandwidth)$. Different players might require different amounts of computing resources. For instance, edge servers that serve players who select higher graphics settings (4K graphics quality, spatial anti-aliasing, etc.) consume more resources to facilitate high-quality graphics rendering. The total resource requirements of players allocated to an edge server must not exceed the available computing resources on that edge server. There are five players located within the coverage of edge server s_3 , whose aggregated resource requirement is $(11, 17, 14, 16)$, which exceeds the available resources of server s_3 $((10, 15, 11, 15))$. Thus, the game vendor needs to allocate some of those players to other edge servers. The objective is to allocate as many players to as few edge servers as possible.

A possible solution to the EUA problem in Fig. 1 is to allocate player u_1 to server s_1 , players u_2, u_3, u_4 , and u_6

to server s_2 , and players u_7 , and u_8 to server s_3 . Player u_4 cannot be allocated to server s_2 since the server is already serving other players and no longer has sufficient resources to serve player u_4 . Neither proximity nor resource constraints are violated in this way. However, this is not the optimal solution in terms of QoS or cost-effectiveness since only seven out of eight players are allocated to edge servers, and all three edge servers are required. A better solution would be to allocate players u_2 , u_4 , u_5 , and u_6 to server s_2 , and players u_1 , u_3 , u_7 , and u_8 to server s_3 . This allocation allows all eight users to be served by edge servers; furthermore, it does not require edge server s_1 at all.

Finding an optimal solution to the EUA problem is not trivial, especially in a large-scale scenario with numerous users covered by numerous edge servers. Fig. 1 is a very small-scale example. In a real-world EUA scenario, there would be many more players (or app users in general) and edge servers, resulting in a much larger solution space for the EUA problem. Assuming that there are n users and m edge servers, the solution space may consist of up to m^n possible solutions. Thus, an effective and efficient user allocation approach is needed for app vendors in such scenarios.

3 BACKGROUND

Definition 1. Classic Bin Packing (BP) Problem. Given an infinite supply of identical bins $S = \{s_1, s_2, \dots\}$ with capacity C and a set of n items $U = \{u_1, u_2, \dots, u_n\}$. Let a value w_j be the size of item u_j that satisfies $0 < w_j \leq C, \forall u_j \in U$. The objective is to pack as many items as possible into the bins such that the total item size in each bin must not exceed the bin capacity C : $\sum_{u_j \in U_{s_i}} w_j \leq C, \forall s_i \in S$, where U_{s_i} is the set of items placed in bin s_i .

The only constraint in the classic BP problem is that the total size of the items packed into a bin must not exceed the capacity of the bin. This combinatorial optimization problem is known to be an \mathcal{NP} -hard problem [17].

Definition 2. Variable-Sized Bin Packing (VSBP) Problem. Given a limited collection of k bin sizes such that $1 = \text{size}(B^1) > \text{size}(B^2) > \dots > \text{size}(B^k) > 0$, there is an infinite supply of bins for each bin type B^l , where $l = 1, \dots, k$. Let $S = \{s_1, s_2, \dots, s_m\}$ be the list of bins needed for packing all items. Given a list of n items $U = \{u_1, u_2, \dots, u_n\}$, each with item size $w_j \in (0, 1]$, where $j = 1, \dots, n$, the aim of the VSBP problem is to pack all items into bins so that the total size of items in each bin must not exceed the bin capacity, and the total size of the required bins $\sum_{s_i \in S} \text{size}(s_i)$ is the minimum.

In the classic BP problem, all bins are homogeneous with the same bin capacity. VSBP is a more general variant of the classic BP in which a limited collection of bin sizes is allowed. VSBP aims at minimizing the total size of the bins used, which is different compared to the aim of the classic BP problem.

Definition 3. Vector Bin Packing (VBP) Problem. Given a set of n items $U = \{u_1, u_2, \dots, u_n\}$, the size of item $u_j \in U$ is denoted as d -dimensional vector $w_j = \langle w_j^1, w_j^2, \dots, w_j^d \rangle$, where $w_j \in [0, 1]^d$. One is given an infinite supply of identical bins $S = \{s_1, s_2, \dots\}$ with capacity $C = \langle 1^1, 1^2, \dots, 1^d \rangle$. The aim is to pack the set U into a minimum number of bins $s_i \in S$ such

that $\|\sum_{u_j \in U_{s_i}} w_j\|_\infty \leq 1, \forall s_i \in S$, where U_{s_i} is the set of all items packed in bin s_i .

In the classic BP problem, the size of an item is presented as a single aggregation measure. In contrast, the size of an item in the VBP problem is associated with a multi-dimensional vector. The aim remains similar, in which the sum of packed item size vectors must not exceed the bin capacity vector in each dimension, which is normalized to 1 without loss of generality. The VBP problem is also known as the multi-capacity BP problem in some literature [18], [19].

EUA Problem. In the EUA problem, each edge server corresponds to a bin with its available computing resources being the bin capacity. An app user corresponds to an item, whose computing resource requirement corresponds to the size of that item.

In this paper, we tackle the EUA problem from the app vendor's perspective. Since different users require various amounts of different computing resources [7], [20], [21], the amount of computing resources required by a user should be presented as a d -dimensional vector where each dimension represents a resource type (CPU, RAM, storage, etc.) instead of a single aggregate measure. This is also applied to the amount of computing resources available on edge servers. Therefore, the EUA problem can be modeled as a mixture of the VSBP problem and the VBP problem, hence being a variable-sized vector bin packing (VSVBP) problem. The EUA problem is \mathcal{NP} -hard since the classic BP problem, which is \mathcal{NP} -hard [17], is a special case of the VSVBP problem where all bins' sizes and all resources' requirement of users are identical.

4 PROBLEM FORMULATION

Given a number of app users belong to a single app vendor using the same application, our objective is to maximize the total number of allocated app users and minimize the number of edge servers required to serve those users. We first introduce the relevant notations used in our model in Table 1. In the EUA problem, every user covered by any edge server must be allocated to an edge server unless all the servers accessible to the user have reached their maximum capacities. 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: Notations

Notation	Description
$\mathcal{S} = \{s_1, s_2, \dots, s_m\}$	a finite set of edge server s_i , where $i = 1, 2, \dots, m$.
$\mathcal{D} = \{CPU, RAM, storage, bandwidth\}$	a set of computing resource types.
$C_i = \langle C_i^1, C_i^2, \dots, C_i^d \rangle$	the capacity of an edge server s_i . C_i is a d -dimensional vector with each dimension $C_i^k, k = 1, \dots, d$, representing the available amount of resource type $k \in \mathcal{D}$ on edge server s_i .
$\mathcal{U} = \{u_1, u_2, \dots, u_n\}$	a finite set of user u_j , where $j = 1, 2, \dots, n$.
$w_j = \langle w_j^1, w_j^2, \dots, w_j^d \rangle$	d -dimensional vector representing the computing resource requirement of user u_j . Each vector component $w_j^k, k = 1, \dots, d$ represents the amount of resource type k needed by user u_j on an edge server.
$\mathcal{U}_{s_i} \subset \mathcal{U}$	a set of users allocated to edge server s_i .
$\mathcal{S}_{u_j} \subset \mathcal{S}$	a set of user u_j 's neighbor edge servers, i.e., edge servers that have user u_j in their coverage areas.
$cov(s_i)$	the coverage of edge server s_i .

In the classic BP problem, an item can be placed in any bin as long as the bin has sufficient remaining capacity. However, in our problem, each edge server covers a limited geographical region. Thus, an item (user) can only be placed in one of several specific bins (edge servers). A user u_j can be allocated to an edge server s_i only if it is located in server s_i 's coverage area:

$$u_j \in cov(s_i), \forall u_j \in \mathcal{U}; \forall s_i \in \mathcal{S} \quad (1)$$

and the total resource requirements of all users allocated to an edge server must not exceed its available computing resources (Constraint (2)). Otherwise, the server will be overloaded, causing service disruptions or performance degradation. Take Fig. 1 for instance, the aggregated resource requirements of users u_1, u_2 , and u_3 are $\langle 7, 11, 8, 10 \rangle$, which are greater than the available computing resources of edge server s_1 ($\langle 4, 6, 6, 10 \rangle$). Thus, allocating all those three users to this edge server violates the capacity constraint.

$$\sum_{u_j \in \mathcal{U}_{s_i}} w_j \preceq C_i, \forall s_i \in \mathcal{S} \quad (2)$$

Our primary objective is to maximize the number of users allocated to edge servers, which ensures the quality of service from the app vendor's perspective:

$$\text{maximize} \sum_{s_i \in \mathcal{S}} |\mathcal{U}_{s_i}| \quad (3)$$

Our secondary objective is to find a users-to-edge-servers allocation such that the number of required edge servers is minimum:

$$\text{minimize} \left| \left\{ s_i \in \mathcal{S} \mid \sum_{u_j \in \mathcal{U}_{s_i}} w_j > 0 \right\} \right| \quad (4)$$

5 APPROACHES

5.1 Lexicographic Goal Programming-based Approach

In this paper, we address the EUA problem with two optimization objectives: 1) *maximizing the number of allocated*

users and 2) *minimizing the number of required edge servers*, while satisfying the *proximity constraint* and *capacity constraint*. Accordingly, we solve the EUA problem with the LGP-based technique [22]. With the LGP technique, multiple optimization objectives are ranked by their levels of importance, or priorities. The problem solver will attempt to find an optimal solution that satisfies the primary objective and then proceed to find a solution for the next objective without deteriorating the previous objective(s). An LGP program can be solved as a series of connected linear programs. The LGP formulation of the EUA problem is as follows:

$$\text{maximize} \sum_{j=1}^n \sum_{i=1}^m x_{ij} \quad (5)$$

$$\text{minimize} \sum_{i=1}^m y_i \quad (6)$$

$$\text{subject to: } x_{ij} = 0 \quad \forall i, j \in \{i, j \mid u_j \notin cov(s_i)\} \quad (7)$$

$$\sum_{j=1}^n w_j^k x_{ij} \leq C_i^k y_i \quad \forall i \in \{1, \dots, m\}; \forall k \in \{1, \dots, d\} \quad (8)$$

$$\sum_{i=1}^m x_{ij} \leq 1 \quad \forall j \in \{1, \dots, n\} \quad (9)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in \{1, \dots, m\}; \forall j \in \{1, \dots, n\} \quad (10)$$

$$y_i \in \{0, 1\} \quad \forall i \in \{1, \dots, m\} \quad (11)$$

x_{ij} and y_i are binary indicator variables such that

$$x_{ij} = \begin{cases} 1, & \text{if user } u_j \text{ is allocated to edge server } s_i. \\ 0, & \text{otherwise.} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if edge server } s_i \text{ is used to serve users.} \\ 0, & \text{otherwise.} \end{cases}$$

Objective (5) maximizes the number of app users allocated to edge servers. Objective (6) minimizes the number of edge servers required to serve allocated users. Here, objective (5) is prioritized over objective (6). There are two groups of binary variables, i.e., x_{ij} (10) and y_i (11).

Proximity constraint (7): An edge server cannot serve users who are located outside its coverage area. An app user may be in the overlapping coverage area of multiple edge servers. For instance, user u_1 can be allocated to either edge server s_1 or s_3 in Fig. 1.

Capacity constraint (8): Each edge server s_i has an available resource capacity of $C_i = \langle C_i^1, C_i^2, \dots, C_i^d \rangle$, a d -dimensional vector. The aggregated resource requirements of each resource type of all users allocated to an edge server must not exceed its available capacity. Take Fig. 1 for example, allocating users u_1 and u_2 to server s_1 is valid since $\langle 2, 4, 2, 4 \rangle \preceq \langle 4, 6, 6, 10 \rangle$.

Constraint family (9) ensures that every app user is allocated to at most one edge server. In other words, a user can be allocated to either an edge server or the app vendor's remote cloud server.

We can find an optimal solution to a small-scale instance of the EUA problem above with an Integer Programming

solver, e.g., Gurobi³ or IBM ILOG CPLEX⁴. In our experiments, we use IBM ILOG CPLEX Optimizer solver V12.8.0 (ilog.cplex⁵ package in Java).

5.2 Most-Capacity-First Heuristic

Due to the \mathcal{NP} -hardness of the problem, finding an optimal solution will take a very long time in large-scale scenarios. This will be demonstrated in our experimental results presented in Section 6. Thus, to help app vendors allocate their users in large-scale scenarios, this section proposes MCF (MOST CAPACITY FIRST), an effective and efficient heuristic for finding a near-optimal solution to the EUA problem.

Heuristic 1 Most Capacity First (MCF)

```

1: initialization:
2: a set of edge servers  $\mathcal{S}$  and a set of users  $\mathcal{U}$ 
3: all users  $u_j, \forall u_j \in \mathcal{U}$ , are unallocated
4: end initialization
5: sort  $\mathcal{U}$  in ascending order of computing resource requirements (i.e., low-demanding users are prioritized, being the first to be allocated)
6: for each user  $u_j \in \mathcal{U}$  do
7:    $\mathcal{S}_{u_j} \triangleq$  user  $u_j$ 's neighbor edge servers;
8:    $\mathcal{S}_{u_j}^{active} \triangleq$  user  $u_j$ 's active neighbor edge servers;
9:   if  $\mathcal{S}_{u_j}^{active} \neq \emptyset$  then
10:    allocate user  $u_j$  to an edge server  $s_i \in \mathcal{S}_{u_j}^{active}$  which has the most available capacity
11:   else
12:    allocate user  $u_j$  to an edge server  $s_i \in \mathcal{S}_{u_j}$  which has the most available capacity
13:   end if
14:   if  $s_i$  cannot be decided then
15:    allocate user  $u_j$  to the central cloud server
16:   end if
17: end for

```

Given a set of edge servers \mathcal{S} and a set of users \mathcal{U} (lines 1-4), MCF efficiently and effectively allocates an app vendor's users to edge servers. Initially, all users are unallocated. In order to maximize the number of users allocated, MCF first sorts the set of users \mathcal{U} in ascending order of users' computing resource requirements (line 5). The computing resource requirement of a user is a multi-dimensional vector with each component being a resource type. Since different resource types (CPU, RAM, storage, or bandwidth) have different scales, we can sort the computing resource requirements by normalizing all resource types' requirements by maximum norm (assuming all resource types are equally important), then calculate the Euclidean norm over those resource types. App vendors can also apply other methods [23] as they see fit to sort users or servers by the amount of computing resources.

Next, MCF allocates users one by one (lines 6-14) in the order of their appearance in the list \mathcal{U} sorted above. Since the list of users has been sorted in ascending order of resource

requirements, users who have lower requirements will be allocated before users who have higher requirements. This helps maximize the number of users allocated because allocating high-demanding users first would exhaust edge servers' resources rather quickly. For each user, MCF retrieves the set of its neighbor edge servers (servers that have the user in their proximity - line 7) and the set of its active neighbor edge servers (servers that have already been used to serve users, or are serving users - line 8). To minimize the number of required edge servers, MCF attempts to allocate the user to one of the active servers first (line 9), instead of using a new edge server. Out of all active edge servers, one with the most available capacity will be chosen to serve the user (line 10). In this way, it will be most likely to have sufficient capacity to accommodate other users later on. If there is no current active server (line 11), MCF will allocate the user to the neighbor edge server which has the most available capacity (line 12). Note that allocating a user to an edge server must not violate the capacity constraint in any case. If all neighbor edge servers of the user have reached their capacity, the user will be directly connected to the app vendor's central cloud server (lines 14-16), which is not desirable. MCF completes once all users have been attended to.

Time Complexity Analysis. As discussed at the end of Section 3, the EUA problem is \mathcal{NP} -hard. It is intractable to find optimal solutions to large-scale EUA scenarios that involve a large number of app users and/or edge servers. MCF is a practical option in such scenarios for its high efficiency. Here, we analyze the worst-case time complexity of the proposed heuristic (MCF - Heuristic 1). The running time of MCF consists of: 1) iterating through all n users (Line 6), which costs $\mathcal{O}(n)$, and 2) sorting a maximum of m edge servers to identify the edge server that has the most available capacity (Line 10 or 12), which costs $\mathcal{O}(m \log(m))$. Therefore, the time complexity of this block (Lines 7-13) is $\mathcal{O}(nm \log(m))$, which is generally more complex than Line 5. Hence, the overall time complexity of MCF is $\mathcal{O}(nm \log(m))$. Note that m is the worst-case scenario; in practice, each user is covered by a much smaller number of edge servers or base stations (i.e., 32 base stations in theory as specified by ETSI [24], or 15 base stations according to the dataset used in our experiments in Section 6). As experimentally demonstrated in Section 6.3.2, the running time of MCF is around 1-2 ms in all scenarios.

Given the low running time of MCF, app vendors are able to continuously run it to react to user movement since it allocates app users one-by-one to edge servers. Specifically, when a user moves out of the coverage area of the edge server serving it, it will be disconnected from the edge server; the occupied computing resources on that edge server will be released; then MCF will consider it as a new user and allocate it to a new edge server following the rules defined in Heuristic 1, Lines 6-14. This is possible as long as switching users from one edge server to another does not incur extra costs, or if the extra costs are trivial. Such extra costs may include the migration cost (e.g., the cost of moving user data across edge servers) or reconfiguration cost (e.g., some services might require reconfiguration to serve new users) [25]. In some use cases, those extra costs could be fairly trivial. Take video streaming for example

3. www.gurobi.com/

4. www.ibm.com/analytics/cplex-optimizer/

5. www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.cplex.help/refjavacplex/html/ilog/cplex/package-summary.html

[26], where videos encoded with different resolutions are cached on edge servers so that a user can access them with low latency. Switching the user across edge servers would only require a small amount of data to be transferred, e.g., which video the user is watching, the resolution of the video, and the position in the video where the user left off. For applications and services where the extra costs are considerable, the new costs will need to be modeled and integrated into the objective functions. The heuristic will thus need to be modified accordingly.

6 EXPERIMENTAL EVALUATION

We have performed a series of experiments on a real-world dataset to evaluate the performance of our approaches against other baseline and state-of-the-art approaches.

6.1 Performance Benchmark

Our LGP-based approach (Section 5.1), referred to as Optimal hereafter, and MCF heuristic (Section 5.2) are compared to four representative approaches, namely a Greedy baseline, a Random baseline, and two state-of-the-art approaches for solving the EUA problem:

- *Greedy*: This approach allocates each user to the edge server with the most available capacity, regardless of the server’s active status. Users are allocated in no particular order.
- *Random*: This approach allocates each user to a random edge server available. Users are allocated in no particular order.
- *ICSOC19*: [14] proposes two approaches to solve the user allocation problem so that the total users’ QoS is maximized. The QoS level, or the computing resource requirement, of each user can be flexibly adjusted. We solve a slightly different problem where each user has a fixed QoS level. Their proposed Integer Programming-based approach will be used in our experimental evaluation.
- *TPDS20*: [7] proposes a game-theoretic approach to solve the user allocation problem with the objective of maximizing the number of allocated users and minimizing system costs, measured by the amount of computing resources needed to serve users and the penalty of having unallocated users.

ICSOC19 and TPDS20 are chosen as the benchmark state-of-the-art approaches as they solve the same problem as ours – user allocation, and their objectives indirectly imply the maximization of the number of allocated users. TPDS20 also implies the minimization of the number of required edge servers. All experiments are written in Java and conducted on a Windows machine equipped with Intel Core i5-7400T processor (4 CPUs, 2.4GHz) and 8GB RAM.

6.2 Experimental Settings

The experiments are conducted on the EUA dataset⁶ [1], which contains the geographical locations of end-users and

6. www.github.com/swinedge/eua-dataset

all cellular base stations in Australia. This dataset was also used in [14] and [7] to evaluate ICSOC19 and TPDS20.

Edge servers: To capture the characteristics of a 5G environment [27], we simulate a highly dense urban area of 1.8 km² covered by 125 base stations, each equipped with an edge server. The coverage radius of each edge server is randomly generated within 100-150m. The initial capacities of edge servers are randomly generated by following a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the average capacity of each resource type in \mathcal{D} , and the standard deviation $\sigma = 10$ for all conducted experiments. Since a normal distribution might contain negative numbers, any negative amount of computing resources generated is rounded up to 1.

Edge users: We assume that there are three possible types of resource requirements, $w_j \in \{(1, 2, 1, 2), (2, 3, 3, 4), (5, 7, 6, 6)\}$, $\forall u_j \in \mathcal{U}$, and $\mathcal{D} = \{CPU, RAM, storage, bandwidth\}$. The computing resource requirement of each user is uniformly randomly selected. We select those three resource requirement levels as representative in our experiments since we have conducted experiments with other types of resource requirement settings, where the difference between resource requirements is large (e.g., $\{(1, 1, 1, 1), (4, 5, 6, 6), (7, 9, 10, 7)\}$), or where the resource requirements are identical, or skewed (e.g., $\{(3, 2, 3, 2), (2, 3, 2, 3), (1, 2, 1, 3)\}$), which all returned similar results (almost identical, or just marginally different).

To comprehensively analyze the performance of our approaches in various EUA scenarios, we conduct a series of experiments with different varying parameters, including the number of users, the number of edge servers, and edge servers’ capacities (μ as defined above). Table 2 summarizes the settings of the experiments, which will be discussed in the next section. Each experiment is repeated 100 times to obtain 100 different user distributions, and the results are then averaged. This allows extreme cases, such as overly dense or sparse user/server distributions, to be neutralized. To evaluate the performance of the approaches in achieving the optimization objective, we compare the number of allocated users and required edge servers achieved by the six approaches, and also the average number of users allocated per required edge server. The efficiency will also be evaluated by the CPU time taken to solve the problem.

TABLE 2: Experimental Settings

	Users	Edge servers	Available resources (μ)
Set #1	100, ..., 1000	50%	35
Set #2	500	10%, ..., 100%	35
Set #3	500	50%	30, 35, ..., 75

6.3 Experimental Results

Figs. 2, 3, and 4 demonstrate the effectiveness of all the approaches in experiment Sets #1, #2, and #3, respectively, in terms of 1) the percentage of users allocated, the higher the better (in sub-figures (c)), 2) the percentage of edge servers needed to serve those users, the lower the better (in sub-figures (b)), and 3) we additionally measure the average number of users allocated per required edge server, the

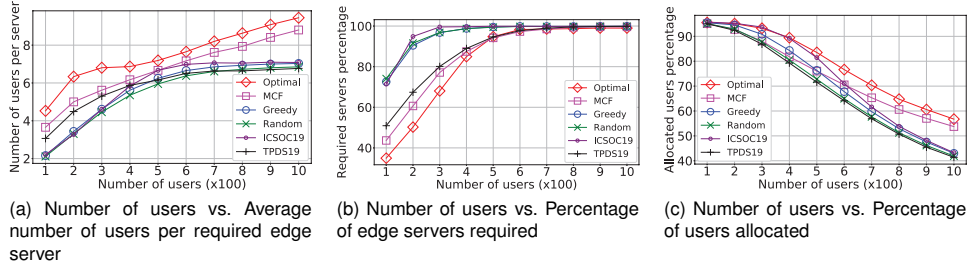


Fig. 2: Experimental results of experiment Set #1 (varying number of users)

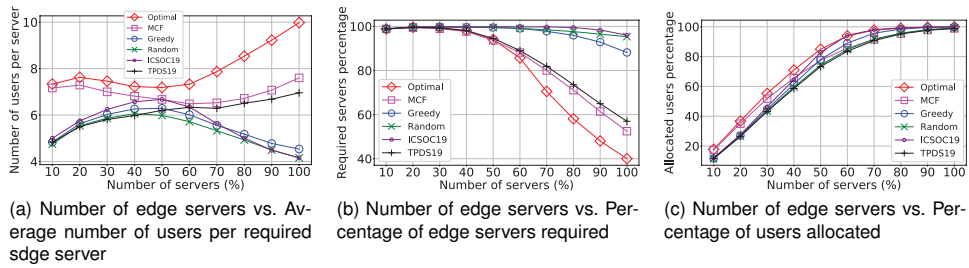


Fig. 3: Experimental results of experiment Set #2 (varying number of edge servers)

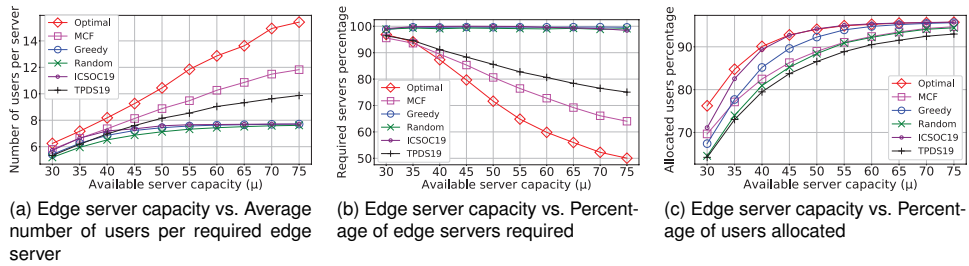


Fig. 4: Experimental results of experiment Set #3 (varying edge server' capacity)

higher the better (in sub-figures (a)). In general, Optimal, being the LGP-based approach for finding optimal solutions, clearly achieves the best performance compared to all other approaches across all experiments – being able to allocate the most number of users to the fewest number of edge servers. This comes at the cost of its very high computational overhead and it is thus inapplicable in large-scale scenarios, where low latency is critical. MCF outperforms all other baseline and state-of-the-art approaches. The efficiency of MCF is demonstrated in Figs. 5 and 6, measured by its computation time.

6.3.1 Effectiveness

Experiment Set #1. In this set of experiments, the number of users varies from 100 to 1,000 in steps of 100. The number of edge servers is fixed at 50% of all edge servers in the simulated area. In Fig. 2a, as the number of users increases, the average number of users allocated per required edge server achieved by Optimal also increases, closely followed by MCF. While the average number of users per required edge server achieved by Optimal and MCF increases linearly, the performance of the other four approaches starts to plateau at some point, e.g., 600 users in our experiments. This depicts the ineffectiveness of the other approaches in large-scale scenarios with a great number of users. With regard to the number of edge servers required

(Fig. 2b), the approaches are divided into two groups based on their performance. The first group, which includes IC-SOC19, Greedy, and Random, requires far more edge servers than the second group, which consists of Optimal, MCF, and TPDS20. Optimal requires the fewest number of edge servers, followed by MCF, then TPDS20. From 600 users onwards, every approach requires 100% number of edge servers since there is a large number of users now. Thus, all available computing resources need to be utilized. Fig. 2c shows the percentage of users allocated. We can observe a decreasing trend here. Since the amount of computing resources is fixed, introducing more users will increase the number of users who cannot be allocated to any edge server. From 100 to 500 users, MCF allocates slightly fewer users than Greedy and IC-SOC19. However, in those cases, Greedy and IC-SOC19 require far more edge servers to serve those users, which results in a lower average number of users per required server in general as shown in Fig. 2a. Other than those cases, MCF allocates considerably more users than Greedy, Random, IC-SOC19, and TPDS20. IC-SOC19 performs poorly (almost as bad as Greedy and Random) because it aims to maximize all users' overall QoE, which in turn needs to consume as much computing resources as possible, hence a high number of required edge servers. In the meantime, IC-SOC19 is also very computationally expensive since it aims to find the optimal solution to an \mathcal{NP} -hard problem. Its low effectiveness and low efficiency can be observed in other experiment sets as well. While being suitable for maximizing users' overall QoE, we can see that IC-SOC19 is not effective in minimizing the number or required edge servers.

Experiment Set #2. In this experiment set, we vary the number of edge servers available to serve users, from 10% to 100% in steps of 10% (Fig. 3). As more edge servers become available, more computing resources are available to serve users, which eventually increases the number of allocated users (Fig. 3c). In this aspect, the difference between all the approaches is marginal. However, when it comes to the number of edge servers needed (Fig. 3b), Optimal, MCF, and TPDS20 significantly outperform other approaches, requiring much fewer edge servers to serve users. IC-SOC19, Greedy, and Random require almost all edge servers in all settings. Meanwhile, the percentage of edge servers required by Optimal, MCF, and TPDS20 rapidly decreases as the number of available edge servers increases. This demonstrates the effectiveness of those approaches in utilizing the given resources. Overall, the average number of allocated users per required server (Fig. 3a) is higher for Optimal and MCF (closely followed by TPDS20 from 40% onwards). Greedy, Random, and IC-SOC19 fail to utilize the increasing number of edge servers, hence the downward trend. In some cases (20% - 40% for Optimal, 20% - 60% for MCF), the average number of allocated users per required edge server decreases because the cost of using edge servers (the number of required edge servers) outweighs the benefit of serving more users. Regardless, Optimal and MCF still beat other approaches.

Experiment Set #3. In this experiment set, we vary the amount of average available computing resources on each server. Similar to experiment Set #2, increasing edge servers' capacities eventually increases the total number of

allocated users (Fig. 4c) and the average number of allocated users per required server (Fig. 4a). It also generates more room for resource utilization. In Fig. 4b, Optimal, MCF, and TPDS20 demonstrate the ability to utilize the given computing resources by the decreasing percentage of required edge servers. MCF, again, requires fewer edge servers than TPDS20. When μ is increasing, the capacity of each edge server becomes increasingly redundant, rendering an increasing number of "unnecessary" edge servers, hence the decrease in the percentage of required edge servers.

6.3.2 Efficiency

Fig. 5 depicts the efficiency of all the approaches, measured by the average CPU execution time taken to solve an instance of the EUA problem. Optimal is the most inefficient approach that might take up to 50 seconds to find an optimal user allocation solution. The elapsed CPU time of Optimal increases considerably as we increase the size of the EUA problem by adding more users (Set #1, Fig. 5a), more edge servers (Set #2, Fig. 5b), and more edge server capacity (Set #3, Fig. 5c). When it reaches a threshold, the elapsed CPU time of Optimal decreases at a slower rate than it increases. This threshold is determined by the number of users to be allocated and the amount of computing resources (available edge server capacity or available edge servers). To be specific, in Fig. 5a, the CPU time starts to decrease at 500 users. This happens because when the number of users exceeds 500, a newly generated user tends to be positioned at the exact location of an existing user. Thus, the IBM CPLEX solver can base its decisions to be made for the new user on the decisions made for the existing user. As a result, we can see the elapsed CPU time is almost symmetrical around that threshold (500 users). In Figs. 5b and 5c, the elapsed CPU time decreases from 80% of the total number of edge servers and from $\mu = 45$ onward, respectively. This occurs because after those thresholds, the whole MEC system is likely to have sufficient computing resources to serve a greater number of users without having to consider many other possible allocation solutions, hence less time required to decide an optimal solution.

Due to the extreme inefficiency of Optimal and IC-SOC19, Fig. 5 does not fully demonstrate the efficiency of other approaches. Therefore, Optimal and IC-SOC19 have been excluded in Fig. 6 so that the efficiency of other approaches can be visualized better. In all experiment sets (Figs. 6a, 6b, and 6c), TPDS20 is two to three orders of magnitude slower than other approaches, and its execution time increases roughly linearly with the increases in any experimental parameters. The rationale of this lies in the algorithm design of TPDS20, which is an iterative mechanism that might involve hundreds of iterations depending on the scale of the problem. MCF, on the other hand, only takes around 1-2 ms to solve the problem in any experimental setting. This demonstrates the scalability of MCF.

Generally speaking, MCF has been demonstrated to be highly effective and efficient, outperforming all the baseline and state-of-the-art approaches. TPDS20, while being suitable for solving that particular problem studied in [7], is not suitable when it comes to solving the EUA problem introduced in this paper. After all, game-theoretic approaches

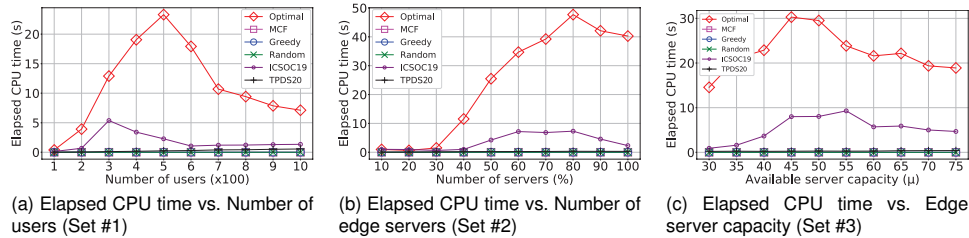


Fig. 5: CPU time consumption in experiment Sets #1, #2, and #3

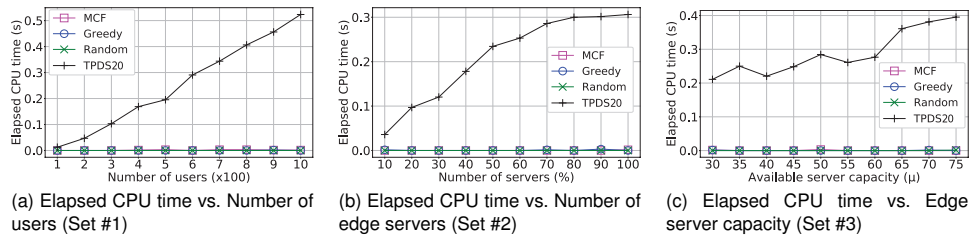


Fig. 6: CPU time consumption in experiment Sets #1, #2, and #3 without Optimal and ICSOC19

have not been seen in the current literature as a method for solving the bin packing problem.

6.3.3 Statistical Analysis

In this paper, each experiment is repeated 100 times to obtain 100 different random user distributions. To determine whether the approaches in comparison (Optimal, MCF, Greedy, Random, ICSOC19, and TPDS20) are really different when applied to all possible user distributions (of the relevant size), we perform one-sided Wilcoxon signed-rank tests [28] with a significance level of 0.01 on all three experiment sets. The detailed statistical results can be found in Appendix A of the supplemental file. In summary, the Wilcoxon signed-rank tests indicate that the difference between the effectiveness of MCF and other baseline approaches under the majority of experimental settings are statistically significant at the $\alpha = 0.01$ level. The statistical results are in line with the experimental results discussed above. With regard to the efficiency, the execution time of MCF is consistent at around 1-2 ms so we do not perform a statistical test for that.

6.4 Threats to Validity

This section has been removed since there is no close connection with the context of the thesis. Please refer to the original paper online for a full, unedited version.

7 RELATED WORK

Edge computing is a natural extension of cloud computing with regard to the network topology and infrastructure

deployment, where the architecture is more geographically distributed compared to cloud computing. This new architecture pushes cloud computing resources closer to end-users. Barcelona in Spain is one of the first cities implementing edge computing with many applications, including power monitoring in public spaces, event-based video streaming, traffic analysis, and connectivity on-demand [29]. There are more than 3,000 edge servers deployed across the city serving thousands of IoT devices. The sheer number of edge servers and end-devices, with the horizontal scaling nature of edge computing, leads to the need for effective and efficient solutions to many different research problems faced by app vendors, including the user allocation problem.

7.1 User Allocation

The literature review has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

7.2 Bin Packing

Bin Packing is a classic combinatorial \mathcal{NP} -hard optimization problem widely applied in different disciplines such as operation research, or computer science (multiprocessor scheduling [30], cloud task and resource allocation [31], [32]). There are many variations of this problem, e.g., packing by weight or by cost [33], multi-dimensional BP [34], dynamic BP [35], and a lot more variants with different modified conditions, constraints, and assumptions to model different problems. The latest attempt at exactly solving the BP problem involves constraint programming. Shaw [36]

proposes a new dedicated constraint based on a set of pruning and propagation rules, which is later on implemented in IBM CPLEX [37], which is the tool used in this paper to search for optimal solutions to EUA problems.

Along with exact algorithms, various approximation algorithms have also been introduced to solve different variants of the classic BP problem. Most of the proposed approximation algorithms are designed for different special cases or variants of the classic BP problem, such as splittable, small items (relative to the size of a bin) [38], variable-sized items with identical bins [23], minimizing the total used bins load with only two bin sizes [39]. Surveys on approximation algorithms for BP problems can be found in [37], [40]. BP is a straightforward problem so any improvement in online solutions would require a special assumption as mentioned above. The authors of [41] state that the theoretical analysis of variable-sized BP remains open even in the case of only three different bin sizes. In an MEC environment, edge servers are most likely to have more than three different sizes (capacities). Therefore, to properly evaluate our approaches, we also compare our approaches with some representative approximation algorithms for solving the BP problem [41], including First Fit, First Fit Decreasing, First Fit Increasing, Best Fit, Best Fit Decreasing, Best Fit Increasing (Fig. 7). Next Fit and Next Fit Decreasing are not suitable for the EUA problem since it allows only one open bin at all times. Our proposed MCF is essentially a variant of Worst Fit Increasing, which prioritizes already-active edge servers and is adapted to multi-dimensional computing resource requirements.

As demonstrated by Figs. 7a and 7c, the increasing variants, which allocate users with low resource requirements first, outperform other approaches, especially the decreasing variants. This highlights the importance of the order of users being allocated. In all experimental settings, especially Set #2 (Fig. 7b), MCF outperforms all other approaches in comparison. More experimental results and statistical tests can be found in Appendix B of the supplemental file. [42] models the machine reassignment problem also as a VSVBP problem. Its proposed heuristic is a generalization of First Fit Decreasing and Best Fit Decreasing, which have been shown above to be not suitable for the EUA problem.

8 CONCLUSION AND FUTURE WORK

Further complementing the conventional cloud computing, edge computing is a promising distributed computing architecture that is expected to deliver many new genres of services and applications, especially those that require low-latency connection and real-time decision making. This comes with many new challenges of which user allocation is one of them. When an edge computing environment scales up, this problem becomes intractable to solve in an efficient manner due to its \mathcal{NP} -hardness. Therefore, we propose MCF (Most Capacity First) – a simple yet highly effective and efficient heuristic, to solve the user allocation in large-scale scenarios, aiming to allocate as many users to as few edge servers as possible. Our experiments on a real-world dataset demonstrate the performance of our approach against the baseline and state-of-the-art approaches.

The future work has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

ACKNOWLEDGMENTS

This research is partly funded by Australian Research Council Discovery Project grants DP170101932, DP180100212, and Laureate Fellowship FL190100035.

REFERENCES

- [1] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2018, pp. 230–245.
- [2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
- [3] 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.
- [4] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proceedings of IEEE Conference on Computer Communications*. IEEE, 2018, pp. 207–215.
- [5] 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*, vol. 26, no. 6, pp. 2651–2664, 2018.
- [6] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Transactions on Cloud Computing*, 2018.
- [7] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.
- [8] T. C. Ferreto, M. A. Netto, R. N. Calheiros, and C. A. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, 2011.
- [9] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *Journal of Network and Computer Applications*, vol. 52, pp. 11–25, 2015.
- [10] "Resource management in clouds: Survey and research challenges, author=Jennings, Brendan and Stadler, Rolf," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, 2015.
- [11] 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.
- [12] 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.
- [13] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, H. Liu, Y. Qin, and P. Chen, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proceedings of IEEE International Conference on Web Services*. IEEE, 2019, pp. 91–98.
- [14] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2019, pp. 86–101.
- [15] 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 and Distributed Systems*, no. 2, pp. 431–445, 2017.
- [16] J. Smith, "The Mobile Gaming Report: Market size, the free-to-play model, and new opportunities to market and monetize," Business Insider Intelligence, Tech. Rep., 2016. [Online]. Available: www.businessinsider.com/the-mobile-gaming-report-market-size-the-free-to-play-model-and-new-opportunities-to-market-and-monetize

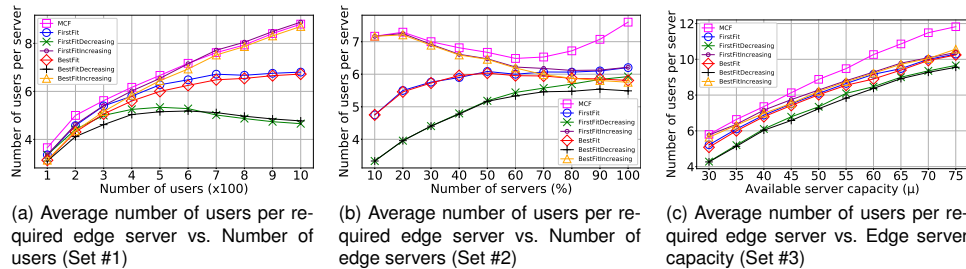


Fig. 7: Comparison of several classic approximation algorithms for the bin packing problem

[17] M. R. Garey and D. S. Johnson, *Computers and intractability*. WH Freeman New York, 2002, vol. 29.

[18] W. Leinberger, G. Karypis, and V. Kumar, "Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints," in *Proceedings of International Conference on Parallel Processing*. IEEE, 1999, pp. 404–412.

[19] H. Hallawi, J. Mehnert, and H. He, "Multi-capacity combinatorial ordering GA in application to cloud resources allocation and efficient virtual machines consolidation," *Future Generation Computer Systems*, vol. 69, pp. 1–10, 2017.

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

[21] M. Azzam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "MeFoRE: QoE based resource estimation at fog to enhance QoS in IoT," in *Proceedings of International Conference on Telecommunications*. IEEE, 2016, pp. 1–5.

[22] C. Romero, *Handbook of critical issues in goal programming*. Elsevier, 2014.

[23] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," Microsoft Research, Tech. Rep., 2011. [Online]. Available: www.microsoft.com/en-us/research/wp-content/uploads/2011/01/VBPackingESA11.pdf

[24] ETSI, "Universal Mobile Telecommunications System (UMTS); Requirements for Support of Radio Resource Management (FDD) (3G TS 25.133 version 3.1.0 Release 1999)," Technical Report, Tech. Rep., 2000. [Online]. Available: www.etsi.org/deliver/etsi_ts/125100_125199/125133/03.01.00_60/ts_125133v030100p.pdf

[25] L. Wang, L. Jiao, J. Li, J. Gedeon, and M. Mühlhäuser, "MOERA: Mobility-agnostic online resource allocation for edge computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1843–1856, 2018.

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

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

[28] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. CRC Press, 2003.

[29] M. Yannuzzi, F. van Lingen, A. Jain, O. L. Parellada, M. M. Flores, D. Carrera, J. L. Pérez, D. Montero, P. Chacin, A. Corsaro, and Others, "A new era for cities with fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 54–67, 2017.

[30] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, "An application of bin-packing to multiprocessor scheduling," *SIAM Journal on Computing*, vol. 7, no. 1, pp. 1–17, 1978.

[31] Y. Li, X. Tang, and W. Cai, "On dynamic bin packing for resource allocation in the cloud," in *Proceedings of ACM symposium on Parallelism in Algorithms and Architectures*. ACM, 2014, pp. 2–11.

[32] C. Li and X. Tang, "On fault-tolerant bin packing for online resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[33] D. Pisinger and M. Sigurd, "The two-dimensional bin packing problem with variable bin sizes and costs," *Discrete Optimization*, vol. 2, no. 2, pp. 154–167, 2005.

[34] C. Chekuri and S. Khanna, "On multidimensional packing problems," *SIAM Journal on Computing*, vol. 33, no. 4, pp. 837–851, 2004.

[35] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, "Dynamic bin packing," *SIAM Journal on Computing*, vol. 12, no. 2, pp. 227–258, 1983.

[36] P. Shaw, "A constraint for bin packing," in *Proceedings of International Conference on Principles and Practice of Constraint Programming*. Springer, 2004, pp. 648–662.

[37] M. Delorme, M. Iori, and S. Martello, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *European Journal of Operational Research*, vol. 255, no. 1, pp. 1–20, 2016.

[38] Y. Azar, I. R. Cohen, A. Fiat, and A. Roytman, "Packing small vectors," in *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2016, pp. 1511–1525.

[39] S. S. Seiden, R. Van Stee, and L. Epstein, "New bounds for variable-sized online bin packing," *SIAM Journal on Computing*, vol. 32, no. 2, pp. 455–469, 2003.

[40] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, "Approximation and online algorithms for multidimensional bin packing: A survey," *Computer Science Review*, vol. 24, pp. 63–79, 2017.

[41] E. G. Coffman Jr, J. Csirik, G. Galambos, S. Martello, and D. Vigo, "Bin packing approximation algorithms: Survey and classification," in *Handbook of Combinatorial Optimization*. Springer, 2013, pp. 455–531.

[42] M. Gabay and S. Zaourar, "Variable size vector bin packing heuristics - Application to the machine reassignment problem," Sep. 2013, working paper or preprint. [Online]. Available: www.hal.archives-ouvertes.fr/hal-00868016

The author biographies have been removed since the biographies have no close connection with the thesis. Please refer to the original paper online for a full, unedited version.

Supplementary Material to “Cost-Effective App User Allocation in an Edge Computing Environment”

Phu Lai, Qiang He, John Grundy, Feifei Chen, Mohamed Abdelrazek, John Hosking, and Yun Yang



APPENDIX A STATISTICAL ANALYSIS

In this section, we present the results of the one-sided Wilcoxon signed-rank tests [1] conducted on all three experiment sets (Tables 1, 2, and 3). The accepted hypotheses that MCF outperforms the other approaches with a significance level of 0.01 are highlighted in bold text. Tables 1, 2, and 3 clearly validate that the difference between the effectiveness of MCF and other baseline approaches are statistically significant under the majority of experimental settings.

APPENDIX B EXPERIMENTAL RESULTS OF SEVERAL CLASSIC ALGORITHMS FOR SOLVING THE BIN PACKING PROBLEM

In this section, we show the experimental results of MCF against other classic algorithms for solving the bin packing problem in all experiments sets in Figs. 1, 2, and 3. In Set #1, MCF, FirstFitIncreasing, and BestFitIncreasing clearly outperform other approaches. In Sets #2 and #3, MCF outperforms the rest, being able to serve more user using much fewer edge servers under almost all experimental settings.

We also present the results of the one-sided Wilcoxon signed-rank tests [1] conducted on all three experiment sets (Tables 4, 5, and 6). The accepted hypotheses that MCF outperforms the other classic algorithms for solving the bin packing problem with a significance level of 0.01 are highlighted in bold text. Tables 4, 5, and 6 clearly validate that the difference between the effectiveness of MCF and other approaches are statistically significant under the majority of experimental settings.

REFERENCES

- [1] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. CRC Press, 2003.

- P. Lai, Q. He, and Y. Yang are with the School of Software and Electrical Engineering, Swinburne University of Technology, 3122, Australia. E-mail: tlai, qhe, yyang@swin.edu.au.
- J. Grundy is with the School of Information Technology, Monash University, 3168, Australia. E-mail: john.grundy@monash.edu.
- F. Chen and M. Abdelrazek are with the School of Information Technology, Deakin University, 3125, Australia. E-mail: mohamed.abdelrazek, feifei.chen@deakin.edu.au.
- J. Hosking is with the School of Science, University of Auckland, Auckland, New Zealand. E-mail: j.hosking@auckland.ac.nz.

	<i>p</i> -value corresponding to each experimental parameter (number of users)									
	100	200	300	400	500	600	700	800	900	1,000
Metric: number of users per server (Fig. 3a in main manuscript)										
MCF vs. Optimal	1	1	1	1	1	1	1	1	1	1
MCF vs. Greedy	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. Random	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. ICSOC19	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. TPDS20	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	.006	<.001
Metric: required server percentage (Fig. 3b in main manuscript)										
MCF vs. Optimal	1	1	1	1	.007	<.001	.475	1	1	1
MCF vs. Greedy	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. Random	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. ICSOC19	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. TPDS20	<.001	<.001	<.001	<.001	.037	.011	<.001	.012	.006	<.001
Metric: allocated users percentage (Fig. 3c in main manuscript)										
MCF vs. Optimal	1	1	1	1	1	1	1	1	1	1
MCF vs. Greedy	1	1	<.001	1	.992	<.001	<.001	<.001	<.001	<.001
MCF vs. Random	.612	.940	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. ICSOC19	1	1	<.001	1	1	.986	<.001	<.001	<.001	<.001
MCF vs. TPDS20	.423	.007	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001

TABLE 1: Statistical results of MCF vs. baseline approaches in experiment Set #1 (varying number of users) with a significance level of 0.01

	<i>p</i> -value corresponding to each experimental parameter (number of edge servers)									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Metric: number of users per server (Fig. 4a in main manuscript)										
MCF vs. Optimal	1	1	1	1	1	1	1	1	1	1
MCF vs. Greedy	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. Random	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. ICSOC19	<.001	<.001	<.001	<.001	<.001	.843	<.001	<.001	<.001	<.001
MCF vs. TPDS20	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
Metric: required server percentage (Fig. 4b in main manuscript)										
MCF vs. Optimal	.500	.013	.010	.004	.059	1	.475	1	1	1
MCF vs. Greedy	.159	.004	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. Random	.159	.004	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. ICSOC19	.159	.004	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. TPDS20	.500	.029	.010	.052	<.001	.004	<.001	<.001	<.001	<.001
Metric: allocated users percentage (Fig. 4c in main manuscript)										
MCF vs. Optimal	1	1	1	1	1	1	1	1	1	1
MCF vs. Greedy	<.001	<.001	<.001	<.001	1	1	<.001	1	1	<.001
MCF vs. Random	<.001	<.001	<.001	<.001	<.001	<.001	<.001	.985	.995	<.001
MCF vs. ICSOC19	<.001	<.001	<.001	<.001	1	1	<.001	1	1	<.001
MCF vs. TPDS20	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001

TABLE 2: Statistical results of MCF vs. baseline approaches in experiment Set #2 (varying number of edge servers) with a significance level of 0.01

	<i>p</i> -value corresponding to each experimental parameter (average edge server capacity)									
	30	35	40	45	50	55	60	65	70	75
Metric: number of users per server (Fig. 5a in main manuscript)										
MCF vs. Optimal	1	1	1	1	1	1	1	1	1	1
MCF vs. Greedy	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. Random	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. ICSOC19	<.001	.697	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. TPDS20	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
Metric: required server percentage (Fig. 5b in main manuscript)										
MCF vs. Optimal	<.001	.018	1	1	1	1	1	1	1	1
MCF vs. Greedy	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. Random	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. ICSOC19	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
MCF vs. TPDS20	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001
Metric: allocated users percentage (Fig. 5c in main manuscript)										
MCF vs. Optimal	1	1	1	1	1	1	1	1	1	1
MCF vs. Greedy	<.001	1	1	1	1	1	1	1	1	1
MCF vs. Random	<.001	<.001	<.001	<.001	<.001	.111	.032	.020	.020	<.001
MCF vs. ICSOC19	1	1	1	1	1	1	1	1	1	1
MCF vs. TPDS20	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001	<.001

TABLE 3: Statistical results of MCF vs. baseline approaches in experiment Set #3 (varying number of edge servers' capacity) with a significance level of 0.01

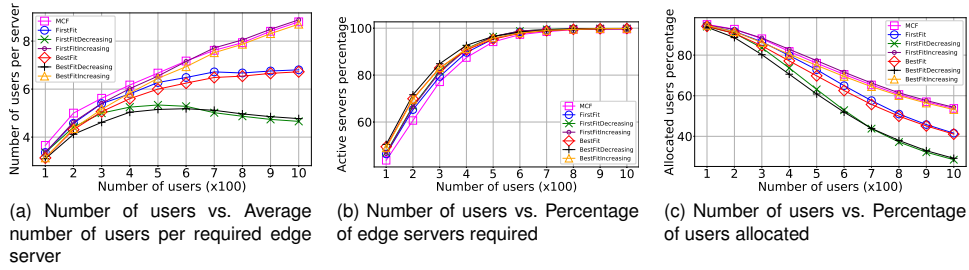


Fig. 1: Experimental results of experiment Set #1 (varying number of users) for several classic approximation algorithms for the bin packing problem

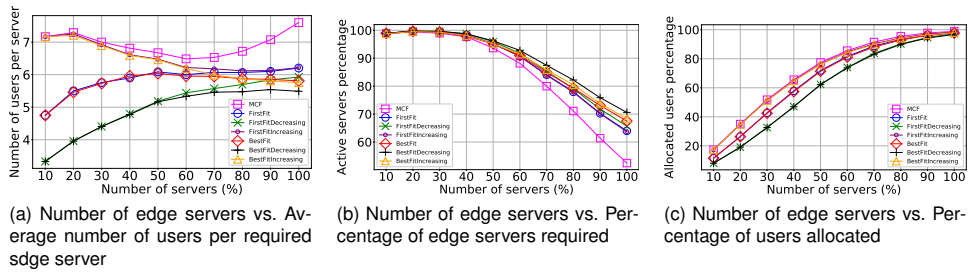


Fig. 2: Experimental results of experiment Set #2 (varying number of edge servers) for several classic approximation algorithms for the bin packing problem

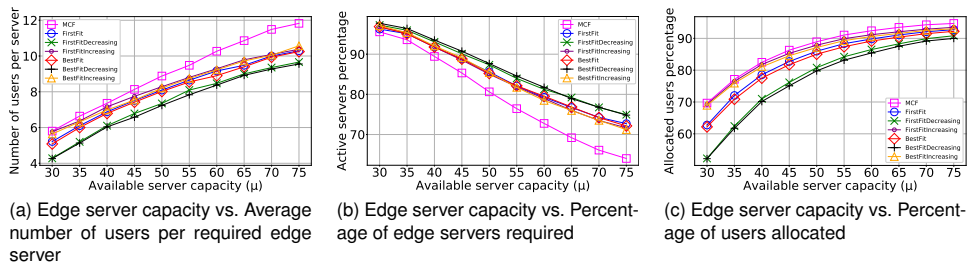


Fig. 3: Experimental results of experiment Set #3 (varying edge servers' capacity) for several classic approximation algorithms for the bin packing problem

	<i>p</i> -value corresponding to each experimental parameter (number of users)									
	100	200	300	400	500	600	700	800	900	1000
Metric: number of users per server (Fig. 1a)										
MCF vs. FirstFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing	< .001	< .001	< .001	< .001	< .001	.564	1	1	1	1
MCF vs. BestFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitIncreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
Metric: required server percentage (Fig. 1b)										
MCF vs. FirstFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFit	< .001	< .001	< .001	< .001	< .001	.004	.002	.120	.037	< .001
MCF vs. BestFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	.004	.021	< .001
MCF vs. BestFitIncreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	.138	.005	< .001
Metric: allocated users percentage (Fig. 1c)										
MCF vs. FirstFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing	.002	.116	.998	1	1	1	1	1	1	1
MCF vs. BestFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitIncreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001

TABLE 4: Statistical results of MCF vs. several classic approximation algorithms for the bin packing problem in experiment Set #1 (varying number of users) with a significance level of 0.01

	<i>p</i> -value corresponding to each experimental parameter (number of edge servers)									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Metric: number of users per server (Fig. 2a)										
MCF vs. FirstFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitIncreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
Metric: required server percentage (Fig. 2b)										
MCF vs. FirstFit	.159	.004	.001	.002	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing	.159	.004	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing	.921	.013	.002	.006	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFit	.159	.004	.002	.171	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitDecreasing	.159	.004	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitIncreasing	.718	.128	.010	.013	< .001	< .001	< .001	< .001	< .001	< .001
Metric: allocated users percentage (Fig. 2c)										
MCF vs. FirstFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFit	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitDecreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitIncreasing	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001

TABLE 5: Statistical results of MCF vs. several classic approximation algorithms for the bin packing problem in experiment Set #2 (varying number of edge servers) with a significance level of 0.01

		<i>p</i> -value corresponding to each experimental parameter (average edge server capacity)									
		30	35	40	45	50	55	60	65	70	75
		Metric: number of users per server (Fig. 3a)									
MCF vs. FirstFit		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFit		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitDecreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitIncreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
		Metric: required server percentage (Fig. 3b)									
MCF vs. FirstFit		.002	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFit		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitDecreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitIncreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
		Metric: allocated users percentage (Fig. 3c)									
MCF vs. FirstFit		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitDecreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. FirstFitIncreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFit		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitDecreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001
MCF vs. BestFitIncreasing		< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001	< .001

TABLE 6: Statistical results of MCF vs. several classic approximation algorithms for the bin packing problem in experiment Set #3 (varying edge servers' capacity) with a significance level of 0.01

Quality of Service-Aware User Allocation in Mobile Edge Computing Systems

In Research Problem 1, the QoS level of each user is fixed. In this chapter, we investigate the scenario where the QoS level of a user can be flexibly adjusted. For example, for a video-streaming service, the app vendor can determine the resolution of the videos being streamed to each user such as 360p, 480p, 720p, 1080p, or 1440p, etc. Each QoS level is associated with a level of user satisfaction (measured by QoE). The app vendor aims to allocate users to edge servers and select a QoS for each user so that the total user satisfaction is maximized.

3.1 An Integer Programming-based Approach and A Greedy Heuristic Approach

We define and model the QoS-aware EUA problem and prove its NP-hardness. To find optimal solutions, we formulate this problem as an integer linear programming problem and employ IBM ILOG CPLEX Optimizer to solve it. Being NP-hard, it is non-trivial to solve this problem in large-scale scenarios. Thus, we propose an efficient greedy heuristic. The proposed optimal and heuristic approaches are then experimentally evaluated against a baseline and a state-of-the-art approach. Predictably, the optimal approach is very inefficient and impractical in most real-world situations. The heuristic sig-

nificantly outperforms the other two approaches in most experimental settings. When the numbers of users are large, the total user QoE produced by the heuristic is lower than the other approaches. This has motivated us to come up with a more effective solution, which is discussed in the next section.

This section is presented in the form of our published paper [29] as **P. Lai**, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, Y. Yang, "Edge user allocation with dynamic quality of service," in *Proceedings of International Conference on Service-Oriented Computing*, Springer, 2019, pp. 86–101. *Note that several sections in the paper included below have been removed or slightly modified to reduce repeated content that has appeared else where in this thesis. For a full, unedited version, please refer to the original paper itself.*

Edge User Allocation with Dynamic Quality of Service

Phu Lai¹, Qiang He¹ (✉), Guangming Cui¹, Xiaoyu Xia², Mohamed Abdelrazek², Feifei Chen², John Hosking⁴, John Grundy³, and Yun Yang¹

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 Integer-Programming based 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

Several introductory paragraphs have been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

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.

2 P. Lai et al.

Many researchers have found that there is a quantitative correlation between QoS and QoE, as visualized in Fig. 1 [2, 5, 12]. 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.

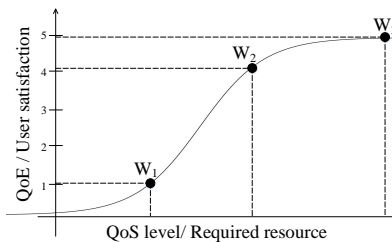


Fig. 1: Quality of Experience - Quality of Service correlation

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 [13]. 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 Integer Programming-based approach based on integer programming (IP) 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 IP-based approach to find optimal solutions 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

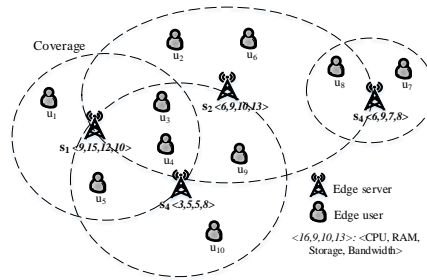


Fig. 2: Dynamic QoS EUA example scenario

Using the game streaming example in Sect. 1, let us consider a simple scenario shown in Fig. 2. There are ten players u_1, \dots, u_{10} , and four edge server s_1, \dots, 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 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.

4 P. Lai et al.

Let us assume server s_2 has already reached its 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 $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$, and n users $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$ 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
$\mathcal{S} = \{s_1, s_2, \dots, s_m\}$	finite set of edge server s_j , where $j = 1, 2, \dots, m$
$\mathcal{D} = \{CPU, RAM, storage, bandwidth\}$	a set of computing resource dimension
$c_j = \langle c_j^1, c_j^2, \dots, c_j^d \rangle$	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 , $k \in \mathcal{D}$
$\mathcal{U} = \{u_1, u_2, \dots, u_n\}$	finite set of user u_i , where $i = 1, 2, \dots, n$
$\mathcal{W} = \{W_1, W_2, \dots, W_q\}$	a set of predefined resource level W_l , where $l = 1, 2, \dots, q$. 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.
$w_i = \langle w_i^1, w_i^2, \dots, w_i^d \rangle$	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 $w_i \in \mathcal{W}$
$\mathcal{U}(s_j)$	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.

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

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 $(20, 28, 24, 24)$, exceeding server s_1 's available resources $(9, 15, 12, 10)$.

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

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, 5, 12], QoS is non-linearly correlated with QoE. When the QoS reaches a specific level, a user's QoE improves very trivially

6 P. Lai et al.

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 [7, 9, 16]. In this research, we use a logistic function (Equation 3), a generalized version of the sigmoid function, to model the QoS - QoE correlation. This gives us more control over the QoE model, including QoE growth rate, making the model more generalizable to different domains.

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

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 $\{u_1, \dots, u_n\} \rightarrow \{s_1, \dots, s_m\}$ with their individual QoS levels $\{w_1, \dots, w_n\}$ in order to maximize the overall QoE of all users:

$$\text{maximize } \sum_{i=1}^n E_i \quad (4)$$

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 $\mathcal{L} = \{w_1, w_2, \dots, w_n\}$ and a set of server resource capacity $\mathcal{C} = \{c_1, c_2, \dots, c_m\}$; 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 $w_i \in \mathcal{W}, \forall i \in \{1, \dots, n\}$, 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 [6], to a specialization of the dynamic QoS EUA decision problem.

Definition 1. (PARTITION) *Given a finite sequence of non-negative integers $\mathcal{X} = (x_1, x_2, \dots, x_n)$, determine whether there exists a subset $\mathcal{S} \subseteq \{1, \dots, n\}$ 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 $\mathcal{X} = (x_1, x_2, \dots, x_n)$ 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 programming (IP) 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 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 IP model of the dynamic QoS problem can be formulated as follows:

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

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

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

$$\sum_{j=1}^m \sum_{l=1}^q x_{ijl} \leq 1 \quad \forall i \in \{1, \dots, n\} \quad (8)$$

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

8 P. Lai et al.

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

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

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 $W_l, \forall l \in \{1, \dots, q\}$. 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 IP problem with an Integer Programming solver, e.g. IBM ILOG CPLEX⁵, or Gurobi⁶, 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( $\mathcal{S}, \mathcal{U}$ )
2:   for each  $u_i \in \mathcal{U}$  do
3:      $\mathcal{S}_{u_i} \leftarrow \{s_j \in \mathcal{S} \mid u_i \in cov(s_j)\}$ ;
4:     if  $\mathcal{S}_{u_i} \neq \emptyset$  then
5:        $s_{u_i} \leftarrow \operatorname{argmax}_{s_j \in \{0\} \cup \mathcal{S}_{u_i}} \{s_j : c_j \geq W_1\}$ ;
6:        $w_i \leftarrow \operatorname{argmax}_{W_l \in \{0\} \cup \mathcal{W}} \{W_l : W_l \leq c_j\}$ ;
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 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,

⁵ www.ibm.com/analytics/cplex-optimizer/

⁶ www.gurobi.com/

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, 2.4GHz) and 8GB RAM. The IP model in Sect. 4.1 was solved with IBM ILOG CPLEX Optimizer.

5.1 Baseline Approaches

Our IP-based approach (referred to as Optimal hereafter) and sub-optimal heuristic (referred to as Heuristic hereafter) 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*: [14] 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 [14], 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.
- 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.

10 P. Lai et al.

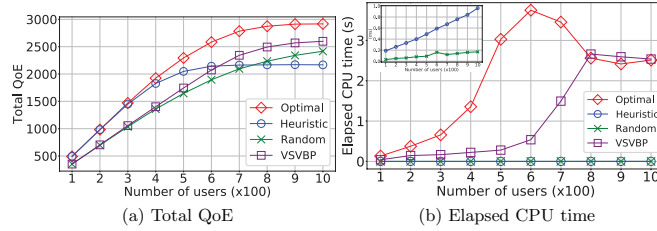


Fig. 3: Experiment set #1 results

- 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 $\mu = 5, 10, 15, \dots, 50$ 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 IP-based approach for finding optimal solutions, 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.

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

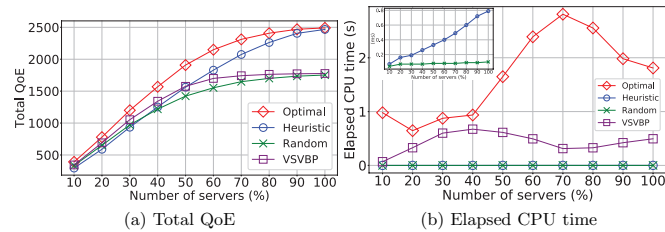


Fig. 4: Experiment set #2 results

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 (5, 7, 6, 6). 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 use 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.

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 Opti-

12 P. Lai et al.

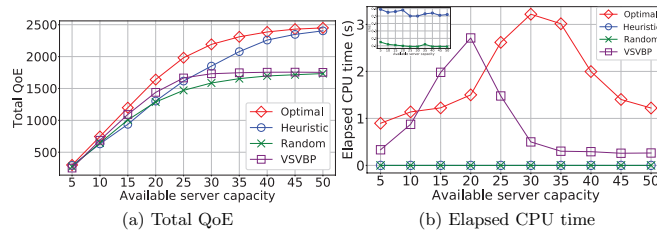


Fig. 5: Experiment set #3 results

mal 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 milliseconds. 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

This section has been removed since there is no close connection with the context of the thesis. Please refer to the original paper online for a full, unedited version.

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 [10]. Su et al. [17] 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. [8] 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. [4] 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. [15] and [1] 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. [11] 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 IP-based 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

14 P. Lai et al.

to evaluate the effectiveness and efficiency of the proposed approaches against a baseline approach and the state of the art.

The future work has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

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. Chen, M., Zhang, Y., Li, Y., Mao, S., Leung, V.C.: Emc: Emotion-aware mobile cloud computing in 5g. IEEE Network **29**(2), 32–38 (2015)
5. Fiedler, M., Hossfeld, T., Tran-Gia, P.: A generic quantitative relationship between quality of experience and quality of service. IEEE Network **24**(2), 36–41 (2010)
6. Garey, M.R., Johnson, D.S.: Computers and intractability, vol. 29. wh freeman New York (2002)
7. Hande, P., Zhang, S., Chiang, M.: Distributed rate allocation for inelastic flows. IEEE/ACM Transactions on Networking (TON) **15**(6), 1240–1253 (2007)
8. 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 Transactions on Circuits and Systems for Video Technology **24**(4), 669–680 (2013)
9. Hemmati, M., McCormick, B., Shirmohammadi, S.: Qoe-aware bandwidth allocation for video traffic using sigmoidal programming. IEEE MultiMedia **24**(4), 80–90 (2017)
10. Hobbeld, T., Schatz, R., Varela, M., Timmerer, C.: Challenges of qoe management for cloud applications. IEEE Communications Magazine **50**(4), 28–36 (2012)
11. 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)
12. Hobbeld, 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)
13. 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)
14. Lai, P., He, Q., Abdelrazek, M., Chen, F., Hosking, J., Grundy, J., Yang, Y.: Optimal edge user allocation in edge computing with variable sized vector bin packing. In: International Conference on Service-Oriented Computing. pp. 230–245. Springer (2018)

15. Mahmud, R., Srirama, S.N., Ramamohanarao, K., Buyya, R.: Quality of experience (qoe)-aware placement of applications in fog computing environments. *Journal of Parallel and Distributed Computing* (2018)
16. Shenker, S.: Fundamental design issues for the future internet. *IEEE Journal on selected areas in communications* **13**(7), 1176–1188 (1995)
17. Su, Z., Xu, Q., Fei, M., Dong, M.: Game theoretic resource allocation in media cloud with mobile social users. *IEEE Transactions on Multimedia* **18**(8), 1650–1660 (2016)

3.2 An Iterative Heuristic Approach

The experimental results presented in Section 3.1 show some limitations of the proposed heuristic when the number of users gets large. Specifically, from 600 users onward (Fig. 3a in the paper shown in Section 3.1), its achieved overall user satisfaction is even worse than the other two baseline approaches. Thus, in this section, we propose a more dynamic and effective heuristic that incrementally improve each user's QoE.

This section is presented in the form of our published paper [31] as **P. Lai**, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, Y. Yang, "QoE-aware user allocation in edge computing systems with dynamic QoS," *Future Generation Computer Systems*, vol. 112, pp. 684–694, 2020. *Note that several sections in the paper included below have been removed or slightly modified to reduce repeated content that has appeared else where in this thesis. For a full, unedited version, please refer to the original paper itself.*

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

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

^a*Swinburne University of Technology, Hawthorn, Australia*

^b*Deakin University, Burwood, Australia*

^c*The University of Auckland, Auckland, New Zealand*

^d*Monash University, Clayton, Australia*

Abstract

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

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

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

*Corresponding author

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

Experience, Resource allocation

1. Introduction

Several introductory paragraphs have been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

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

- We formally define the QoE-aware EUA problem and show that it is an \mathcal{NP} -hard problem.
- We propose an optimal approach based on Integer Programming (IP) for solving this problem exactly.
- In our previous work [1], we introduced a heuristic approach to tackle the complexity of the problem. However, its effectiveness under resource-scarce scenarios needs to be improved. As a result, in this paper, we introduce QoEUA – a new heuristic that performs better than our previous heuristic under resource-scarce circumstances.
- Comprehensive experiments based on a real-world dataset are conducted to demonstrate the effectiveness and efficiency of QoEUA against several baseline and state-of-the-art approaches.

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

2. Motivating Example

The motivating example been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

3. Problem Definition

The problem definition been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

Table 1: Key Notations

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

4. Our Approach

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

4.1. Integer Programming Model

This section been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

4.2. Heuristic Approach

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

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

The user sorting is then followed by an iterative process (lines 6-11). In each

Algorithm 1 QoEUA

```

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

```

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

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

The high efficiency of QoEUA allows app vendors to continuously run it, or execute it on demand, to respond to user mobility. The consideration of user mobility would not affect the initial problem formulation and the proposed heuristic in some situations. Specifically, when a user moves outside the coverage of its serving edge server, it will be disconnected from the edge server; the occupied computing resources on that edge server will be released; QoEUA will then consider it as a new user. That user, together with other new users who need to be allocated, will be allocated to edge servers based on the rules defined

in Algorithm 1, lines 5-11 (one can now consider the set \mathcal{U} as a set of new users who need to be allocated). This is feasible as long as migrating users across edge servers does not incur extra costs, or if the extra costs are trivial. The extra costs could be the migration cost or service reconfiguration cost [2]. In some use cases, those extra costs could be relatively insignificant. Taking video streaming for example [3], where videos encoded with different resolutions are cached on edge servers, which allows a user to access them with low latency, switching the user across edge servers only requires a very small amount of data to be transferred, e.g., which video the user is watching, the position in the video where the user left off, and the resolution of the video. For applications and services where the extra costs are noticeable, the new costs will need to be modeled. Thus, the initial problem formulation and the proposed solution will need to be modified.

5. Experimental Evaluation

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

5.1. Benchmark Approaches

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

- *Random*: Each user is allocated to a random edge server as long as that server has sufficient remaining resources to accommodate this user and has this user within its coverage area. The QoS level to be assigned to this user is randomly determined based on the server's remaining resources. For example, if the maximum QoS level can be achieved the server is W_2 , the user will be randomly assigned either W_1 or W_2 .
- *ICSOC19* [1]: This is the greedy-like heuristic proposed in our previous work.
- *TPDS20* [4]: This approach solves the EUA problem with the objectives of maximizing the number of allocated users and minimizing the overall system cost calculated based on the costs of required computing resources on edge servers. Since TPDS20 does not consider dynamic QoS, users' QoS levels are randomly pre-specified.
- *ICSOC18* [5]: This work proposes an optimal approach that maximizes the number of allocated users while minimizing the number of edge servers required to serve the allocated users. Similar to TPDS20, this work does not consider dynamic QoS either. Thus, users are assigned the same QoS levels as TPDS20.

Table 2: Experimental Settings

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

5.2. Experimental Settings

The experiments are conducted on the EUA dataset¹ [5], which contains the geographical locations of end-users and all cellular base stations in Australia. This dataset is also used in [4], [1], and [5] to evaluate TPDS20, ICSOC19, and ICSOC18, respectively.

Edge servers: To capture the characteristics of a MEC environment [6], we simulate a highly dense urban area of 1.8 km² covered by 125 base stations, each equipped with an edge server. The coverage radius of each edge server is randomly generated within 100-150m. The computing resources available on the edge servers, or their capacities, are randomly generated by following a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the average capacity of each resource type in \mathcal{D} , and the standard deviation $\sigma = 10$ for all conducted experiments.

Edge users: We assume that for each user, there are three possible QoS levels $\mathcal{W} = \{< 1, 2, 1, 2 >, < 2, 3, 3, 4 >, < 5, 7, 6, 6 >\}$, and $\mathcal{D} = \{\text{CPU, RAM, storage, bandwidth}\}$. Those four types of resources are the representative ones. The proposed approaches can accommodate other types of resources that are specific to app vendors' applications. We have conducted experiments with other settings and achieved similar results. Thus, we select those three QoS levels as representative in our experiments. Different values for the QoE model have also been tested. In the experiments, we set $L = 5$, $\alpha = 1.5$, and $\beta = 2$ as representative.

To comprehensively analyze the performance of our approaches in various EUA scenarios, we conduct a series of experiments with different varying parameters, including the number of users, number of edge servers, and edge server capacity. Table 2 summarizes the settings of the experiments, which will be discussed in the next section. Note that the values specified in the table are representative. Other experiments with different values other than those have been conducted, which yield similar results. Each experiment is repeated 100 times to obtain 100 different user distributions and the results are then averaged. This allows extreme cases, such as overly dense or sparse user/server distributions, to be neutralized. To evaluate the performance of the approaches in achieving the optimization objective, which is to maximize the total QoE of all users as discussed in Section 3, we compare the total QoE of all users achieved by the six approaches, the higher the better. In addition, we measure the number of users allocated to edge servers by each approach, the higher the better. The efficiency of all approaches is also evaluated.

¹www.github.com/swinedge/eua-dataset

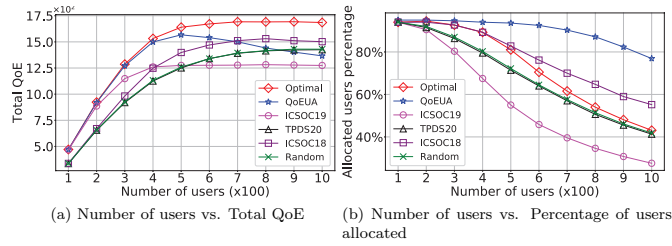


Figure 1: Experiment Set #1 results

5.3. Experimental Results and Discussion

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

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

In experiment Sets #2 and #3, we vary the number of edge servers available to serve users (Figure 2) and edge server capacity (Figure 3). Increasing those two parameters consequently increases the redundancy of computing resources

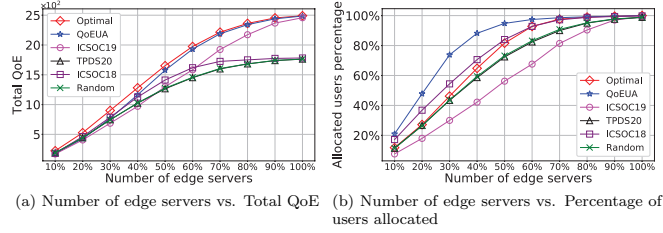


Figure 2: Experiment Set #2 results

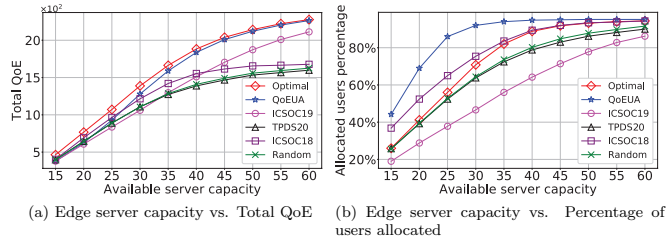


Figure 3: Experiment Set #3 results

available to serve users. As a result, we can observe the same trend in those two experiment sets, where the total QoE and the percentage of users allocated increase with the increase in the number of edge servers and the edge servers' capacities. This pattern can be observed under all experimental settings and for all approaches. In terms of the total QoE of all users (Figures 2a and 3a), QoEUA's performance is very close to Optimal, just slightly lower. In the meantime, QoEUA manages to allocate the most number of users, considerably greater than all other approaches under any experimental settings (Figures 2b and 3b). Clearly, QoEUA significantly outperforms the baseline and state-of-the-art approaches in experiment Sets #2 and #3. For more experimental results on the percentage of users allocated with different QoS levels, please refer to Appendix A.

2) *Efficiency*: Figure 4 illustrates the efficiency of all the approaches in the study, measured by the elapsed CPU time. The execution time of Optimal and ICSOC18 follow a similar pattern in all three experiment sets. As the experimental parameters increase from the starting point to a threshold somewhere in the middle – 400 users in Set #1, 80% of the number of edge servers in Set #2, and 55 average server resource capacity in Set #3 – the time rapidly rises

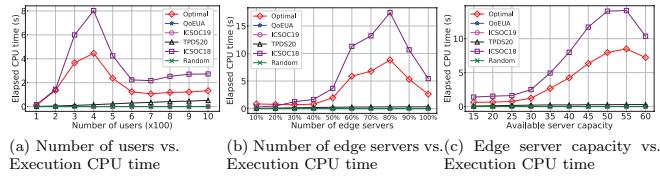


Figure 4: Execution CPU time in experiment Sets #1, #2, and #3

until it reaches a cap of around a hefty 8 seconds (for Optimal), which is unacceptable for real-time, latency-sensitive applications. This is expected since the QoE-aware EUA problem is \mathcal{NP} -hard. The rationale for this lies in the complexity of the problem. Adding up more users, edge servers, and available resources generates more possible options and solutions for Optimal to select from. After passing those thresholds, the execution time decreases then tends to converge. This happens in experiment Set #1 (Figure 4a) since when the number of users exceeds 400, a newly generated user tends to be positioned at the exact location of an existing user. Thus, the IBM CPLEX solver’s decision to be made for the new user can be based on the decision made for the existing user. As a result, we can see the elapsed CPU time of Optimal is almost symmetrical around that threshold (400 users) then roughly stabilizes at around 1 second, which is still very slow for real-time applications. In experiment Sets #1 and #2, the execution time decreases because after the experimental parameters exceed the above-mentioned thresholds, the available computing resources gradually become more redundant so that more users can obtain the highest QoS level without having to compete with each other. This generates fewer possible options for Optimal, thus takes less time to complete.

As the complexity of the problem increases by adding more users, edge servers, and edge server capacity, the execution time of other approaches also increases gradually. In all experiment sets, K50C19 takes 0.5-1 millisecond to solve the allocation problem. QoEUA is an iterative algorithm. The number of iterations it takes to complete is an important indicator of its efficiency. In the experiments, QoEUA requires 2-4 iterations, or 1-3 milliseconds of CPU time, which is acceptable for real-time applications and services.

5.4. Threats to Validity

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

distributions.

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

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

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

6. Related Work

This section has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

7. Conclusion and Future Work

App users' quality of experience (QoE) is of great importance for app vendors where user satisfaction is taken seriously. Despite being significant, there is very limited work in edge computing considering this aspect. Therefore, we have identified and formally formulated the QoE-aware edge user allocation problem with the goal of maximizing users' overall QoE as the first step of tackling QoE-oriented problems in edge computing. Having been proven to be \mathcal{NP} -hard and also experimentally illustrated, the optimal approach is not efficient once the problem scales up. In our previous work [1], we have proposed a heuristic to deal with the high complexity of the problem. However, that approach is not suitable in resource-scarce scenarios. This has led to the development of a more effective heuristic in this paper. We also conduct extensive experiments on a real-world dataset to evaluate the effectiveness and efficiency of our new approach against several baseline and state-of-the-art approaches.

The future work has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

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

Appendix A. Extra experimental results

In this appendix, we present the experimental results on the percentage of users allocated with different QoS levels in three experiment sets (Figures A.5, A.6, and A.7). As set in Section 5.2, there are three possible QoS levels $\mathcal{W} = \{W_1, W_2, W_3\}$, where W_3 is the most demanding QoS level, i.e., requiring the highest amount of computing resources among the three QoS levels, and W_1 is the least demanding one. We only present the results produced by Optimal, QoEUA, and ICSOC19 since they are the most relevant approaches which consider dynamic QoS level. The other approaches just randomly pre-assign QoS levels to users before the allocation process.

Figure A.5 depicts the percentage of unallocated users and users assigned with QoS levels W_1 , W_2 , and W_3 among all users in experiment Set #1. When the number of users to be allocated is low at 100, there are sufficient resources to accommodate almost all users at the highest QoS level. As the number of users increases, Optimal tends to allocate most users with W_2 since it is the most “economical” option, i.e., highest QoE per unit of resources. Although QoEUA does not tend to pick the most economical option, it is able to allocate more users than Optimal (also refer to Figure 1b). Greedy, being a greedy approach, allocates most users with the highest QoS level.

Figures A.6 and A.7 illustrate the percentage of unallocated users and users assigned with QoS levels W_1 , W_2 , and W_3 among all users in experiment Sets #2 and #3. Their results follow the same pattern since the two varying experimental parameters, i.e., number of edge servers and available server capacity, both directly affect the amount of available resources which can be used to serve users. When the amount of available resources is low, Optimal assigns W_2 , which is the most economical option, to most users. With QoEUA, most users get W_1 . As the amount of available resources increases, Optimal and QoEUA can then start assigning higher QoS levels to the users.

References

- [1] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, Y. Yang, Edge user allocation with dynamic quality of service, in: Proceedings of International Conference on Service-Oriented Computing, Springer, 2019, pp. 86–101.
- [2] L. Wang, L. Jiao, J. Li, J. Gedeon, M. Mühlhäuser, MOERA: Mobility-agnostic online resource allocation for edge computing, IEEE Transactions on Mobile Computing 18 (8) (2018) 1843–1856.
- [3] C. Liang, Y. He, F. R. Yu, N. Zhao, Enhancing video rate adaptation with mobile edge computing and caching in software-defined mobile networks, IEEE Transactions on Wireless Communications 17 (10) (2018) 7013–7026.
- [4] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, Y. Yang, A game-theoretical approach for user allocation in edge computing environment, IEEE Transactions on Parallel and Distributed Systems 31 (3) (2020) 515–529.

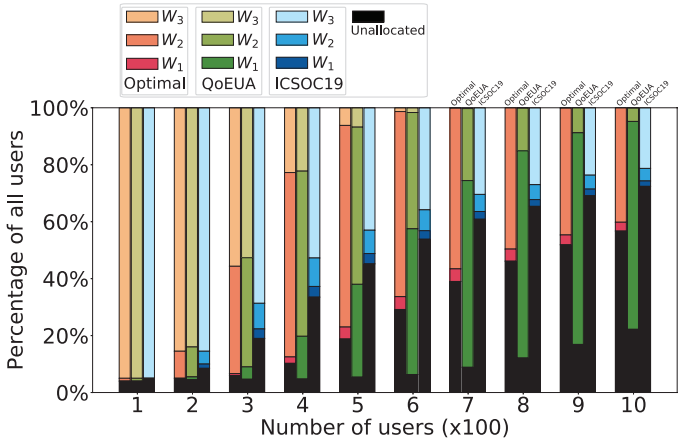


Figure A.5: Proportion of users by QoS levels in experiment Set #1

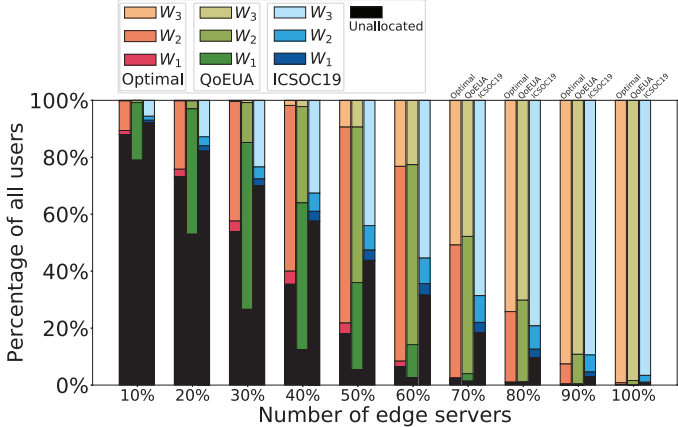


Figure A.6: Proportion of users by QoS levels in experiment Set #2

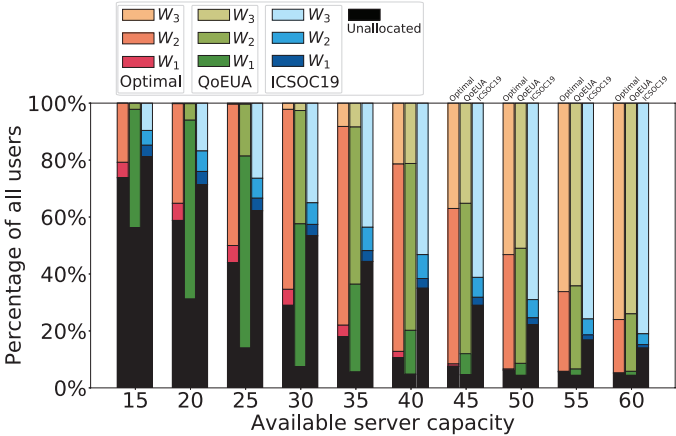


Figure A.7: Proportion of users by QoS levels in experiment Set #3

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

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

3.3 A Game-Theoretical Approach

The experimental results in Section 3.2 still show some limitations of the proposed heuristic when the number of users scales up. Even though it still outperforms the heuristic proposed in Section 3.1 in all experimental settings, it starts getting outperformed by the two baseline approaches when the number of users becomes really large (see Fig. 3a in the paper shown in Section 3.2). Thus, in this section, we improve it by formulating this problem as a potential game then solve it with a decentralized game-theoretical approach that is shown to be able to reach a Nash equilibrium. The results are now very close to optimal.

This section is presented in the form of our published paper [32] as **P. Lai**, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, Y. Yang, "Quality of experience-aware user allocation in edge computing systems: a potential game," in *Proceedings of IEEE International Conference on Distributed Computing Systems*, pp. 223-233, Singapore, 2020. ©2021 IEEE. Reprinted, with permission, from IEEE International Conference on Distributed Computing Systems. ***Note that several sections in the paper included below have been removed or slightly modified to reduce repeated content that has appeared else where in this thesis. For a full, unedited version, please refer to the original paper itself.***

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

Phu Lai*, Qiang He*[¶], Guangming Cui*, Feifei Chen[†], Mohamed Abdelrazek[‡],
John Grundy[‡], John Hosking[§], and Yun Yang*

*Swinburne University of Technology, Australia. Email: {tlai, qhe, gcui, yyang}@swin.edu.au

[†]Deakin University, Australia. Email: {feifei.chen, mohamed.abdelrazek}@deakin.edu.au

[‡]Monash University, Australia. Email: john.grundy@monash.edu

[§]The University of Auckland, New Zealand. Email: j.hosking@auckland.ac.nz

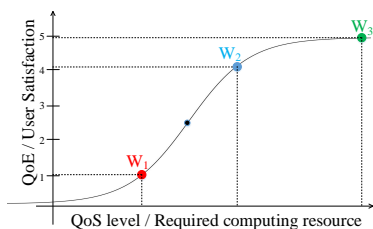


Fig. 1: QoS - QoE correlation

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

Index Terms—Edge computing, User allocation, Quality of Experience, Quality of Service, Game theory

I. INTRODUCTION

The main contributions of this paper include:

[¶]Qiang He is the corresponding author of this paper.

- We formulate this problem as a potential game [1] that aims to maximize the overall user QoE. The game is then theoretically analyzed and proven to admit a Nash equilibrium.
- Due to the \mathcal{NP} -hardness of the problem, finding an optimal solution to this problem is intractable, especially when a small geographic area could involve a great number of users. To effectively deal with its high complexity, we propose QoEGame, a distributed iterative algorithm for finding a Nash equilibrium. This algorithm simulates each user as a player in the game, whose decision, along with the decisions of other players, will benefit towards the final objective – maximizing the total QoE of all users.
- Extensive evaluations based on a real-world dataset are carried out to demonstrate the effectiveness and efficiency of QoEGame. The results show that QoEGame significantly outperforms the state of the art and other baseline approaches.

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

II. MOTIVATING EXAMPLE

This section has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

III. PROBLEM FORMULATION

A. System Model

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

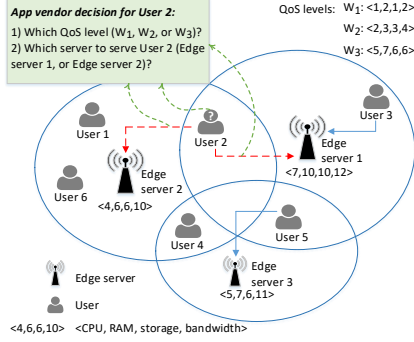


Fig. 2: Example MEC scenario

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

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

B. Allocation Decisions

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

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

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

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

User u_i can only be allocated to one of the neighbor edge servers \mathcal{S}_{u_i} , which are the edge servers that have user u_i in

their coverage areas (*proximity constraint*):

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

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

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

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

C. System Benefit (QoE) Model

In the QoS-aware EUA problem, an app vendor benefits from satisfying its users, or maximizing their users' QoE. In general, a higher QoS level results in a higher QoE level. As demonstrated in [2]–[4], QoS and QoE exhibit a nonlinear correlation. When the QoS reaches a particular level, a user's QoE shows a very trivial improvement regardless of a noticeable increase in the QoS. Take the model in Figure 1 for example, the QoE gained from the $W_2 - W_3$ upgrade is nearly 1. In the meantime, the QoE gained from the $W_1 - W_2$ upgrade is approximately 3 at the expense of a little extra resource. The logistic function (3) has been widely acknowledged and employed in a lot of works [5]–[7] to model the correlation between QoE and QoS due to its generality and simplicity, which increases the generalizability of this work.

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

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

D. Optimization Model

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

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

This formulation maximizes the total QoE of all users while satisfying the proximity constraint (1) and resource constraint (2). The solution to this problem is an allocation decision profile \mathbf{p} . [8] proves that this problem is \mathcal{NP} -hard by reducing the Partition problem to a special case of the decision version of this QoS-aware EUA problem.

IV. QOE-AWARE USER ALLOCATION GAME

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

A. Game Formulation

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

Then, we model the above QoE-aware problem as a game $\Gamma = (\mathcal{U}, \{\mathcal{P}_i\}_{u_i \in \mathcal{U}}, \{E_i\}_{u_i \in \mathcal{U}})$, where the set of players is the set of users \mathcal{U} , \mathcal{P}_i is the set of possible allocation strategies for users u_i , and E_i is the benefit function which measures the benefit (total QoE) produced by user u_i 's decision $p_i \in \mathcal{P}_i$. In the game, users' allocation strategies might conflict. For example, in Figure 2, allocating users 2 and 4 to edge server 2 might exhaust its available computing resources, preventing users 1 and 6 from using the app with high QoS levels or even from being served by edge server 2. A better solution would be to allocate users 2 and 4 to edge servers 1 and 3, respectively, if they have sufficient resources, and users 1 and 6 to edge server 2. In this way, the total QoE of all users is maximized, every user is happy and does not desire to deviate from their existing allocation strategies. Next, we investigate whether this game admits at least one Nash equilibrium – a stable state of the game in which no player can make a decision that improves its own benefit if other players' strategies remain unchanged [9]. In the game, it is a stable state where no user can make a decision that improves the overall benefit of all neighbor users instead of its own benefit because our objective

is to maximize the total benefit of all the users, as discussed above.

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

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

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

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

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

B. Game Property

A critical property of a potential game is that it admits at least one Nash equilibrium [1]. In this section, we confirm the existence of a Nash equilibrium in the QoE-aware EUA game by proving that this is a potential game.

Definition 5. (POTENTIAL GAME) A game is a potential game if the following holds for a potential function $\phi(\mathbf{p})$:

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

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

Based on Definition 5, we define a potential function:

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

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

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

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

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

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

$$\frac{\sum_{d \in \mathcal{D}} b_i^d}{|\mathcal{D}|} < \frac{\sum_{d \in \mathcal{D}} b_i'^d}{|\mathcal{D}|}, \text{ which implies } \sum_{d \in \mathcal{D}} b_i^d < \sum_{d \in \mathcal{D}} b_i'^d \quad (8)$$

Based on (8), we have:

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

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

V. DISTRIBUTED USER ALLOCATION ALGORITHM

In this section, we introduce QoEGame – an iterative and distributed user allocation algorithm for finding a Nash equilibrium in a potential game. QoEGame is inspired by *best response dynamics* [10], an evolutionary process that involves a finite number of iterations. In every iteration, each individual user develops the best allocation strategy in response to other users' strategies. It is important to note that the actual computation happens on edge servers, not on user devices. The process ends when no user desires to update their decisions, i.e. a Nash equilibrium. This is called the *Finite Improvement Property* of potential games.

A. Algorithm Design

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

Algorithm 1 QoEGame

```

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

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

updated decision p'_i must produce a higher total QoE compared to the previous decision p_i .

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

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

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

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

Proof: According to (7), we have:

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

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

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

According to the proof of Theorem 1, we have:

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

Since b_i^d is the normalized amount of type- d resource required by user u_i , we have $\sum_{d \in \mathcal{D}} b_i^{d'} - \sum_{d \in \mathcal{D}} b_i^d \geq 1$. Thus, $\phi_{\mathbf{p}}(p'_i) - \phi_{\mathbf{p}}(p_i) \geq 1 \sum_{u_j \neq u_i} \sum_{d \in \mathcal{D}} b_j^d = \sum_{u_j \neq u_i} Q_j \geq (n-1)Q_{\min} = \delta_{\text{user}}$

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

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

Therefore, Theorem 2 holds. \blacksquare

B. Price of Anarchy in Total QoE

The design of QoEGame involves non-deterministic factors – if there are multiple users proposing allocation decisions with the same system benefit improvement, one of them will be randomly selected as the winner in that iteration. This leads to the fact that there might be more than one Nash equilibrium in the game. Thus, we evaluate the performance of QoEGame by analyzing the Price of Anarchy (PoA) in the total QoE, which indicates the ratio between the worst Nash equilibrium and the optimal allocation strategy [1]. Let χ denote the set of decision profiles that are able to reach different Nash equilibria in the game and $\mathbf{p}^* = (p_1^*, p_2^*, \dots, p_n^*)$ denote the optimal decision profile. Given a decision profile $\mathbf{p} \in \chi$, let $poa_{QoE}(\mathbf{p})$ be the PoA measured by the ratio between the total QoE produced by \mathbf{p} and \mathbf{p}^* , $poa_{QoE}(\mathbf{p})$ is calculated as follows:

$$poa_{QoE}(\mathbf{p}) = \frac{\min_{\mathbf{p} \in \chi} \sum_{u_i \in \mathcal{U}} E_{\mathbf{p}}(p_i)}{\sum_{u_i \in \mathcal{U}} E_{\mathbf{p}^*}(p_i)} \quad (9)$$

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

Based on the above definitions, we have Theorem 3.

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

$$1 \geq poa_{QoE}(\mathbf{p}) \geq \frac{\sum_{u_i \in \mathcal{U}} QoE_{\min}(\mathbf{p})I_{\{p_i \neq (0,0)\}}}{\sum_{u_i \in \mathcal{U}} QoE_{\max}(\mathbf{p}^*)I_{\{p_i \neq (0,0)\}}} \quad (10)$$

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

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

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

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

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

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

Therefore, the minimum PoA of the total QoE is

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

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

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

VI. EMPIRICAL EVALUATION

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

A. Performance Benchmark

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

- *Optimal:* This is the integer programming (IP)-based approach introduced in [8], which finds optimal solutions to QoE-aware EUA problems, i.e. the solutions with the highest total QoE. This approach is implemented with the IBM ILOG CPLEX Optimizer solver¹.
- *ICSOC19:* Proposed in [8], this greedy approach allocates each user to an edge server that has the most available computing resources. Each user is then assigned the highest possible QoS level given the computing resources available on the edge server serving it.
- *TPDS20* [11]: This approach solves the EUA problem with the objectives of maximizing the number of allocated users and minimizing the overall system cost calculated based on the costs of required computing resources on edge servers. Since TPDS20 does not consider dynamic QoS, users' QoS levels are randomly pre-specified.
- *ICSOC18* [12]: This approach models the EUA problem as a variable-sized vector bin packing problem and proposes a lexicographic goal programming-based approach that maximizes the number of allocated users while

¹www.ibm.com/analytics/cplex-optimizer/

minimizing the number of edge servers required to serve the allocated users. Similar to TPDS20, this approach does not consider dynamic QoS either. Thus users are assigned the same QoS levels as TPDS20.

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

All the experiments are conducted on a Windows 10 Enterprise computer equipped with Intel Core i5-7400T processor (4 CPUs, 2.4GHz) and 8GB RAM.

B. Experimental Settings

The experiments are conducted on the EUA dataset² [12], which contains the geographical locations of end-users and all cellular base stations in Australia. This dataset was also used in [8], [11], and [12] to evaluate ICSOC19, TPDS20, and ICSOC18.

Edge servers: To capture the characteristics of a 5G environment [13], we simulate a 1.8 km² Melbourne CBD area covered by 125 base stations, each equipped with an edge server. The coverage radius of each edge server is randomly generated within 100-150m. The computing resources available on the edge servers are randomly generated following a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the average capacity of each resource type in \mathcal{D} , and the standard deviation $\sigma = 10$ for all conducted experiments. Since a normal distribution might contain negative numbers, any negative amount of computing resources generated is rounded up to 1.

Edge users: We assume that for each user, there are three possible QoS levels $\mathcal{W} = \{< 1, 2, 1, 2 >, < 2, 3, 3, 4 >, < 5, 7, 6, 6 >\}$, and $\mathcal{D} = \{CPU, RAM, storage, bandwidth\}$. We have conducted experiments with other settings and achieved similar results. Thus, we select those three QoS levels as representative in this section. Different values of the parameters in the QoE model (3) have also been tested. In this section, we employ $L = 5$, $\alpha = 1.5$, and $\beta = 2$, and present the corresponding results.

To comprehensively analyze the performance of QoEGame in various EUA scenarios, we conduct a series of experiments with different varying parameters, including the number of users, number of edge servers, and edge servers' available computing resources. Table 1 summarizes the settings of the experiments, which will be discussed in the next section. Each experiment is repeated 100 times to obtain 100 different user distributions and the results are then averaged. This allows extreme cases, such as overly dense or sparse user/server distributions, to be neutralized. To evaluate the performance of the approaches in achieving the optimization objective, which

TABLE I: Experimental Settings

	Users	Edge servers	Available resources (μ)
Set #1	100, ..., 1000	70%	35
Set #2	500	10%, ..., 100%	35
Set #3	500	70%	5, 10, ..., 50

is to maximize the total QoE of all users as discussed in Section III, we compare the total QoE of all users achieved by the six approaches, the higher the better. To evaluate the approaches from another perspective, we also measure the number of users that are allocated to edge servers by each approach, the higher the better. The efficiency of QoEGame is also evaluated.

C. Experimental Results

Figures 3, 4, and 5 demonstrate the effectiveness of all approaches in experiment Sets #1, #2, and #3 in terms of the total QoE of all users. Figures 6, 7, and 8 demonstrate their effectiveness in terms of the number of allocated users. In general, Optimal, being the IP-based approach for finding optimal solutions, clearly achieves the highest QoE compared to all other approaches across all experiments. This comes at the cost of its very high computational overhead (could go up to over 3 seconds as demonstrated in [8]) and is thus inapplicable in real-world 5G scenarios, where low latency is critical. QoEGame achieves a QoE performance very close to Optimal and clearly outperforms all other approaches. At the same time, QoEGame is able to allocate a good number of users to edge servers. We will also demonstrate the efficiency of QoEGame, measured by its convergence time, i.e. the number of iterations taken to reach a Nash equilibrium. This is a critical and machine-independent efficiency indicator for game-theoretic approaches [14]–[17].

1) Effectiveness:

Experiment Set #1. In this set of experiments, the number of users is varied from 100 users to 1,000 users in steps of 100. The number of edge servers is fixed at 70% of all edge servers in the simulated area. Figure 3 shows the total QoE of all users in the experiments. Under all experimental settings, the difference in the total QoE achieved by Optimal and QoEGame is very marginal, which, with the theoretical analysis in Section V-B, confirms the near-optimality of QoEGame. From 100 to 400 users, ICSOC19 achieves a QoE almost as high as Optimal and QoEGame. This occurs under those settings because the available computing resources are redundant and therefore almost all users receive the highest QoS level. However, as the number of users continues to increase while the total amount of computing resources is fixed, the average amount of computing resources for each user becomes more scarce. As a result, the performance of ICSOC19 deteriorates quickly. Random and the other two state-of-the-art approaches, i.e. ICSOC18 and TPDS20, are outperformed by Optimal and QoEGame since they do not consider the scenario where the QoS level of a user can be dynamically adjusted. TPDS20 is even worse than Random since it focuses on minimizing system costs.

²www.github.com/swinedge/eua-dataset/

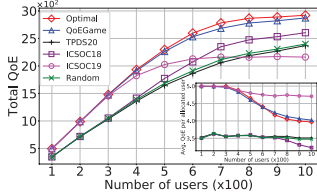


Fig. 3: Total QoE vs. number of users (Set #1).

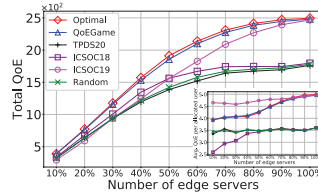


Fig. 4: Total QoE vs. number of edge servers (Set #2).

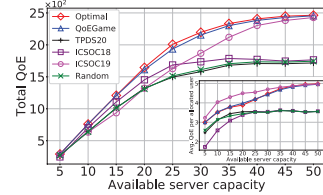


Fig. 5: Total QoE vs. edge server's available computing resources (Set #3).

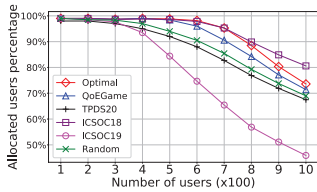


Fig. 6: Percentage of allocated users vs. number of users (Set #1).

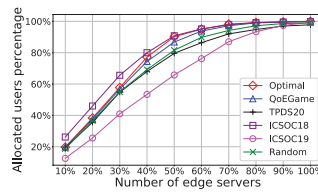


Fig. 7: Percentage of allocated users vs. number of edge servers (Set #2).

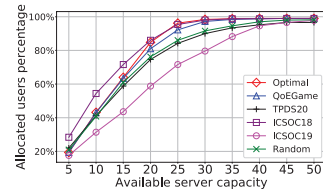


Fig. 8: Percentage of allocated users vs. edge server's available computing resources (Set #3).

Figure 6 shows the percentage of allocated users. We can observe a decreasing trend here. Clearly, since the amount of computing resources is fixed, introducing more users will increase the number of users who cannot be allocated to any edge servers. ICSOC18, aiming to maximize the number of users served, is obviously able to allocate the most users as also demonstrated in Figures 7 and 8, closely followed by Optimal and QoEGame. Random and TPDS20 allocate fewer users than ICSOC18, Optimal, and QoEGame. ICSOC19 is by far the worst because it rapidly exhausts the available computing resources on edge servers.

Note that given the same amount of computing resources under all experimental settings, increasing the number of users will consequently decrease the average QoE of each allocated user as shown in the inset graph in Figure 3 since the same amount of computing resources is now to be shared among more users. ICSOC19 appears to be the best approach in this aspect since every user is greedily assigned the highest possible QoS level on an edge server. Nevertheless, this results in an extremely low number of allocated users. This can be observed in all other sets of experiments.

Experiment Sets #2 & #3. In these experiment sets, we vary the number of edge servers available to serve users, from 10% to 100% in steps of 10% (Figures 4 and 7), and the amount of available computing resources on edge servers, from 5 to 50 in steps of 5 (Figures 5 and 8). The total QoE depicted in Figures 4 and 5 exhibits a trend similar to experiment Set #1 (Figure 3), where QoEGame achieves a very near-optimal performance and outperforms all other approaches by considerable margins. As we increase the number of edge servers and the available

computing resources, the total computing resources become more redundant, allowing more users to enjoy the highest QoS level. This is the reason why ICSOC19 also gradually approaches the performance of the Optimal. Additionally, as the computing resources become redundant, more users can be allocated to edge servers as illustrated in Figures 7 and 8. As can be seen in the figures, Optimal and QoEGame continue to outperform TPDS20 and Random. The percentage of users allocated by ICSOC19 is remarkably lower compared to all other approaches.

Optimal and QoEGame also outperform TPDS20, IC-SOC18, and Random, in terms of the average QoE per allocated user (inset graphs in Figures 4 and 5). ICSOC19 achieves the best performance in this aspect due to the same reason discussed for experiment Set #1.

2) Efficiency:

It is intractable to find an optimal solution to the $\mathcal{N}\mathcal{P}$ -hard QoE-aware EUA problem with Optimal in large-scale scenarios. As a result, we introduce QoEGame, an iterative and distributed algorithm. In this section, we discuss the efficiency of QoEGame and how well it scales with different experiment parameters. Figures 9, 10, and 11 demonstrate the convergence time of QoEGame in experiment Sets #1, #2, and #3, respectively, measured by the average number of iterations required by QoEGame to reach a Nash equilibrium. In Figure 9, the number of iterations increases linearly with the increase in the number of users from 100 to 500 users. From 500 users onwards, the convergence time decreases at a slower pace than it increases. The rationale for these phenomena lies behind the competitiveness of the game. In

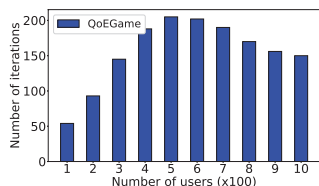


Fig. 9: Number of decision iterations vs. number of users (Set #1).

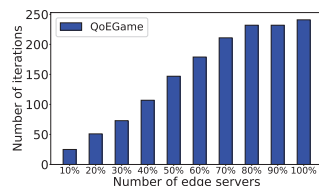


Fig. 10: Number of decision iterations vs. number of edge servers (Set #2).

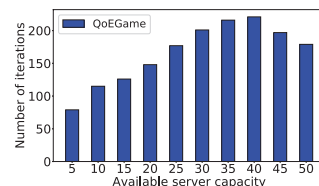


Fig. 11: Number of decision iterations vs. edge server's available capacity (Set #3).

the 100-500 user range, there are still sufficient computing resources to accommodate almost all the users (Figure 6). More users thus lead to more possible decisions to be made for each individual user, hence the increase in the convergence time. As the number of users enters the range of 600-1,000, most of the extra users are unallocated due to the scarcity of computing resources. However, because of the high density of users, who collectively contribute to finding the solution in parallel, QoEGame is able to find a Nash equilibrium quicker.

In experiment Set #2 (Figure 10), increasing the number of edge servers increases the possibility of decision updates, i.e. more decision updates, or more required iterations, hence the gradual increase in the convergence time of QoEGame. In experiment Set #3, as shown in Figure 11, increasing the available computing resources also increases the decision update possibility. However, after a certain point, 40 in this case, the convergence time decreases since there are now relatively redundant computing resources and more users can be served with high QoS levels without much competition.

VII. RELATED WORK

This section has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

VIII. CONCLUSIONS AND FUTURE WORK

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

of experiments conducted on a real-world dataset, against a number of baseline and state-of-the-art approaches.

The future work has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

ACKNOWLEDGMENT

This research is partly supported by Australian Research Council (ARC) Discovery Projects DP170101932 and DP180100212. John Grundy is supported by ARC Laureate Fellowship FL190100035.

REFERENCES

- [1] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996.
- [2] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between Quality of Experience and Quality of Service," *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010.
- [3] M. Alreshoodi and J. Woods, "Survey on QoE/QoS correlation models for multimedia services," *International Journal of Distributed and Parallel Systems*, vol. 4, no. 3, pp. 53–72, 2013.
- [4] T. Hofbeld, M. Seufert, M. Hirth, T. Zimmer, P. Tran-Gia, and R. Schatz, "Quantification of YouTube QoE via crowdsourcing," in *Proceedings of International Symposium on Multimedia*. IEEE, 2011, pp. 494–499.
- [5] M. Hemmati, B. McCormick, and S. Shirmohammadi, "QoE-aware bandwidth allocation for video traffic using sigmoidal programming," *IEEE MultiMedia*, vol. 24, no. 4, pp. 80–90, 2017.
- [6] S. Shenker, "Fundamental design issues for the future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.
- [7] P. Hande, S. Zhang, and M. Chiang, "Distributed rate allocation for inelastic flows," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1240–1253, 2007.
- [8] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2019, pp. 86–101.
- [9] R. B. Myerson, *Game theory*. Harvard University Press, 2013.
- [10] M. J. Osborne and A. Rubinstein, *A course in game theory*. MIT Press, 1994.
- [11] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.
- [12] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2018, pp. 230–245.
- [13] W. H. Chin, Z. Fan, and R. Haines, "Emerging technologies and research challenges for 5G wireless networks," *IEEE Wireless Communications*, vol. 21, no. 2, pp. 106–112, 2014.

- [14] 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.
- [15] 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.
- [16] 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.
- [17] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for joint cooperation and competition of edge resource allocation," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2019, pp. 503–513.

Dynamic User Allocation in Stochastic Mobile Edge Computing Systems

In Research Problems 1 and 2, we address a static situation where an app vendor needs to allocate a known, fixed set of users. This is a limitation to a dynamic scenario where users come and go randomly over time. Taking an online gaming service for instance, a user would come and start playing, stay for the duration of the game, then leave. The game provider does not know how many and when users would come or go. A user can be allocated to either an edge server or the remote cloud. If the user is allocated to an edge server whose resources are exhausted, the user would have to wait in a queue until existing users leave and free up the resources – this incurs a queuing delay cost. If allocated to the cloud, the user would not have to wait in a queue as the cloud has an ample amount of resources. However, this incurs a latency cost since the cloud is much further away from users compared to edge servers. The app vendor needs to decide where to allocate its users so that the system benefit is maximized. The system benefit is measured by the number of users allocated to edge servers minus the queuing delay and latency costs.

In this chapter, we propose an online algorithm based on the Lyapunov optimization framework, which is very effective in optimizing a system benefit metric in the long term while stabilizing the system, making sure all the queue backlogs are not overloaded. An app vendor is able to flexibly adjust its priority for the throughput benefit (measured by the number of users allocated to edge servers), queuing

delay cost, and cloud latency cost. The proposed algorithm does not rely on any prior knowledge of the statistics of user arrivals or departures and is able to automatically adapt to the priority set by the app vendor and allocate users accordingly to achieve the time-average optimization objective. We theoretically analyse its optimality and experimentally evaluate it against two baseline and three state-of-the-art approaches. Our approach achieves the highest system benefit among all the approaches in comparison. We also experimentally demonstrate that the trade-off between the throughput benefit and the user latency and queuing delay costs can be flexibly adjusted.

This chapter is presented in the form of our published paper [34] as **P. Lai**, Q. He, X. Xia, F. Chen, M. Abdelrazek, J. Grundy, J. Hosking, and Y. Yang, "Dynamic User Allocation in Stochastic Edge Computing Systems," *IEEE Transactions on Services Computing*, 2021. doi: 10.1109/TSC.2021.3063148. ©2021 IEEE. Reprinted, with permission, from IEEE Transactions on Services Computing. *Note that several sections in the paper included below have been removed or slightly modified to reduce repeated content that has appeared elsewhere in this thesis. For a full, unedited version, please refer to the original paper itself.*

Dynamic User Allocation in Stochastic Mobile Edge Computing Systems

Phu Lai, Qiang He, Xiaoyu Xia, Feifei Chen, Mohamed Abdelrazek, John Grundy, John Hosking, and Yun Yang

Abstract—Mobile edge computing (MEC) is a new distributed computing paradigm where edge servers are deployed at, or near cellular base stations in close proximity to end-users. This offers computing resources at the edge of the network, facilitating a highly accessible platform for real-time, latency-sensitive services. A typical MEC environment is highly stochastic with random user arrivals and departures over time. Here, we address the user allocation problem from a service provider’s perspective, who needs to allocate its users to the cloud or edge servers in a specific area. A user, who has a multi-dimensional resource requirement, can be allocated to either the remote cloud, which incurs a high latency, or an edge server, which results in a low latency but might require the user to wait in a queue. This study aims to achieve a controllable trade-off between performance (throughput) and several associated costs such as queuing delay and latency costs. We model this problem as a stochastic optimization problem, propose SUAC (Stochastic User Allocation) – an online Lyapunov optimization-based algorithm, and prove its performance bounds. The experimental results demonstrate that SUAC outperforms existing approaches, effectively allocating users with a desired trade-off while keeping the system strongly stable.

Index Terms—Mobile edge computing, user allocation, Lyapunov optimization, resource allocation

1 INTRODUCTION

IN recent years, mobile edge computing (MEC) has been introduced to tackle a major challenge in cloud computing – unpredictable and high latency, which is holding back the development of latency-sensitive applications and services such as VR/AR, smart cities, critical system warning, healthcare and so on. In a MEC environment, numerous edge servers are distributed at, or near cellular base stations or access points [1], which are much closer to end-users compared to remote cloud servers. Thus, this new distributed computing paradigm remarkably reduces end-to-end latency. A service provider such as Uber or YouTube can deploy its services on edge servers to better serve its users [2]. To minimize non-service areas, i.e., the areas that are not covered by any edge server, the coverage areas of adjacent edge servers usually partially overlap [3]. Fig. 1 depicts an example of a MEC system in a small area.

In a MEC environment, the edge user allocation (EUA) problem has arisen as a critical problem that challenges service providers [4], [5], [6], [7]. As thin clients, e.g., mobile or IoT devices, are not capable of executing computationally demanding tasks, they will be allocated to cloud servers or edge servers, which are more powerful and able to process those tasks. A service provider needs to decide where to allocate its users so that some optimization objectives are

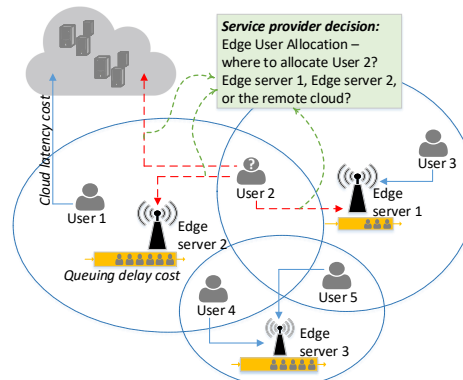


Fig. 1: An example MEC system

achieved. In an EUA scenario, each user consumes a specific amount of computing resources, e.g., CPU, RAM, storage, and bandwidth, on an edge server or a cloud server once allocated. An edge server will not be able to serve a user if it does not meet that user’s resource requirement. In this article, we focus on non-preemptive allocation [8], [9] – allocation without interrupting ongoing services of allocated users, i.e., it is not possible to reallocate allocated users.

Ideally, all end-users in a specific geographic area should be allocated to edge servers in their proximity. However, a service provider can only hire a relatively small amount of computing resources since an edge server usually has a very limited computing capacity [10], [11]. It is thus not always possible to serve all users simultaneously. As a consequence,

- P. Lai, Q. He, and Y. Yang are with the School of Software and Electrical Engineering, Swinburne University of Technology, 3122, Australia. E-mail: tlai, qhe, yyang@swin.edu.au.
- X. Xia, F. Chen and M. Abdelrazek are with the School of Information Technology, Deakin University, 3125, Australia. E-mail: xiaoyu.xia, feifei.chen, mohamed.abdelrazek@deakin.edu.au.
- J. Grundy is with the School of Information Technology, Monash University, 3168, Australia. E-mail: john.grundy@monash.edu.
- J. Hosking is with the School of Science, University of Auckland, Auckland, New Zealand. E-mail: j.hosking@auckland.ac.nz.

some users may have to wait to be served in a queue. Since an excessively long queue results in a long waiting time, some users will be allocated to the remote cloud to be served by a server in the cloud. As the cloud platform has the elasticity to scale up or down in real-time, it has virtually unlimited computing resources, hence no queuing needed for the users allocated to the remote cloud. This reduces the waiting time to almost zero but at the expense of high latency between the cloud and the users.

Existing research on the EUA problem has a number of limitations, which we address in this study. 1) *They assume that all the users have identical computing resource requirements or the arrivals of users are always known* [4], [5], [6], [7]. This is unrealistic because the MEC environment is highly stochastic. Users with various resource requirements join the MEC system dynamically and randomly over time. A user's exact duration in the MEC system is also unknown, i.e., they randomly depart from the MEC system. One can only give a rough estimate of how long a user would stay in the system. After a user's departure, the computing resources are released to serve new users. 2) *Existing approaches do not consider the fact that users can come and go over time* [4], [6]. Thus proposed approaches can only be executed to allocate users that join the MEC system in a single time slot. In the next time slot, when new users arrive, those approaches would have to be re-executed without taking any long-term objectives into account, thus the allocation solutions computed in different time slots might be conflicting each other. 3) More importantly, the limited computing resources in mobile edge computing [10], [11] and the massive number of users in 5G networks [12] further complicate the EUA problem. When the number of users is very large and the MEC system does not have sufficient computing resources, *our approach allows users to wait in a distributed queuing system*. In fact, the queuing method has been widely employed recently in edge computing [3], [13], [14]. The aforementioned existing approaches for EUA assume that users will be allocated to the cloud *immediately* when edge servers are exhausted of computing resources. However, *their models do not incorporate any costs that might incur when users are allocated to the cloud*. In addition, the allocated users do not stay in the MEC system permanently. The computing resources released when they leave the MEC system can be utilized to serve new users. This also has not been taken into account by existing approaches.

The *throughput* of a time-slotted MEC system is defined as the average number of users allocated to edge servers over time. At the beginning of each time slot, a service provider needs to decide where to allocate its users, the remote cloud or which edge server. Under the unpredictability of user arrivals and departures, the objectives of this stochastic EUA problem are twofold: 1) maximizing the throughput benefit by allocating as many users as possible to edge servers as long as it is beneficial to do so, and 2) minimizing the penalty incurred by the queuing delay and cloud-to-user latency. In order to achieve a controllable trade-off between the performance (throughput) and costs, we employ the Lyapunov optimization framework [15], which can make decisions based on the current state of the system without any future information about user arrivals and departures. Its unique advantage over other

online optimization approaches is its ability to optimize the performance of a system in a metric (the system benefit measured collectively by the throughput benefit, queuing delay cost, and latency cost in our study), while stabilizing the system (the length of the queues of users waiting to be allocated to each edge server in our study). It can be used to transform a long-term optimization problem into a series of short-term optimization problems, which are to be solved in each time slot. Doing this over multiple time slots will collectively achieve the long-term objective, i.e., system benefit maximization, while stabilizing all the user queues. The main contributions of this study include:

- We formally model the stochastic EUA problem that aims to help service providers allocate their users in order to achieve their long-term objectives, i.e., to maximize system throughput, and to minimize users' latency and queuing costs while keeping the stochastic MEC system strongly stabilized.
- We present SUAC, an online algorithm based on the Lyapunov optimization framework. Although SUAC does not require future information of user arrivals and departures, it can be theoretically shown that SUAC is able to find near-optimal solutions with an $[O(1/V), O(V)]$ performance-cost trade-off.
- A series of experiments are conducted on a real-world dataset to comprehensively evaluate the performance of SUAC.

The organization of this article is as follows. Section 2 introduces the MEC system model. Section 3 first introduces the throughput benefit and associated costs, then formulates the stochastic EUA problem. Section 4 presents the SUAC algorithm, which is then evaluated in Section 5. Section 6 reviews the related literature. Finally, Section 7 concludes the article and points out future work.

2 SYSTEM MODEL

In this section, we formally model the MEC system and its associated key characteristics. A MEC system in a specific geographical area consists of a set of edge servers. It is a time-slotted system where multiple users arrive in each time slot and need to be allocated to either an edge server or the cloud. Each edge server maintains a queue to hold users that have been allocated to the server but have not been served yet due to insufficient computing resources. The main notations used in this article are summarized in Table 1.

2.1 System Description

Edge Servers: The set of S edge servers denoted by $\mathcal{S} = \{1, 2, \dots, S\}$. Each edge server $s \in \mathcal{S}$ has a certain amount of different computing resource types \mathcal{R} such as CPU, RAM, storage, or bandwidth. Each edge server is equipped with a limited amount of computing resources. The computing capacity of an edge server s is an $|\mathcal{R}|$ -dimensional vector C_s . Each edge server covers a particular geographic area, as illustrated in Fig. 1.

Edge Users: There are K types of users categorized by their computing resource requirements, $k \in \mathcal{K} = \{1, 2, \dots, K\}$. Let an $|\mathcal{R}|$ -dimensional vector c_u denote user u 's computing resource requirement.

TABLE 1: Main Notations

Notation	Description
\mathcal{S}	the set of S edge servers s . $\mathcal{S} = \{1, 2, \dots, S\}$
\mathcal{R}	the set of computing resource types, or computing capacity dimensions. $\mathcal{R} = \{CPU, RAM, storage, bandwidth, \dots\}$
C_s	computing capacity of edge server s . C_s is an $ \mathcal{R} $ -dimensional vector where each dimension is the capacity of a resource type in \mathcal{R}
\mathcal{K}	the set of K user types k , categorized by their computing resource requirements. $\mathcal{K} = \{1, 2, \dots, K\}$
$\mathcal{A}_k(t)$	the set of type- k users u ($k \in \mathcal{K}$) arrive in time slot t . The set of all users arrive in time slot t is $\mathcal{A}(t) = \bigcup_{k \in \mathcal{K}} \mathcal{A}_k(t)$
c_u	computing resource requirement of user u . c_u is an $ \mathcal{R} $ -dimensional vector where each dimension is the capacity of a resource type in \mathcal{R}
$\mathbf{a}_{u,s}(t)$	allocation decision on whether user u will be allocated to edge server s in time slot t
$\mathbf{b}_u(t)$	allocation decision on whether user u will be allocated to the remote cloud in time slot t
$Q_s(t)$	queue backlog of edge server s , i.e., number of users waiting to be served by edge server s , in time slot t
$D_s(t)$	number of users who leave the queue and start being served by edge server s in time slot t
\bar{b}_r	time-average benefit gained from system throughput
\bar{c}_d	time-average queuing delay for all users
\bar{c}_h	time-average latency of all users allocated to the remote cloud
r_s	time-average throughput of edge server s
w_s	the weight that indicates service provider's priority for the throughput benefit gained from serving users by edge server s
N_s	service rate of edge server s , i.e., the maximum number of users can be simultaneously served by edge server s
ℓ	expected user session length
$n_s(t)$	number of users being served by edge server s in time slot t
h_u	user u 's cloud-to-user latency
$\omega_r, \omega_d, \omega_h$	normalizing parameters for throughput benefit, queuing delay cost, and latency cost
V	Lyapunov control parameter

User Arrivals and Departures: The operational timeline of this MEC system is discretely slotted with normalized slot duration $t \in \{0, 1, 2, \dots\}$. Each time slot t may range from several milliseconds, seconds, to a few minutes, depending on the application context. Let $\mathcal{A}_k(t)$ denote the set of type- k users that arrive in time slot t , the set of all users that arrive in time slot t is $\mathcal{A}(t) = \bigcup_{k \in \mathcal{K}} \mathcal{A}_k(t)$. Each random variable $|\mathcal{A}_k(t)|$, $\forall k \in \mathcal{K}$, is independent of the current number of users in the system. Without loss of generality, we assume that the number of users that arrive in a time slot is bounded, i.e., $\sum_{k \in \mathcal{K}} |\mathcal{A}_k(t)| \leq A^{max}$, $\forall t$, where A^{max} is the maximum of users of all types arrive in a time slot. The time-average arrival rate is given by $\lambda_k = \mathbb{E}\{|\mathcal{A}_k(t)|\}$. The total time-average user arrival rate λ of the system is thus $\sum_{k \in \mathcal{K}} \lambda_k$. A user's duration in the system (the length of a user session) is unknown at all time, i.e., users depart from the MEC system randomly, and is measured by the number of time slots. Our approach does not rely on any prior knowledge of the statistics of user arrivals or departures.

2.2 Allocation Decisions

In each time slot, a number of new users arrive and need to be allocated to either 1) ideally, edge servers, or 2) the

remote cloud. Let $\mathbf{a}_{u,s}(t), \mathbf{b}_u(t) \in \{0, 1\}$ be the allocation decision to be made for user u in time slot t . We have $\mathbf{a}_{u,s}(t) = 1$ if user u is to be allocated to edge server s , and $\mathbf{b}_u(t) = 1$ if user u is to be allocated to the remote cloud. Let $\mathbf{a}(t) = \{\mathbf{a}_{u,s}(t), \mathbf{b}_u(t)\}_{u \in \mathcal{A}(t)}$ denote the allocation strategy for all users that arrive in time slot t , which must satisfy the following constraints.

Firstly, each user u can be allocated to only one server, either the remote cloud server or an edge server:

$$\mathbf{b}_u(t) + \sum_{s \in \mathcal{S}} \mathbf{a}_{u,s}(t) = 1, \forall u \in \mathcal{A}(t), \forall t \quad (1)$$

Note that we do not consider users who are not located within the coverage of any edge server. A user u can be allocated to an edge server s only if it is located in that edge server's coverage area cov_s (proximity constraint):

$$\mathbf{a}_{u,s}(t) = 0 \text{ if } u \notin cov_s, \forall u \in \mathcal{A}(t), \forall s \in \mathcal{S}, \forall t \quad (2)$$

and the accumulated computing resource requirements of the users being served by an edge server must not exceed the capacity of that edge server (capacity constraint):

$$\sum_{u \in \mathcal{D}(t)} (\mathbf{a}_{u,s}(t) c_u) \preceq C_s, \forall s \in \mathcal{S}, \forall t \quad (3)$$

where $\mathcal{D}(t)$ is the set of users being served by edge server s .

2.3 Queuing Dynamics and Stability

We introduce a distributed queuing architecture where each edge server $s \in \mathcal{S}$ maintains a local queue. If a user is decided to be allocated to an edge server by a mechanism to be discussed later, it will be buffered in the queue of that edge server until that edge server has sufficient computing resources to serve it. We denote queue backlog $Q_s(t)$ as the queue length (number of waiting users) of edge server s in time slot t . We have the following queuing dynamics:

$$Q_s(t+1) = [Q_s(t) - D_s(t)]_+ + \mathbf{a}_s(t) \quad (4)$$

where $\mathbf{a}_s(t) = \sum_{u \in \mathcal{A}(t)} \mathbf{a}_{u,s}(t)$ is the number of users decided to be allocated to edge server s in time slot t , and $D_s(t)$ is the number of users that leave the queue and start being served by edge server s . Note that $D_s(t)$ is not the number of users who depart from the system (the user session has ended, or the service is no longer required) in time slot t . A queue $Q_s(t)$ is considered *strongly stable* [15] if the queue backlog is bounded: $\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{Q_s(t)\} < \infty, \forall s \in \mathcal{S}$. Intuitively, this means that the queue length remains finite and does not blow up to infinite over time. A MEC system is strongly stable when all the queues' lengths are bounded. We will later theoretically and experimentally show that our approach can stabilize the MEC system.

Remark: We consider a distributed queuing model rather than a centralized one, i.e., all edge servers maintain a single queue, for two main reasons. First, the distributed queuing architecture is more common in data centers since the queue memory operates at a much slower speed [8]. Secondly, as the central queue might be far away from users, it would incur extra communication delay, which is not acceptable in the MEC environment.

3 PROBLEM FORMULATION

We now identify several key MEC components that are taken into account in our model, including the performance (benefit gained from system throughput), and the costs (incurred by queuing delay and cloud-to-user latency).

3.1 Time-Average Throughput

From the service provider's perspective, the overall system throughput, i.e., the number of users allocated to edge servers, is one of the key performance metrics that reflects the quality of service. We define the time-average throughput r_s of each edge server s as follows:

$$r_s = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left\{ \sum_{u \in \mathcal{A}(t)} \mathbf{a}_{u,s}(t) \right\}, \forall s \in \mathcal{S} \quad (5)$$

We have $\sum_{s \in \mathcal{S}} r_s$ as the overall throughput of the MEC system, which ideally should be maximized and is obviously subject to the following constraint: $\sum_{s \in \mathcal{S}} r_s \leq \lambda$, i.e., the time-average throughput cannot exceed the time-average user arrival rate. The time-average benefit that a service provider gains from system throughput is measured by:

$$\bar{b}_r = \sum_{s \in \mathcal{S}} (w_s r_s) \quad (6)$$

where w_s is a non-negative weight for the throughput r_s of each edge server $s \in \mathcal{S}$. This weight enables the service provider to adjust the benefit gained from different edge servers' throughput. A high value of w_s indicates that, given the same throughput, the benefit gained from edge server s is more significant than other edge servers with lower w_s . For example, if the prices for the computing resources on edge server s are cheaper than those on other edge servers, the throughput on edge server s is more beneficial or profitable.

3.2 Queuing Delay and Latency Cost

For each user, there are two possible allocation options, either the remote cloud or an edge server. Each option is associated with a type of cost. If allocated to an edge server, the user is placed in a queue waiting for its turn to be served since being allocated to an edge server is more desirable, hence highly demanding. This incurs a *queuing delay cost*. If allocated to the remote cloud, which has an ample volume of computing power to serve many users simultaneously, the user skips the queuing but suffers a high cloud latency throughout its service session. This incurs a *latency cost*.

3.2.1 Queuing Delay Cost

The queuing delay cost for a user depends on the current congestion state of its assigned queue and the computing power of the edge server associated with that queue. For example, a more congested queue would apparently lead to a longer queuing delay. However, one also needs to take into account the computing capacity of the edge server associated with that queue. If the edge server's computing capacity is high, it can serve more users simultaneously, resulting in a higher service rate than those with low computing capacities.

The service rate of an edge server s is N_s/ℓ , where ℓ is the expected user session length, measured by the number

of time slots, and N_s is the maximum number of users of all types that can be simultaneously served by edge server s in one time slot. In practice, ℓ can be empirically estimated based on historical data. N_s can be calculated based on the computing capacity C_s of edge server s . For example, say there are three possible types of user resource requirements $\langle 1, 3, 1, 2 \rangle$, $\langle 2, 1, 3, 1 \rangle$, and $\langle 3, 1, 2, 2 \rangle$, and edge server s has a capacity of $\langle 43, 43, 41, 41 \rangle$. This edge server s has enough capacity to serve 22 users concurrently (e.g., 10 users of type $\langle 1, 3, 1, 2 \rangle$, 5 users of type $\langle 2, 1, 3, 1 \rangle$, and 7 users of type $\langle 3, 1, 2, 2 \rangle$).

Let $n_s(t)$ denote the number of users being served by edge server s in time slot t . Then, the queuing delay of a user u who has just arrived at edge server s can be measured by $\frac{[n_s(t) - N_s + Q_s(t) + 1]_+}{N_s/\ell}$, where $Q_s(t)$ is the current queue length of edge server s , i.e., the number of queuing users excluding new user arrivals. The estimated queuing delay of another user who arrived right after user u is thus $\frac{[n_s(t) - N_s + Q_s(t) + 2]_+}{N_s/\ell}$. Intuitively, $[n_s(t) - N_s + Q_s(t) + i]_+$, $\forall i \leq \mathbf{a}_s(t)$ represents the number of users queuing ahead of the new users when the edge server is full, including the users currently being served by the edge server. $[n_s(t) - N_s + Q_s(t) + i]_+ = 0$ if the server has sufficient resources to serve new users right away without queuing them. We can see that the queuing delay of users allocated to an edge server is proportional to the current queue length $Q_s(t)$ and the number of users allocated to that edge server $\mathbf{a}_s(t)$. Let $M_s(t) = n_s(t) - N_s + Q_s(t)$, the time-average queuing delay for all users in all edge servers can be defined as follows:

$$\bar{c}_d = \sum_{s \in \mathcal{S}} \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left\{ \frac{[M_s(t) + 1]_+}{N_s/\ell} + \dots + \frac{[M_s(t) + \mathbf{a}_s(t)]_+}{N_s/\ell} \right\} \quad (7)$$

3.2.2 Latency Cost

If allocated to the remote cloud, the user suffers from high latency throughout its service session. The latency, or communication delay, is influenced by many factors such as transmission medium, distance, bandwidth, etc. To simplify the model, we let h_u be the cloud-to-user latency for user u . The time-average latency of all users allocated to the remote cloud is:

$$\bar{c}_h = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left\{ \sum_{u \in \mathcal{A}(t)} (\mathbf{b}_u(t) h_u) \right\} \quad (8)$$

3.3 Time-Average System Benefit Maximization

In the previous sections, we have modeled the throughput benefit \bar{b}_r , the queuing delay cost \bar{c}_d , and the latency cost \bar{c}_h . Now, we formulate the maximization of the time-average system benefit as the following stochastic optimization problem:

$$(\mathbf{P1}) \quad \max_{\mathbf{a}_{u,s}, \mathbf{b}_u} \omega_r \bar{b}_r - \omega_d \bar{c}_d - \omega_h \bar{c}_h \quad (9)$$

s.t. (1), (2), (3)

where ω_r, ω_d , and ω_h are non-negative weights acting as normalizing parameters for throughput benefit, queuing delay cost, and latency cost, respectively, since they have

different scales. This also allows service providers to adjust their priority for the throughput benefit and other costs. Solving **P1** in an offline and centralized fashion is intractable. Optimally solving this optimization problem requires complete offline future information such as user arrivals and departures, user locations, and user requirements. In the highly stochastic MEC environment, that information are often unpredictable and vary over time. Therefore, we tackle this challenge by introducing an online algorithm based on the Lyapunov optimization framework, which can make decisions in every time slot without future information.

4 STOCHASTIC USER ALLOCATION

In this section, we introduce the trade-off between the system benefit and system stability, measured by the queue congestion. We then propose SUAC, an online algorithm based on the Lyapunov optimization framework to solve the stochastic EUA problem.

4.1 Stability-Benefit Trade-off

Let $Q(t) = (Q_1(t), \dots, Q_S(t))$ be the queue backlog vector. For each time slot t , we define a quadratic Lyapunov function $L(Q(t))$ as:

$$L(Q(t)) \triangleq \frac{1}{2} \sum_{s \in \mathcal{S}} Q_s(t)^2 \quad (10)$$

This function is a scalar measure of the congestion state of the MEC system. Intuitively, $L(Q(t))$ is large when at least one of the edge server's queue $Q_s(t)$ is congested, and $L(Q(t))$ is small when all the queue backlogs are small, representing a stable system. Given the Lyapunov function, we define the *conditional Lyapunov drift* to measure the change in this function between two consecutive time slots:

$$\Delta(Q(t)) \triangleq \mathbb{E}\{L(Q(t+1)) - L(Q(t)) | Q(t)\} \quad (11)$$

Our objective is to find a user allocation strategy $\mathbf{a}(t)$ to coordinate the queue congestion state, throughput benefit, and other related costs in every time slot. By incorporating the queue stability into the performance-cost trade-off, we come up with a *drift-minus-benefit* expression as follows:

$$\Delta(Q(t)) - V \mathbb{E}\{\omega_r \bar{b}_r - \omega_d \bar{c}_d - \omega_h \bar{c}_h | Q(t)\} \quad (12)$$

where V is a non-negative control parameter to balance the trade-off between the drift $\Delta(Q(t))$ (queue stability) and the system benefit (throughput benefit minus queuing delay and latency costs). $V = 0$ indicates that the system should be stabilized as much as possible regardless of the throughput benefit, i.e., to allocate all users to the remote cloud. Depending on the situation, a service provider can flexibly change the value of V to adjust the trade-off. For example, they can decrease V to keep the queue backlogs small, avoiding system congestion while maximizing the system benefit as much as possible. Under the Lyapunov optimization framework, the user allocation strategy should be chosen to minimize the supremum bound of the above drift-minus-benefit expression (12).

Lemma 1. *Given any allocation strategies in any time slots, the following bound of the drift-minus-benefit (12) holds:*

$$\begin{aligned} & \Delta(Q(t)) - V \mathbb{E}\{\omega_r \bar{b}_r - \omega_d \bar{c}_d - \omega_h \bar{c}_h | Q(t)\} \\ & \leq B + \sum_{s \in \mathcal{S}} \mathbb{E}\left\{\mathbf{a}_s(t)^2 V K + \mathbf{a}_s(t) P_s(t) | Q(t)\right\} \\ & \quad + \sum_{u \in \mathcal{A}(t)} \mathbb{E}\left\{V \omega_h h_u \left(1 - \sum_{s \in \mathcal{S}} \mathbf{a}_{u,s}(t)\right) | Q(t)\right\} \end{aligned} \quad (13)$$

where $B = \frac{1}{2}(\sum_{s \in \mathcal{S}} N_s^2 + A^{max^2})$ is a constant, $\mathbf{a}_s(t) = \sum_{u \in \mathcal{A}(t)} \mathbf{a}_{u,s}(t)$, $M_s(t) = n_s(t) - N_s + Q_s(t)$, $P_s(t) = Q_s(t) + V(K(2M_s(t) + 1) - \omega_r w_s)$, and $K = \frac{\omega_d \ell}{2N_s}$.

Proof. Let $\mathbf{a}_s(t) = \sum_{u \in \mathcal{A}(t)} \mathbf{a}_{u,s}(t)$. Since $([a - b]_+ + c)^2 \leq a^2 + b^2 + c^2 - 2a(b - c)$, $\forall a, b, c \geq 0$, we can derive:

$$\begin{aligned} & \Delta(Q(t)) = \mathbb{E}\{L(Q(t+1)) - L(Q(t)) | Q(t)\} \\ & = \mathbb{E}\left\{\frac{1}{2} \sum_{s \in \mathcal{S}} \left(\left([Q_s(t) - D_s(t)]_+ + \mathbf{a}_s(t)\right)^2 - Q_s(t)^2\right) | Q(t)\right\} \\ & \leq \mathbb{E}\left\{\frac{1}{2} \sum_{s \in \mathcal{S}} \left(D_s(t)^2 + \mathbf{a}_s(t)^2\right.\right. \\ & \quad \left.\left. - 2Q_s(t)(D_s(t) - \mathbf{a}_s(t))\right) | Q(t)\right\} \end{aligned} \quad (14)$$

Since $\sum_{s \in \mathcal{S}} \mathbf{a}_s(t) \leq A^{max}$ and $\mathbf{a}_s(t) \geq 0$, $\forall s \in \mathcal{S}$, we have $\sum_{s \in \mathcal{S}} \mathbf{a}_s(t)^2 \leq A^{max^2}$. In addition, $D_s(t) \leq N_s$, $\forall s \in \mathcal{S}$, $\forall t$, thus $\sum_{s \in \mathcal{S}} (D_s(t)^2 + \mathbf{a}_s(t)^2)$ is bounded by $\sum_{s \in \mathcal{S}} N_s^2 + A^{max^2}$. Also, $-2Q_s(t)D_s(t) \leq 0$, $\forall s \in \mathcal{S}$, $\forall t$, therefore,

$$\Delta(Q(t)) \leq B + \mathbb{E}\left\{\sum_{s \in \mathcal{S}} (\mathbf{a}_s(t) Q_s(t)) | Q(t)\right\} \quad (15)$$

where $B = \frac{1}{2}(\sum_{s \in \mathcal{S}} N_s^2 + A^{max^2})$ is a constant.

By subtracting $V \mathbb{E}\{\omega_r \bar{b}_r - \omega_d \bar{c}_d - \omega_h \bar{c}_h | Q(t)\}$ from both sides of (15), we have:

$$\begin{aligned} & \Delta(Q(t)) - V \mathbb{E}\{\omega_r \bar{b}_r - \omega_d \bar{c}_d - \omega_h \bar{c}_h | Q(t)\} \\ & \leq B + \mathbb{E}\left\{\sum_{s \in \mathcal{S}} (\mathbf{a}_s(t) Q_s(t)) | Q(t)\right\} - V \mathbb{E}\left\{\omega_r \sum_{s \in \mathcal{S}} (w_s r_s) \right. \\ & \quad \left. - \omega_d \sum_{s \in \mathcal{S}} \left(\frac{[M_s(t) + 1]_+}{N_s/\ell} + \dots + \frac{[M_s(t) + \mathbf{a}_s(t)]_+}{N_s/\ell}\right) \right. \\ & \quad \left. - \omega_h \sum_{u \in \mathcal{A}(t)} (\mathbf{b}_u(t) h_u) | Q(t)\right\} \\ & \stackrel{\dagger}{\leq} B + \mathbb{E}\left\{\sum_{s \in \mathcal{S}} (\mathbf{a}_s(t) Q_s(t)) | Q(t)\right\} - V \mathbb{E}\left\{\sum_{s \in \mathcal{S}} (\omega_r w_s \mathbf{a}_s(t)) \right. \\ & \quad \left. - \sum_{s \in \mathcal{S}} \frac{\omega_d \ell (2M_s(t) + 1 + \mathbf{a}_s(t)) \mathbf{a}_s(t)}{2N_s} \right. \\ & \quad \left. - \omega_h \sum_{u \in \mathcal{A}(t)} \left(h_u \left(1 - \sum_{s \in \mathcal{S}} \mathbf{a}_{u,s}(t)\right)\right) | Q(t)\right\} \\ & = B + \sum_{s \in \mathcal{S}} \mathbb{E}\left\{\mathbf{a}_s(t)^2 V K \right. \\ & \quad \left. + \mathbf{a}_s(t) \left(Q_s(t) + V(K(2M_s(t) + 1) - \omega_r w_s)\right) | Q(t)\right\} \\ & \quad + \sum_{u \in \mathcal{A}(t)} \mathbb{E}\left\{V \omega_h h_u \left(1 - \sum_{s \in \mathcal{S}} \mathbf{a}_{u,s}(t)\right) | Q(t)\right\} \end{aligned} \quad (16)$$

where $K = \frac{\omega_d \bar{c}_d}{2N_s}$ and $M_s(t) = n_s(t) - N_s + Q_s(t)$. The inequality \dagger is because of the constraint (1) and the fact that $[a+1]_+ + \dots + [a+n]_+ \geq na + n(n+1)/2, \forall a \in \mathbb{R}, \forall n \geq 0$. Therefore, Lemma 1 holds. \square

We have now transformed the stochastic optimization problem **P1** into the bounding of the drift-minus-benefit. In the next section, we propose an online allocation algorithm given the drift-minus-benefit bound found in Lemma 1.

Remark: This model is easily extensible by incorporating one or more virtual queues into (10). A virtual queue could be a deficit queue of energy consumption, operating budget, etc., allowing service providers to enforce more constraints.

4.2 Stochastic User Allocation Algorithm

In this section, we propose SUAC, a Stochastic User Allocation algorithm (Algorithm 1), which observes the state of the MEC system in every time slot t and determines an allocation strategy $\mathbf{a}(t)$ to minimize the supreme bound on the drift-minus-benefit (12), i.e., the right-hand side of (13). Employing the concept of opportunistically minimizing an expectation, this can be achieved by solving the optimization problem **P2** below.

$$\begin{aligned} (\text{P2}) \quad \min_{\mathbf{a}_{u,s}, \mathbf{b}_u} \quad & \sum_{s \in \mathcal{S}} \left(\mathbf{a}_s(t)^2 V K + \mathbf{a}_s(t) P_s(t) \right) \\ & + \sum_{u \in \mathcal{A}(t)} \left(V \omega_h h_u \left(1 - \sum_{s \in \mathcal{S}} \mathbf{a}_{u,s}(t) \right) \right) \quad (17) \\ \text{s.t.} \quad & (1), (2), (3) \end{aligned}$$

Algorithm 1 SUAC ALGORITHM

Input: $\mathcal{S}, V, \omega_r, \omega_d, \omega_h$
Output: allocation decisions $\mathbf{a}_{u,s}(t), \mathbf{b}_u(t), \forall t, \forall s \in \mathcal{S}$
 1: for each time slot $t = 0, 1, \dots, \infty$ do
 2: Observe incoming users $\mathcal{A}(t)$ and edge servers' queues $Q_s(t), \forall s \in \mathcal{S}$;
 3: Choose $\mathbf{a}_{u,s}(t), \mathbf{b}_u(t), \forall u \in \mathcal{A}(t)$ by solving **P2**;
 4: Update the queue $Q_s, \forall s \in \mathcal{S}$ according to (4);
 5: end for

In every time slot, new users arrive and then get allocated to either the remote cloud or edge servers depending on the solutions to optimization problem **P2**. In the meantime, each edge server serves the users waiting in its queue if it has sufficient computing resources. After a random number of time slots, those users depart and release the computing resources so that the queuing users can start using the service on a first-come, first-served (FCFS) basis. Note that SUAC requires neither the knowledge of future user arrivals nor general statistical user distributions. Given the pre-defined parameters $V, \omega_r, \omega_d, \omega_h$, SUAC can achieve a controllable trade-off between throughput and other associated costs while guaranteeing a stable MEC system. In other words, all the edge servers' queues are stable at all times, bounding users' expected queuing delay. Problem **P2** can be solved using an integer programming solver, e.g., IBM ILOG CPLEX Optimizer¹ or Gurobi²). According to

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

our experiments, IBM ILOG CPLEX Optimizer can handle as many as 1,000 users arriving at 26 edge servers in each 30-second time slot.

SUAC is an online algorithm that allocates users as they arrive in the MEC system. When a user moves within the same edge server's coverage area across two time slots, they will not be considered as a new user. If they move from one edge server's coverage area into another edge server's coverage area across two time slots, they will be allocated as a new user in the second time slot. In this way, SUAC can accommodate user mobility. Within one time slot, we study the quasi-static scenarios where users do not move across edge servers, similar to many other studies [3], [4], [13], [16].

The following theorem demonstrates the existence of a performance-cost trade-off $[O(1/V), O(V)]$ in the proposed SUAC algorithm, allowing service providers to adjust the control parameter V to achieve its desired trade-off between the throughput benefit and other associated costs, namely queuing delay and latency costs.

Theorem 1. For any user arrival rate in any time slot, employing SUAC with any non-negative V satisfies the following performance bounds:

1) The time-average system benefit is within a gap (B/V) to the optimal solution:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \{ \omega_r \bar{b}_r - \omega_d \bar{c}_d - \omega_h \bar{c}_h \} \geq \beta^* - \frac{B}{V} \quad (18)$$

2) The average queue backlog is upper bounded:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{s \in \mathcal{S}} \mathbb{E} \{ Q_s(t) \} \leq B + V(\beta^* - \beta^{\min}) \quad (19)$$

where $\beta^* = \omega_r \bar{b}_r^* - \omega_d \bar{c}_d^* - \omega_h \bar{c}_h^*$, and \bar{b}_r^*, \bar{c}_d^* , and \bar{c}_h^* are the optimal values of Problem **P2**, and $B = \frac{1}{2}(\sum_{s \in \mathcal{S}} N_s^2 + A^{\max})$, and β^{\min} is defined in the proof.

Proof. First, we prove the first part of Theorem 1. Using the result obtained in Theorem 4.5 in [15], we can show that there exists a stationary randomized allocation policy Φ for **P2** that determines feasible allocation strategies $\mathbf{a}_{u,s}^\Phi(t), \mathbf{b}_u^\Phi(t), \forall u \in \mathcal{A}(t), \forall s \in \mathcal{S}, \forall t$, independent of the current queue backlogs $Q_s(t), \forall s \in \mathcal{S}$ in every time slot t , and yields the following steady state values:

$$\begin{aligned} \mathbb{E} \{ \mathbf{a}_s^\Phi(t) \} &= \mathbb{E} \left\{ \sum_{u \in \mathcal{A}(t)} \mathbf{a}_{u,s}^\Phi(t) \right\} = r_s^*, \\ \mathbb{E} \{ \mathbf{a}_s^\Phi(t) (\mathbf{a}_s^\Phi(t) + 2M_s(t) + 1) \} &\leq \bar{c}_d^*, \\ \mathbb{E} \left\{ \sum_{u \in \mathcal{A}(t)} \left((1 - \sum_{s \in \mathcal{S}} \mathbf{a}_{u,s}^\Phi(t)) h_u \right) \right\} &= \bar{c}_h^* \quad (20) \end{aligned}$$

Let $\beta = \omega_r \bar{b}_r - \omega_d \bar{c}_d - \omega_h \bar{c}_h$ denote the system benefit (Section 3.3) in time slot t . For all feasible allocation solutions, which includes those produced by Φ , the proposed

SUAC algorithm minimizes the drift-minus-benefit bound proven in Lemma 1, thus the following inequality holds:

$$\begin{aligned} & \Delta(Q(t)) - V \mathbb{E}\{\beta(t)|Q(t)\} \\ & \leq B + \sum_{s \in \mathcal{S}} \mathbb{E} \left\{ \mathbf{a}_s^\Phi(t)^2 V K + \mathbf{a}_s^\Phi(t) P_s(t) | Q(t) \right\} \\ & \quad + \sum_{u \in \mathcal{A}(t)} \mathbb{E} \left\{ V \omega_h h_u (1 - \sum_{s \in \mathcal{S}} \mathbf{a}_{u,s}^\Phi(t)) | Q(t) \right\} \end{aligned} \quad (21)$$

Taking expectations of the above inequality and using the law of iterated expectations give us:

$$\begin{aligned} & \mathbb{E}\{L(Q(t+1))\} - \mathbb{E}\{L(Q(t))\} - V \mathbb{E}\{\beta(t)\} \\ & \leq B + \sum_{s \in \mathcal{S}} \mathbb{E} \left\{ \mathbf{a}_s^*(t)^2 V K + \mathbf{a}_s^*(t) P_s(t) | Q(t) \right\} \\ & \quad + \sum_{u \in \mathcal{A}(t)} \mathbb{E} \left\{ V \omega_h h_u (1 - \sum_{s \in \mathcal{S}} \mathbf{a}_{u,s}^*(t)) | Q(t) \right\} \end{aligned} \quad (22)$$

By plugging (20) into the right-hand side of (22) and rearranging the terms, we have:

$$\begin{aligned} \mathbb{E}\{L(Q(t+1))\} - \mathbb{E}\{L(Q(t))\} - V \mathbb{E}\{\beta(t)\} & \leq B - V \beta^* \\ & \quad + \sum_{s \in \mathcal{S}} \mathbb{E}\{\mathbf{a}_s^*(t) Q_s(t) | Q(t)\} \end{aligned} \quad (23)$$

The above holds for all $t \in \{0, 1, 2, \dots\}$. By summing the above inequality over $t \in \{0, 1, \dots, T-1\}$ for some integer $T > 0$ and applying the law of telescoping sums, we have:

$$\begin{aligned} \mathbb{E}\{L(Q(T))\} - \mathbb{E}\{L(Q(0))\} - V \sum_{t=0}^{T-1} \mathbb{E}\{\beta(t)\} & \leq TB - VT \beta^* \\ & \quad + \sum_{t=0}^{T-1} \sum_{s \in \mathcal{S}} \mathbb{E}\{\mathbf{a}_s^*(t) Q_s(t) | Q(t)\} \end{aligned} \quad (24)$$

Since $L(Q(0)) = 0$, $L(Q(T)) \geq 0$, and $\mathbf{a}_s^*(t) Q_s(t) \geq 0$, $\forall s \in \mathcal{S}, \forall t$, dividing both side of the above inequality by T yields:

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\beta(t)\} \geq \beta^* - \frac{B}{V} \quad (25)$$

The proof of the first part of Theorem 1 completes by letting $T \rightarrow \infty$.

Next, we prove the second part of Theorem 1. Suppose there are constants $B \geq 0$, $V \geq 0$, $\epsilon \geq 0$, and β^* such that for all time slot t and all possible values of $Q(t)$, we have:

$$\Delta(Q(t)) + V \mathbb{E}\{\beta(t)|Q(t)\} \leq B + V \beta^* - \epsilon \sum_{s \in \mathcal{S}} Q_s(t) \quad (26)$$

Taking expectations of both sides, summing over $t \in \{0, 1, \dots, T-1\}$, and applying the law of iterated expectations yield:

$$\begin{aligned} & \mathbb{E}\{L(Q(T))\} - \mathbb{E}\{L(Q(0))\} + V \sum_{t=0}^{T-1} \mathbb{E}\{\beta(t)|Q(t)\} \\ & \leq T(B + V \beta^*) - \epsilon \sum_{t=0}^{T-1} \sum_{s \in \mathcal{S}} \mathbb{E}\{Q_s(t)\} \end{aligned} \quad (27)$$

Assume the expected system benefit $\beta(t)$ is lower bounded by a finite value β^{min} so that we have $\mathbb{E}\{\beta(t)\} \geq$

β^{min} for all possible allocation decisions. As $L(Q(0)) = 0$ and $L(Q(T)) > 0$, plugging this into the above inequality and rearranging terms yield:

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{s \in \mathcal{S}} \mathbb{E}\{Q_s(t)\} \leq \frac{B + V(\beta^* - \beta^{min})}{T} \quad (28)$$

Letting $T \rightarrow \infty$ completes the proof of the second part of Theorem 1. \square

5 EVALUATION

We have performed a series of experiments to verify and evaluate the performance of our proposed SUAC algorithm.

5.1 Experiment Setup

Edge servers: We use the EUA dataset³ [6], which contains the geographic locations of end-users and all cellular base stations in Australia. Then, we simulate a highly dense 500m×500m area covered by 26 base stations, assuming each base station is equipped with an edge server. The coverage radius of each edge server is randomly generated within a range of 100-150m. The edge server capacities are randomly generated following a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the average capacity of each resource type (CPU, RAM, storage, and bandwidth), and the standard deviation $\sigma = 10$ for all experiments conducted in this article. Since a normal distribution might contain negative numbers, any negative amount of computing resources generated is rounded up to 1. We set $w_s = 1$ for all edge servers $s \in \mathcal{S}$ so that the benefit gained from the same throughput would be equal among all edge servers.

Edge users: All edge servers are able to serve $N = \sum_{s \in \mathcal{S}} N_s$ users simultaneously. The number of newly-arrived users in each time slot $|\mathcal{A}(t)|$ is drawn from a Poisson distribution with rate $[0, \zeta N]$, where $\zeta \in (0, 0.1]$ controls the traffic intensity. We assume $\mathcal{R} = \{CPU, RAM, storage, bandwidth\}$. Each resource requirement is a $|\mathcal{R}|$ -dimensional vector, where each vector component is the normalized amount of a resource type in \mathcal{R} . Each user's resource requirement is randomly generated using a uniform distribution within $\langle 1, 1, 1, 1 \rangle$ and $\langle 4, 4, 4, 4 \rangle$. Each user's session duration is uniformly distributed in $[10, 20]$ time slots. Each time slot is set at 30-second length. The latency experienced by a user, if allocated to the remote cloud, is randomly set in $[50, 250]$ ms [17].

All the experiments are conducted on a Windows machine equipped with Intel Core i5-7400T processor (4 CPUs, 2.4GHz) and 8GB RAM. The optimization problem **P2** is solved with IBM ILOG CPLEX Optimizer⁴ in line 3 of Algorithm 1.

5.2 Performance Benchmark

We evaluate SUAC against the state of the art and two baseline approaches:

- **Join-the-Shortest-Queue (JSQ):** The authors of [8], [9] propose a class of randomized algorithms for placing VMs in physical servers that can achieve

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

4. www.ibm.com/analytics/cplex-optimizer

TABLE 2: Experiment sets

	μ	ζ	V	ω_r	ω_d	ω_h
Set #1	40	0.01, 0.02, ..., 0.1	0.3	300	7	200
Set #2	10, 20, ..., 90	0.07	0.3	300	7	200
Set #3	40	0.07	0.05, 0.1, 0.15, ..., 0.45	200, 300, 400, 500	7	200
Set #4	40	0.07	0.05, 0.1, 0.15, ..., 0.45	300	3, 5, 7, 9	200
Set #5	40	0.07	0.05, 0.1, 0.15, ..., 0.45	300	7	50, 200, 350, 500

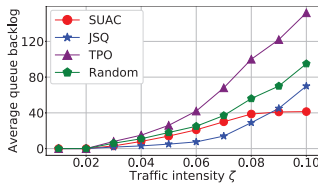
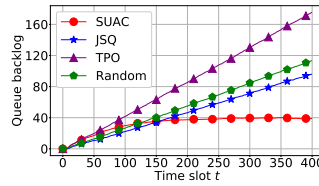
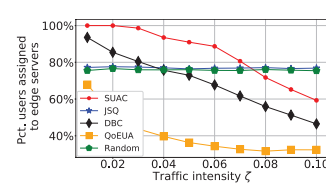

 Fig. 2: Average queue backlog vs. varying traffic intensity ζ under four algorithms (Set #1).

 Fig. 3: Queue backlog evolution under four algorithms with $\zeta = 0.3$ (Set #1).


Fig. 4: Percentage of users allocated to edge servers (Set #1).

maximum throughput without preemptions in a cloud computing environment. The users and edge servers in our problem can be seen as VMs and physical servers in their problems. Since our problem has a third tier (the remote cloud server), we modify their approach to incorporate this extension. We consider the remote cloud server as a "virtual" edge server without a queue. Suppose a user u arrives at time slot t , it can be allocated to the cloud with probability $\Pr(\mathbf{b}_u = 1) = \Pr(\mathbf{a}_{u,1} = 1) = \dots = \Pr(\mathbf{a}_{u,S} = 1)$. If not allocated to the cloud, it will be allocated to the edge server with the shortest queue. Each queue maintains a Poisson clock to control when to serve the queuing users.

- Density-based Clustering (DBC):** In [18], users are allocated to cloudlets (equivalent to edge servers in our work) using a density-based clustering algorithm that takes into account the distances between users and their neighbor edge servers. Users are to be allocated to an edge server that has the most candidate users first, i.e., users that are covered by the server. For example, say users u_1 and u_2 can be allocated to either edge servers s_1 or s_2 . Edge server s_1 covers more users than edge server s_2 . User u_1 is closer to s_1 than it is to s_2 , and u_2 is closer to s_2 than it is to s_1 . Hence, the users allocation will start from s_1 since it has more candidate users. User u_1 has a higher allocation priority than u_2 when being assigned to s_1 . Since this approach does not take queuing system into account, we assume that excessive users, i.e., users that are allocated to edge servers but the edge servers do not have sufficient computing resources to serve the users, will be redirected to the remote cloud.
- QoE-aware User Allocation (QoEUA):** In [7], the authors map each user resource requirement to a quality-of-experience (QoE) level. They optimally allocate users to edge servers using an integer pro-

gramming approach so that the total QoE of all the users is maximized. Similar to DBC, this approach does not take queuing system into account. Thus, we assume that excessive users will be redirected to the remote cloud.

- Throughput Optimal (TPO):** To achieve the maximum throughput benefit, this approach completely ignores the remote cloud. Every user will be allocated to the edge server with the shortest queue. Predictably, this approach may well result in a high queuing delay cost.
- Random:** New users are uniformly allocated to either the cloud or edge servers at random.

5.3 Experiment Sets

We conduct a series of experiments with different varying parameters to analyze the performance of SUAC in various MEC scenarios. Table 2 summarizes all the experiment sets which will be discussed in the next section. We first experiment with various traffic intensities (ζ) in Set #1, ranging from light traffic to very intense traffic, to simulate different user arrival rates. In Set #2, we vary the computing resource capacity (μ) that each edge server has to serve users, ranging from scarce resources to abundant resources; this impacts the service rate of each edge server. After that, we vary the control parameter V used in Eq. (17), and at the same time, the throughput benefit weight ω_r , queuing delay cost weight ω_d , and latency cost weight ω_h in Sets #3, #4, and #5, respectively. These domain-specific parameters are used in Eq. (17) to indicate the priorities for SUAC's pursuit of system throughput, queuing delay, and latency, respectively, when SUAC allocates users to edge servers over time. In a real-world setting, a service provider can determine those parameters according to their needs. Note that in those three experiment sets, parameters $V, \omega_r, \omega_d, \omega_h$ have no impact on the allocation results produced by the five benchmark approaches since they work independently of those parameters. Each experiment setting is executed for

a duration of 1,000 time slots.

5.4 Experiment Results

5.4.1 Impact of Traffic Intensity (User Arrival Rate)

In Set #1, we vary the traffic intensity ζ from 0.01, 0.02 to as high as 0.1. For example, say the 26 edge servers can serve 500 users simultaneously, $\zeta = 0.1$ means that there are 50 new users joining the system in every time slot on average. Fig. 2 illustrates the time-average queue backlog under four approaches when the MEC system is put under various traffic intensities. The time-average queue backlog is the average number of users queuing per server during the simulated duration. Fig. 3 depicts the evolution of the queue backlog in Set #1 when $\zeta = 0.3$ during the first 400 time slots. Note that DBC and QoEUA are not shown in those two figures because they do not employ a queuing system, or have no queues.

The time-average queue backlogs under JSQ, TPO, and Random (Fig. 2) grow linearly with the traffic intensity. At $\zeta = 0.1$, the average queue backlog of TPO is very long, i.e., roughly 140 users queuing per server, which is unacceptable for any latency-sensitive services. This is unsurprising since TPO only allocates all users to edge servers. On the other hand, under SUAC, the time-average queue backlog slowly increases with the increasing traffic intensity, then converges and remains unchanged at approximately 40 users regardless of the varying traffic intensity ($\zeta = 0.08 - 0.1$). A service provider can easily increase or decrease the queue backlog level by increasing or decreasing the control parameter V . In Fig. 3, the queue backlogs of all four approaches gradually increase during the first few time slots. After that, while the other three approaches keep getting their queues more congested, SUAC stops allocating too many users to edge servers and starts directing them to the remote cloud, stabilizing the MEC system instead of overloading it. We noticed the same phenomenon in all other experiments, which are not presented here for brevity. This strongly demonstrates the ability of SUAC to stabilize the MEC system under any traffic conditions.

Fig. 4 shows the percentage of users allocated to edge servers (the higher the better). The numbers of users allocated to edge servers by SUAC, DBC, and QoEUA decrease as we increase the traffic intensity ζ . This is expected because the edge servers can only serve up to a certain number of users and the rest have to be allocated to the remote cloud. Otherwise, the queues would be heavily congested. The numbers of users allocated to edge servers by JSQ and Random remain constant since they decide if a user is allocated to the remote cloud by using the same randomness factor. Under some experiment settings ($\zeta = 0.08 - 0.1$), JSQ and Random manage to allocate more users to edge servers than SUAC. However, SUAC still beats them in terms of system benefit because under JSQ and Random, the queuing delay cost outweighs the throughput benefit. Under all other experiment settings, SUAC significantly outperforms all other approaches. TPO is not presented here because it does not allocate users to the remote cloud.

Fig. 5 visualizes the normalized time-average system benefit gained by the six approaches under different traffic intensities. At the very beginning, all four approaches achieve relatively equal performance since the traffic was

very light, thus all the users could be allocated to edge servers without queuing. As more users arrive, SUAC starts to significantly outperform the other three approaches since they suffer from a very high queuing delay cost (JSQ, TPO, and Random) or cloud latency cost (DBC and QoEUA). The throughput benefit and queuing delay cost produced by SUAC remain unchanged when increasing traffic intensity since all the queues are kept stabilized as discussed above. To stabilize the system under intense traffic conditions, SUAC directs new users to the remote cloud, hence the considerable increase in the cloud latency cost, which in turn results in the loss of system benefit. We can infer that in order to deal with an increasing user arrival rate, a service provider needs to hire more edge computing resources to increase the service rate.

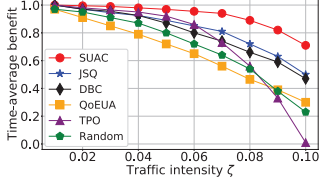
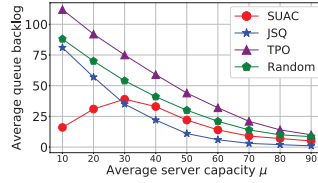
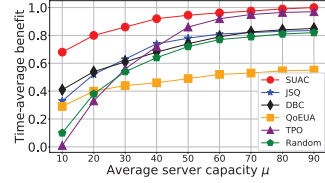
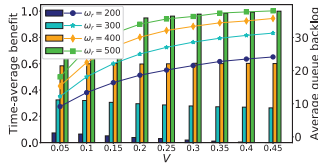
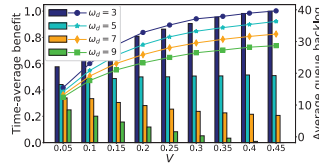
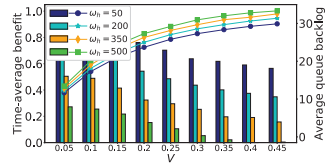
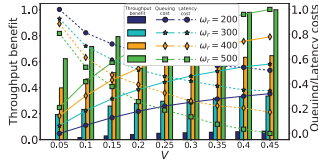
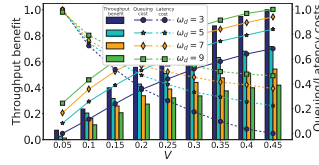
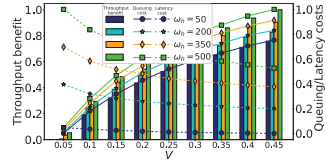
5.4.2 Impact of Edge Server Capacity (Service Rate)

In this section, we evaluate the impact of service rate (Set #2), which is determined by the amount of computing resources available on edge servers. An increase in the average edge server capacity eventually leads to an increase in the service rate of the system, meaning an edge server can hold more users in its queue without increasing the queuing delay. This is demonstrated in Fig. 6, as the average server capacity μ increases from 10 to 30, the average queue backlog under SUAC also gradually increases, starting from around 10 users to almost 40 users per edge server's queue. In other words, SUAC can allocate more users to edge servers only when it is safe to do so given the current service rate. From $\mu = 30$ onward, its average queue backlog steadily decreases since the service rate is now relatively high, which allows edge servers to serve more users simultaneously. DBC and QoEUA do not employ a queuing system so they are presented in this figure.

In contrast, the other three approaches (JSQ, TPO, and Random) work independently of the service rate. As a result, increasing the service rate will lessen the stress on the queue backlogs under those approaches. In the figure, we can see that the average queue backlogs of JSQ, TPO, and Random gradually decrease as the average server capacity increases. We can predict that when all the edge servers have an abundant amount of computing resources, the queue backlogs under all four approaches will eventually converge to close to zero. However, as aforementioned, such resource-abundant situations are extremely unlikely to happen in the MEC environment. In such cases, joint load balancing (i.e., SUAC) and dynamic management of MEC resources, or any other simple approaches, can actually control queue backlog effectively.

Fig. 7 visualizes the corresponding normalized time-average system benefit in the experiment analyzed above. Again, SUAC clearly outperformed all other approaches under any experiment setting. For the same reason discussed above, the time-average system benefit of all approaches will eventually converge once the service provider is able to hire an excessive amount of edge computing resources, which would be highly expensive and unlikely to happen in any real-world scenarios. JSQ and Random have a randomness factor so there will always be some users allocated to the cloud despite a large amount of computing resources. DBC does not employ a queuing system so some users will

10


 Fig. 5: Time-average system benefit vs. varying traffic intensity ζ under four algorithms (Set #1).

 Fig. 6: Average queue backlog vs. varying server capacity μ under four algorithms (Set #2).

 Fig. 7: Time-average system benefit vs. varying server capacity μ under four algorithms (Set #2).

 Fig. 8: Time-average system benefit vs. varying values of V under SUAC with different values of ω_r (Set #3).

 Fig. 9: Time-average system benefit vs. varying values of V under SUAC with different values of ω_d (Set #4).

 Fig. 10: Time-average system benefit vs. varying values of V under SUAC with different values of ω_h (Set #5).

 Fig. 11: Throughput benefit, queuing delay cost, and latency cost vs. varying values of V under SUAC with different values of ω_r (Set #3).

 Fig. 12: Throughput benefit, queuing delay cost, and latency cost vs. varying values of V under SUAC with different values of ω_d (Set #4).

 Fig. 13: Throughput benefit, queuing delay cost, and latency cost vs. varying values of V under SUAC with different values of ω_h (Set #5).

be allocated to the cloud straightaway, even though they could have just waited in a queue for a short period of time. Similar to DBC, QoEUA does not employ a queuing system. Given those rationales, JSQ, Random, DBC, and QoEUA do not perform as good as SUAC or TPO even when the average service capacity is large.

5.4.3 Impact of Control Parameter V and Associated Weights ω_r , ω_d , and ω_h

In this section, we investigate the impact of the trade-off control parameter V as well as other associated weights, namely ω_r , ω_d , and ω_h . Since V , ω_r , ω_d , and ω_h do not influence the user allocation decisions made by JSQ, TPO, and Random, we do not include those approaches in this section. The effectiveness of SUAC against them has already been experimentally analyzed in Sections 5.4.1 and 5.4.2.

In Sets #3, #4, and #5, we simulate varying values of V under SUAC with different values of throughput benefit weight ω_r (Set #3), queuing delay cost weight ω_d (Set #4), and cloud latency cost weight ω_h (Set #5). In Figs. 8, 9, 10, 11, 12, and 13, the y-axis on the left-hand side corresponds to the value of the bar graph, and the y-axis on the right-hand side corresponds to the value of the line graph. As expected,

a higher value of V results in a longer average queue backlog (hence higher queuing delay cost) as can be seen in all experiment sets. This demonstrates the flexibility of SUAC that enables service providers to control the congestion state of their MEC systems.

The sensitivity of throughput benefit weight ω_r (Set #3). Fig. 8 plots the average queue backlog and time-average system benefit gained by SUAC with different values of ω_r . A higher ω_r means that the service provider places a higher priority on the benefit gained from system throughput. As a result, SUAC attempts to allocate more users to edge servers, leading to a higher queue backlog and more expensive queuing delay cost. Since more users are being allocated to edge servers, fewer users will be allocated to the remote cloud, lowering the cloud latency cost (Fig. 11). A greater ω_r gains a higher time-average system benefit under the same value of V because the higher throughput benefit and lower cloud latency cost outweigh the queuing delay cost.

In this experiment set, the time-average system benefit under the same ω_r remains virtually unchanged across varying values of V even when the queue backlog changes. The

reason is that the increasing throughput benefit, increasing queuing delay cost and decreasing cloud latency cost are balancing out each other. This will not be the case if the service provider adjusts ω_d or ω_h . Sets #4 and #5 will demonstrate how the changing patterns of time-average system benefit with changing V can be influenced differently compared to the pattern observed in this experiment set.

The sensitivity of queuing delay cost weight ω_d (Set #4). Fig. 9 plots the average queue backlog and time-average system benefit gained by SUAC with different values of ω_d . A higher value of ω_d means that a more congested queue is penalized harder than a less congested queue, and thus fewer users are allocated to edge servers, resulting in a lower queue backlog. This is why under the same V , a lower ω_d leads to a higher throughput benefit and a lower latency cost (Fig. 12). When ω_d is small, even though the queue backlog is large, the queuing delay cost is still low because of the small ω_d . Collectively, a lower ω_d gains a higher system benefit.

In terms of the time-average system benefit with V changing, there are several patterns here. With $\omega_d = 3$, the time-average system benefit tends to increase with the increasing V because the increasing throughput benefit and decreasing latency cost far outweigh the increasing queuing delay cost. With $\omega_d = 5$, they balance out each other so there is not much difference across different values of V . With $\omega_d = 7$ and 9, the increasing throughput benefit and decreasing latency cost start being outweighed by the increasing queuing delay cost, hence the decrease in time-average system benefit. Again, those patterns can be changed if the service provider adjusts the values of ω_h or ω_r .

The sensitivity of latency cost weight ω_h (Set #5). Fig. 10 plots the average queue backlog and time-average system benefit gained by SUAC with different values of ω_h . A higher value of ω_h means that the more users allocated to the remote cloud, the harder the penalty. Therefore, with the same V , SUAC will lean towards allocating more users to edge servers under a higher value of ω_h , leading to a higher average queue backlog, which also incurs a higher queuing delay cost. Since the cloud latency cost of a higher ω_h is much more expensive than that of a lower ω_h (Fig. 13), the system benefit gained by a higher ω_h is lower.

As V increases, the time-average system benefit gained by SUAC decreases for all experimental values of ω_h . This occurs because the increasing queuing delay cost outweighs the increasing throughput benefit and decreasing latency cost. Similar to Sets #3 and #4, the pattern can be adjusted with a different value of ω_r or ω_d .

As demonstrated, the control parameter V , and weight parameters ω_r , ω_d , and ω_h control the congestion state of a MEC system, which in turn influence the system benefit. Those parameters are selected by the service provider depending on their needs, or the significance of the throughput benefit, queuing delay cost, and latency cost. Given those pre-selected parameters, SUAC guides the user allocation process over time so that the system benefit is maximized.

6 RELATED WORK

This section has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

7 CONCLUSION AND FUTURE WORK

In this study, we investigate the stochastic edge user allocation problem in a time-slotted MEC system. A service provider needs to take into account several factors such as queuing delay and cloud latency costs while maximizing the system throughput. We address a realistic MEC environment where users come and go dynamically and randomly over time, making it hard to find an optimal allocation due to the lack of future information. We propose SUAC – a Lyapunov optimization-based online algorithm that allocates users without requiring any information about user arrivals and departures. As theoretically proven, SUAC achieves a bounded performance guarantee. We conduct a series of experiments based on a real-world dataset, which clearly demonstrates the superiority of SUAC over the existing approaches and its ability to achieve a desired tradeoff as well as strongly stabilize the system.

The future work has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

ACKNOWLEDGMENTS

This research is partly funded by Australian Research Council Discovery Project grants DP170101932, DP180100212, and DP200102491. John Grundy is supported by Laureate Fellowship FL190100035. Qiang He is the corresponding author of this article.

REFERENCES

- [1] D. Wang, Y. Peng, X. Ma, W. Ding, H. Jiang, F. Chen, and J. Liu, "Adaptive wireless video streaming based on edge computing: Opportunities and approaches," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 685–697, 2018.
- [2] Y. Liang, G. Jidong, S. Zhang, J. Wu, Z. Tang, and B. Luo, "A utility-based optimization framework for edge service entity caching," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 11, pp. 2384–2395, 2019.
- [3] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2018, pp. 207–215.
- [4] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.
- [5] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, H. Liu, Y. Qin, and P. Chen, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proceedings of IEEE International Conference on Web Services*. IEEE, 2019, pp. 91–98.
- [6] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2018, pp. 230–245.
- [7] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2019, pp. 86–101.

- [8] J. Ghaderi, "Randomized algorithms for scheduling vms in the cloud," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1-9.
- [9] K. Psychas and J. Ghaderi, "Randomized algorithms for scheduling multi-resource jobs in the cloud," *IEEE/ACM Transactions on Networking*, no. 99, pp. 1-14, 2018.
- [10] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759-2774, 2019.
- [11] 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*, vol. 26, no. 6, pp. 2651-2664, 2018.
- [12] A. Morgado, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "A survey of 5G technologies: Regulatory, standardization and industrial perspectives," *Digital Communications and Networks*, vol. 4, no. 2, pp. 87-97, 2018.
- [13] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang, and S. Ou, "Dynamic resource scheduling in mobile edge cloud with cloud radio access network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2429-2445, 2018.
- [14] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 1459-1467.
- [15] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1-211, 2010.
- [16] 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.
- [17] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky *et al.*, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proceedings of ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1-14.
- [18] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725-737, 2015.

The author biographies has been removed since the biographies have no close connection with the thesis. Please refer to the original paper online for a full, unedited version.

Cost-Effective User Allocation in NOMA-based Mobile Edge Computing Systems

We recognize that wireless communication is an inherent part of MEC as users are to be connected to base stations through wireless communication. Thus, the communication aspect must not be overlooked when allocating users to edge servers like the previous three research problems. Here, we attempt to solve the EUA problem while considering several key characteristics in wireless communication, including multi-channel, achievable data rate, and non-orthogonal multiple access (NOMA), etc. NOMA is a new multiple-access scheme in wireless communication that allows multiple users to share the same wireless channel. This is a promising enabler for the massive connectivity demanded by the forthcoming 5G/6G networks. An app vendor needs to allocate users to proper channels in edge servers and allocate transmit power to those users to minimize the system cost, which is measured by computing resource cost and transmit power cost, while ensuring the user data rate requirement. We tackle two scenarios where users are static (Section 5.1), and where users come and go randomly over time (Section 5.2).

5.1 A Static Scenario

In this section, we consider the scenario where users are static. We formulate this problem as an ordinal potential game and then find Nash equilibria by a decentralized game-theoretical approach, minimizing the system cost measured by the computing resource cost and transmit power cost. Its optimality and convergence rate are then theoretically analyzed. The proposed approach is also experimentally demonstrated to be highly effective and efficient, significantly outperforming two baseline and two state-of-the-art approaches.

This section is presented in the form of a published paper [33] as **P. Lai**, Q. He, G. Cui, F. Chen, J. Grundy, M. Abdelrazek, J. Hosking, and Y. Yang, "Cost-Effective User Allocation in 5G NOMA-based Mobile Edge Computing Systems," *IEEE Transactions on Mobile Computing*, 2021. doi: 10.1109/TMC.2021.3077470. ©2021 IEEE. Reprinted, with permission, from IEEE Transactions on Mobile Computing. *Note that several sections in the paper included below have been removed or slightly modified to reduce repeated content that has appeared elsewhere in this thesis. For a full, unedited version, please refer to the original paper itself.*

Cost-Effective User Allocation in 5G NOMA-based Mobile Edge Computing Systems

Phu Lai, Qiang He, *Senior Member, IEEE*, Guangming Cui, Feifei Chen, *Member, IEEE*, John Grundy, *Senior Member, IEEE*, Mohamed Abdelrazek, John Hosking, and Yun Yang, *Senior Member, IEEE*

Abstract—Mobile edge computing (MEC) allows edge servers to be placed at cellular base stations. App vendors like Uber and YouTube can rent computing resources and deploy latency-sensitive applications on edge servers for their users to access. Non-orthogonal multiple access (NOMA) is an emerging technique that facilitates the massive connectivity of 5G networks, further enhancing the capability of MEC. The edge user allocation (EUA) problem faces new challenges in 5G NOMA-based MEC systems. In this study, we investigate the EUA problem in a multi-cell multi-channel downlink power-domain NOMA-based MEC system. The main objective is to help mobile app vendors maximize their benefit by allocating maximum users to edge servers in a specific area at the lowest computing resource and transmit power costs. To this end, we introduce a decentralized game-theoretic approach to effectively select a channel and edge server for each user while fulfilling their resource and data rate requirements. We theoretically and experimentally evaluate our solution, which significantly outperforms various state-of-the-art and baseline approaches.

Index Terms—Non-orthogonal multiple access (NOMA), mobile edge computing, user allocation, game theory

1 INTRODUCTION

MOBILE edge computing (MEC) [1] is introduced to tackle one of the most challenging obstacles in cloud computing – high and unpredictable latency. By deploying edge servers at cellular base stations (BSs), mobile network operators can offer computing resources at the network edge, much closer to end-users. Mobile app vendors like Uber and YouTube can rent these computing resources to host their services and serve their users with low latency. This is of paramount importance for latency-sensitive services and applications such as interactive VR/AR gaming, vital monitoring systems, etc.

The rapid growth of mobile subscriptions promoted by the forthcoming 5G, which is predicted to reach 9 billion in 2025 [2], has put a great burden on the existing wireless communication infrastructure. Several multiple-access techniques for wireless communication have been widely adopted for decades, e.g., code division multiple access (CDMA), orthogonal frequency division multiple access (OFDMA), and time division multiple access (TDMA). In conventional systems that employ those orthogonal multiple access techniques, different users are allocated orthogonal resources in time, code, or frequency domain. Take OFDMA scheme, for example, each individual user is allocated a dedicated channel, which prevents multiple access interference. However, such schemes are not capable of supporting a massive number of simultaneous users. As

a result, non-orthogonal multiple access (NOMA) was proposed to facilitate the massive connectivity demanded by 5G [3], [4], [5]. NOMA achieves high spectrum efficiency by allowing multiple users to be served simultaneously using the same time and frequency resources (channels) in power or code domain [3]. It also lowers the transmission latency and signaling cost [6]. NOMA can be implemented flexibly to enhance the performance of many wireless technologies, namely MEC, multiple-input multiple-output (MIMO), visible light communications, millimeter-wave communications, cognitive and cooperative communications, massive MIMO, energy harvesting, and wireless caching [7]. MEC brings computing power closer to users with low latency. By integrating NOMA into MEC, network latency will be lowered even further. This is critical, especially when the numbers of 5G applications and MEC users are expected to grow rapidly in the near future.

The problem of allocating users to edge servers/base stations¹ in an MEC system is referred to as an *edge user allocation* (EUA) problem. Recently, researchers are starting to investigate the impact of NOMA on the computation offloading problem [8] in MEC systems but not the EUA problem. Existing user allocation approaches do not consider both communication and computation aspects in the MEC system at the same time. User allocation approaches in pure cellular systems [9], [10], [11] often lack the computation aspects of MEC, e.g., the existence of computing resources and their scarcity and heterogeneity. To lower the complexity of the problem, many of them even do not consider multi-cell and multi-channel [10], [11], [12], or limit the number of users on a channel [9], [13], [14], which holds back the potential of NOMA. Meanwhile, user allocation

- P. Lai, Q. He, G. Cui and Y. Yang are with the School of Software and Electrical Engineering, Swinburne University of Technology, 3122, Australia. E-mail: tlai, qhe, gcui, yyang@swin.edu.au.
- F. Chen and M. Abdelrazek are with the School of Information Technology, Deakin University, 3125, Australia. E-mail: feifei.chen, mohamed.abdelrazek@deakin.edu.au.
- J. Grundy is with the Faculty of Information Technology, Monash University, 3168, Australia. E-mail: john.grundy@monash.edu.
- J. Hosking is with the School of Science, University of Auckland, Auckland, New Zealand. E-mail: j.hosking@auckland.ac.nz.

¹ We speak interchangeably of edge servers and base stations. For the sake of consistency, we will hereafter try to use the term “edge server” instead of “base station”. In situations where the communication/networking aspects are discussed, “base station” will be used.

approaches in MEC systems [15], [16], [17], [18], [19], [20] often neglect or have not properly considered the communication aspects (multiple wireless channels, interference, and power control), especially in a NOMA setting. This is particularly uneconomical when MEC applications now have access to network information, e.g., QoS, throughput, neighbor cells, received signal, received power [21], [22]. These can be utilized by app vendors to optimize their services deployed on edge servers. In other words, some of the networking responsibilities such as power allocation can now be delegated to app vendors. In this study, we address the cost-effective EUA problem in a downlink multi-cell multi-channel multi-user 5G NOMA-based MEC system from an app vendor's perspective. An app vendor allocates each user by making a joint decision: 1) user allocation, which is the allocation of users to channels in edge servers, and 2) transmit power allocation, to maximize the number of allocated users and minimize the system cost (costs of computing resources and transmit power). Meanwhile, a number of unique constraints of MEC systems must be satisfied (minimum user data rate requirement, proximity, and resource constraints).

Applications hosted on edge servers often need to send a substantial amount of data to users, e.g., videos produced by content providers or rendered graphics produced by interactive VR/AR applications. In those applications, users only send light-weight instructions (job/task description) to edge servers. Thus, this study focuses on downlink transmissions in NOMA-based MEC systems. To deal with the *intra-cell interference* caused by multiple users sharing the same channel, successive interference cancellation (SIC), a multi-user signal separation technique, is applied at the receivers when decoding wireless signals. By multiplexing users in the power domain at the transmitters (BSs) and employing SIC at the receivers (users), NOMA has been shown to remarkably improve the capacity and user throughput performance compared with conventional multiple access schemes [3]. Together with *intra-cell interference*, *inter-cell interference* caused by nearby BSs also impacts the performance of a NOMA system. When the interference is severe, a user would require a lot of transmit power to achieve the minimum data rate. Therefore, interference must be taken into account when allocating users so that the app vendor can guarantee the quality of their service by maximizing the number of users that achieve a satisfactory data rate with minimum transmit power.

In addition to communication resources, computing resources rented on edge servers also need to be optimized. Similar to cloud computing, MEC also benefits from *multi-tenancy* [16], where multiple tenants/users can be simultaneously served by a single software instance or share the same infrastructure or database in an efficient manner [23], [24]. It allows higher resource utilization, energy efficiency, and overall performance on edge servers through workload consolidation [25]. In an MEC environment, multi-tenancy benefit can be achieved by allocating maximum users to an edge server provided that it does not overload the computing and communication resources in that edge server. Leveraging multi-tenancy effectively allows an app vendor to reduce the amount of computing resources required to serve its users, saving system costs or operating costs. In

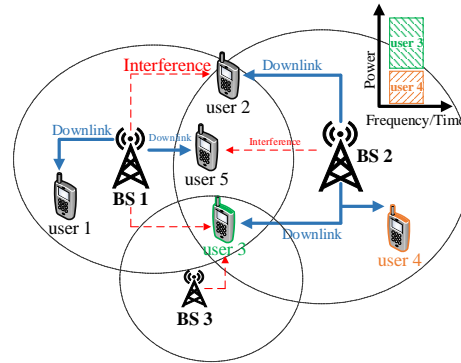


Fig. 1: An example of downlink multi-cell NOMA-based MEC networks

the meantime, saving edge servers' computing resources enables the app vendor to accommodate more users. This is essential for every app vendor and must be seriously considered in the allocation of users to edge servers.

The NOMA-EUA problem is NP-hard (proved later in Appendix B) so it is almost impossible to find an optimal solution in a real-world scenario. To mitigate this issue, we introduce an efficient game-theoretic approach to find sub-optimal solutions. The key contributions of this article are:

- We define and formulate the NOMA-EUA problem, taking into account multiple channels, interference, and power control.
- We model the NOMA-EUA problem as a potential game and propose a decentralized algorithm, named miUA, to efficiently find a Nash equilibrium. We then theoretically analyze the performance of miUA.
- We perform a series of experiments to empirically evaluate the performance of miUA. It is shown that miUA significantly outperforms all state-of-the-art and baseline approaches.

The rest of this article is organized as follows. Section 2 discusses the key motivation for this work. In Section 3, we review the relevant literature. Section 4 introduces our NOMA system model. Section 5 formulates the NOMA-EUA problem. Section 6 proposes a solution to the NOMA-EUA problem, including a theoretical analysis. The proposed solution is experimentally evaluated in Section 7. Finally, we draw a conclusion and point out future work in Section 8.

2 MOTIVATION

In a 5G MEC system, BSs are densely distributed, especially in high-traffic areas. Their cell coverage areas often partly overlap to minimize non-service areas. An app vendor can allocate a user positioned in an overlapping region to one of its neighbor edge servers that has sufficient communication and computing resources. Multiple users can be concurrently served on the same wireless channel. A channel cannot serve too many users at once due to the

high interference it would cause. Furthermore, compared to a typical cloud environment, resources on edge servers in an MEC system are highly constrained and expensive [26], [27]. An ineffective user allocation will soon exhaust the computing and networking resources and result in a poor data rate for users. Similar to [16], [28], [29], [30], [31], we study a quasi-static scenario, in which users stay fairly still in one place and do not quickly travel across different BSs/edge servers. Typical scenarios include those that involve mobile or IoT users not moving quickly, traffic cameras, smart sensors, etc.

In downlink NOMA, SIC is facilitated by differentiating the transmit power between users sharing the same channel [4]. In single-cell NOMA scenarios, to ensure successful decoding of the superposed signal sent by a BS, stronger users on a channel (who have higher channel gains) are allocated less transmit power, and weaker users (who have lower channel gains) are allocated more transmit power [32]. Each user employs SIC to cancel the signal interference caused by weaker users. Nevertheless, decoding solely by the order of channel gains does not apply to multi-cell NOMA scenarios because users' channel conditions are now also affected by the *inter-cell interference* caused by their unassociated neighbor BSs [10], which could be very extreme in a dense multi-cell network.

Take Fig. 1 for example, where user 3 and user 4 suffer from intra-cell interference as they share the same channel in the same edge server. In addition to intra-cell interference, user 3 is also impacted by the inter-cell interference caused by its neighbor base stations, i.e., BS 1 and BS 2. Fortunately, it has been demonstrated that an effective power allocation and decoding order can considerably reduce inter-cell interference [10], which in turn improves users' data rate, or system throughput in general [10], [32]. Thus, the EUA problem must be jointly solved with the power allocation problem to ensure that every user receives an app-specific satisfactory data rate.

3 RELATED WORK

This section has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

4 SYSTEM MODEL

4.1 System Description

Edge servers: Let $\mathcal{S} = \{s_1, s_2, \dots, s_M\}$ denote the set of M BSs or edge servers in an MEC system. The computing capacity of an edge server $s_j \in \mathcal{S}$ is represented by a $|\mathcal{T}|$ -dimensional vector $Q_j = (Q_j^t)$, where each dimension Q_j^t is the capacity of resource type $t \in \mathcal{T} = \{\text{CPU, RAM, storage, } \dots\}$. Each edge server s_j covers a cell of radius R_j .

For each edge server s_j , the total bandwidth B is equally divided between a set of V channels $\mathcal{C}_j = \{c_j^1, c_j^2, \dots, c_j^V\}$. The bandwidth of each channel $c_j^k \in \mathcal{C}_j$ is thus $B_j^k = B/V$, where $k \in \{1, 2, \dots, V\}$.

Mobile users: We use $\mathcal{U} = \{u_1, u_2, \dots, u_N\}$ to represent the set of N users. Let a $|\mathcal{T}|$ -dimensional vector $w_i = (w_i^t)$, $t \in \mathcal{T}$, denote user u_i 's computing resource requirement, i.e., the amount of computing resources that could be consumed

by an edge server assigned to serve user $u_i \in \mathcal{U}$. Let $d_{j,i}$ be the distance between edge server s_j and user u_i . The set of user u_i 's neighbor edge servers is denoted by $\mathcal{S}_i = \{s_j \in \mathcal{S} | d_{j,i} \leq R_j\}$. Note that a user can only be allocated to a single channel in an edge server. To allocate each user $u_i \in \mathcal{U}$, two decisions need to be made as defined below:

Definition 1. (User Allocation Decision) Let $\mathbf{a}_{j,i}^k = \{0, 1\}$ be the binary decision variable for user u_i . We have $\mathbf{a}_{j,i}^k = 1$ if user u_i is allocated to channel c_j^k in edge server s_j ; otherwise $\mathbf{a}_{j,i}^k = 0$. We use $\mathbf{a} = \{\mathbf{a}_i | u_i \in \mathcal{U}\}$ to denote the user allocation strategy composed by the decisions for all the users $\forall u_i \in \mathcal{U}$. We have $\mathbf{a}_i \triangleq (s_j, c_j^k)$, where $\mathbf{a}_{j,i}^k = 1$, indicate the pair of channel and edge server that is assigned to serve user u_i .

Let $\mathcal{U}_j = \{u_i \in \mathcal{U} | \sum_{k=1}^V \mathbf{a}_{j,i}^k = 1\}$, $\forall s_j \in \mathcal{S}$, denote the set of users allocated to edge server s_j , and $\mathcal{U}_j^k = \{u_i \in \mathcal{U} | \mathbf{a}_{j,i}^k = 1\}$, $\forall s_j \in \mathcal{S}, \forall c_j^k \in \mathcal{C}_j$, denote the set of users allocated to channel c_j^k on edge server s_j .

Definition 2. (Power Allocation Decision) Let $\mathbf{p}_{j,i}^k$ denote the transmit power allocated to user u_i on channel c_j^k of edge server s_j , i.e., the amount of power consumed by edge server s_j to transmit data to user u_i on channel c_j^k . We use $\mathbf{p} = \{\mathbf{p}_i | u_i \in \mathcal{U}\}$ to denote the power allocation strategy composed by the power allocation decisions for all the users, and $\mathbf{p}_i \triangleq (\mathbf{p}_{j,i}^k)$, $\forall u_i \in \mathcal{U}, \forall s_j \in \mathcal{S}, \forall c_j^k \in \mathcal{C}_j$.

Appendix A of the supplementary file summarizes the notations used in this article.

4.2 Signal Model

According to the NOMA scheme [3], edge server s_j broadcasts a superposition-coded signal x_j^k to all users allocated on channel c_j^k simultaneously. The transmitted signal x_j^k can be expressed as follows:

$$x_j^k = \sum_{u_i \in \mathcal{U}_j^k} \sqrt{\mathbf{p}_{j,i}^k} x_{j,i}^k \quad (1)$$

where $x_{j,i}^k$ is the signal transmitted from edge server s_j to user u_i on channel c_j^k . NOMA facilitates a simultaneous transmission of multiple users' signals [10], [33], whose power levels are differentiated, over the same transmission period and channel. We denote the total transmit power of edge server s_j on channel c_j^k by $p_j^k = \sum_{u_i \in \mathcal{U}_j^k} \mathbf{p}_{j,i}^k$. The total transmit power allocated to all users on all channels of an edge server s_j must not exceed its maximum transmit power P_j : we have $\sum_{c_j^k \in \mathcal{C}_j} p_j^k \leq P_j$.

For each user u_i allocated to edge server s_i on channel c_j^k (when $\mathbf{a}_{j,i}^k = 1$), its received signal $y_{j,i}^k$ is the summation of its intended signal, intra-cell interference (caused by other users sharing the same channel), inter-cell interference (caused by nearby BSs/edge servers), and other noise. Note that (1) includes both the signal intended for user u_i and the signal intended for the other users sharing the same channel with u_i , which causes intra-cell interference. $y_{j,i}^k$ is defined as follows:

$$y_{j,i}^k = \underbrace{h_{j,i}^k x_j^k}_{\text{intended signal}} + \underbrace{\sum_{s_l \in \mathcal{S} \setminus \{s_j\}} h_{l,i}^k x_l^k}_{\text{inter-cell interference}} + \underbrace{o_{j,i}^k}_{\text{noise}} \quad (2)$$

where $h_{j,i}^k$ is the complex channel coefficient between user u_i and edge server s_j on channel c_j^k , and $o_{j,i}^k$ is the additive white Gaussian noise with variance σ^2 , i.e., $o_{j,i}^k \sim \mathcal{CN}(0, \sigma^2)$. User u_i 's channel gain on channel c_j^k is $|h_{j,i}^k|^2$, which includes all the factors that can influence a signal.

4.3 Successive Interference Cancellation

This overview section provides an established foundation of Successive Interference Cancellation (SIC) in a downlink NOMA system. We do not consider uplink transmissions because in the scenario being studied, users only send light-weight job instructions to edge servers, which will return a substantial amount of data back to the users. SIC is implemented for users sharing the same channel, i.e. \mathcal{U}_j^k , so that they can decode their received superposed signal. Assume that \mathcal{U}_j^k has been determined, i.e., some users have been allocated to the k -th channel in edge server s_j . With SIC, stronger users detect and cancel the signals of weaker users, who treat the signals of stronger users as noise [3]. Without loss of generality, suppose all the users in \mathcal{U}_j^k are ordered by their channel conditions: $u_1, u_2, \dots, u_{|\mathcal{U}_j^k|}$, where u_1 and $u_{|\mathcal{U}_j^k|}$ has the weakest and strongest channel conditions, respectively. SIC is not required for user u_1 since it is the first in \mathcal{U}_j^k to decode signal. User u_1 first decodes $x_{1,j}^k$ and subtracts its components from $y_{1,j}^k$. The user who comes next in the decoding order (user u_2) can thus decode its received signal without interference from user u_1 . Following this principle, the signal-to-interference-plus-noise ratio (SINR) of the signal received by user $u_i \in \mathcal{U}_j^k$ is:

$$\gamma_{j,i}^k = \frac{|h_{j,i}^k|^2 \mathbf{P}_{j,i}^k}{|h_{j,i}^k|^2 \sum_{q=i+1}^{|\mathcal{U}_j^k|} \mathbf{P}_{j,q}^k + I_{j,i}^k + \sigma^2} \quad (3)$$

where $I_{j,i}^k = \sum_{s_i \in \mathcal{S}_i \setminus \{s_j\}} |h_{j,i}^k|^2 p_i^k$ is the inter-cell interference caused by user u_i 's neighbor edge servers on channel c_j^k . Given (3), the SINR of user $u_{|\mathcal{U}_j^k|}$, which is the last user to decode the received signal, is: $\gamma_{j,|\mathcal{U}_j^k|}^k = (|h_{j,|\mathcal{U}_j^k|}^k|^2 \mathbf{P}_{j,|\mathcal{U}_j^k|}^k) / (I_{j,|\mathcal{U}_j^k|}^k + \sigma^2)$.

Suppose $u_i, u_q \in \mathcal{U}_j^k$ and $i < q$, i.e., user u_q has a stronger channel condition. According to [10], [34], [35], in order to ensure a successful SIC, user u_q 's achievable data rate for decoding user u_i 's signal must be greater than or equal to user u_i 's data rate for decoding its own signal: $r_{j,q \rightarrow i}^k \geq r_{j,i \rightarrow i}^k$. If this condition is not satisfied, user u_i 's achievable data rate will decrease due to the intra-cell interference not being canceled. Thus, user u_i 's achievable data rate $r_{j,i}^k$ on channel c_j^k can be given by:

$$r_{j,i}^k = \min\{r_{j,q \rightarrow i}^k | \forall q \geq i\} \quad (4)$$

where $r_{j,q \rightarrow i}^k$, i.e. user u_q 's data rate for decoding user u_i 's signal is:

$$r_{j,q \rightarrow i}^k = B_j^k \log_2 \left(1 + \frac{|h_{j,q}^k|^2 \mathbf{P}_{j,i}^k}{|h_{j,q}^k|^2 \sum_{t=i+1}^{|\mathcal{U}_j^k|} \mathbf{P}_{j,t}^k + I_{j,q}^k + \sigma^2} \right) \quad (5)$$

Intuitively, user u_i 's achievable data rate is the minimum data rate of the users that come after user u_i in the SIC decoding order, which will be discussed next.

SIC Decoding Order. As analyzed above, the position of a user in the decoding order plays an important role in its

achievable data rate. Therefore, the decoding order cannot be overlooked when the data rate is being optimized. By transforming (4), user u_i 's achievable data rate $r_{j,i}^k$ can be expressed by:

$$r_{j,i}^k = B_j^k \log_2 \left(1 + \frac{\mathbf{P}_{j,i}^k}{\sum_{t=i+1}^{|\mathcal{U}_j^k|} \mathbf{P}_{j,t}^k + H_{j,i}^k} \right) \quad (6)$$

where

$$H_{j,i}^k = \max \left\{ \frac{I_{j,q}^k + \sigma^2}{|h_{j,q}^k|^2} | \forall q \geq i \right\} \quad (7)$$

To ensure an acceptable data rate with low transmit power for all the users, the decoding order should be determined based on their channel conditions and inter-cell interference as follows: $H_{j,1}^k \geq \dots \geq H_{j,|\mathcal{U}_j^k|}^k$. This order is guaranteed if the decoding order of users allocated to channel c_j^k on edge server s_j follows the sequence:

$$\Theta(c_j^k) \triangleq \frac{I_{j,1}^k + \sigma^2}{|h_{j,1}^k|^2} \geq \dots \geq \frac{I_{j,|\mathcal{U}_j^k|}^k + \sigma^2}{|h_{j,|\mathcal{U}_j^k|}^k|^2} \quad (8)$$

It has been shown that this decoding order is an optimal order for efficiently increasing the data rate of each individual users [10]. If this decoding order is satisfied, user u_i 's achievable data rate $r_{j,i}^k$ is:

$$r_{j,i}^k = B_j^k \log_2 \left(1 + \frac{|h_{j,i}^k|^2 \mathbf{P}_{j,i}^k}{|h_{j,i}^k|^2 \sum_{t=i+1}^{|\mathcal{U}_j^k|} \mathbf{P}_{j,t}^k + I_{j,i}^k + \sigma^2} \right) \quad (9)$$

4.4 Resource Utilization Model

Multi-tenancy is an important feature in computing resource management [24] and must also be considered in EUA [16]. By allowing users to share the same software instance, app vendors can efficiently utilize the computing resources rented on edge servers. This is critical in MEC systems where computing resources on edge servers are relatively scarce [31]. This drives app vendors to aggregate their users to a small set of edge servers. For example, say two users need to be served who require one CPU each. Allocating them to two different edge servers would require two CPUs to serve them. Taking advantage of multi-tenancy, allocating them to the same edge server to be served by the same software instance would require slightly less than two CPUs. According to [36], the CPU utilization of edge server s_j with multi-tenancy can be estimated based on the number of users served by s_j :

$$f_j = -\log_z(|\mathcal{U}_j|) \quad (10)$$

where z is decided based on app-specific computation task size ($0.9 < z < 1$) and $|\mathcal{U}_j| > 1$ is the number of users allocated to edge server s_j . When $|\mathcal{U}_j|$ increases, the CPU utilization of edge server s_j increases monotonically until it converges at some point. The convergence occurs when $|\mathcal{U}_j|$ is sufficiently high and incurs an expensive computational overhead for resource sharing [25]. When the extra computational overhead exceeds the corresponding multi-tenancy benefit, it does not benefit the app vendor as much as before, and it is more cost-effective to serve the extra users with another edge server.

As shown in [25], the storage utilization also follows a model similar to (10). Assuming the utilization of other computing resources, e.g., RAM or storage, also follows a similar model, the utilization of the computing resource $t \in \mathcal{T}$ on edge server s_j when user u_i is being allocated can be measured by:

$$f_{j,i}^t = -\log_{z_t^t}(|\mathcal{U}_j|) \quad (11)$$

where z_t^t is decided based on the computation task size of user u_i and is dependent of computing resource type $t \in \mathcal{T}$, $|\mathcal{U}_j|$ is the number of users on server s_j to which user u_i is allocated. We have $0 < f_{j,i}^t < 1, \forall t \in \mathcal{T}, \forall s_j \in \mathcal{S}$.

4.5 Computing Resource Cost Model

In an MEC system, an app vendor needs to pay for computing resources rented on edge servers. Thus, it is important to utilize multi-tenancy to the fullest extent to save on computing resource costs. Given a user allocation strategy \mathbf{a} , the computing resource cost incurred by the decision of user $u_i \in \mathcal{U}$ is:

$$M_{\mathbf{a}}(u_i) = \begin{cases} \sum_{t \in \mathcal{T}} \tau^t (1 - f_{j,i}^t) w_i^t, & \text{if } \sum_{c_j^k \in \mathcal{C}_j} \mathbf{a}_{j,i}^k = 1 \\ \epsilon \sum_{t \in \mathcal{T}} \tau^t w_{max}^t, & \text{if } \sum_{s_j \in \mathcal{S}} \sum_{c_j^k \in \mathcal{C}_j} \mathbf{a}_{j,i}^k = 0 \end{cases} \quad (12)$$

where $(1 - f_{j,i}^t) w_i^t$ is the amount of computing resource type $t \in \mathcal{T}$ in edge server s_j that the app vendor would need to reserve for user u_i if user u_i is allocated to edge server s_j , and τ^t is the weight controlling the app vendor's priority for saving computing resource type $t \in \mathcal{T}$ by leveraging multi-tenancy. For example, if an app is compute-intensive, saving processing power such as CPU would be more beneficial than saving other computing resources such as storage. When $\sum_{s_j \in \mathcal{S}} \sum_{c_j^k \in \mathcal{C}_j} \mathbf{a}_{j,i}^k = 0$, i.e., user u_i is not allocated to any edge server, the cost incurred is modeled as $\epsilon \sum_{t \in \mathcal{T}} \tau^t w_{max}^t$, where $\epsilon > 1$ is the weight that indicates the severity of the penalty when the user is unallocated, and w_{max}^t is the maximum amount of computing resource of type $t \in \mathcal{T}$ that a user in the system may consume. From an app vendor's perspective, it is essential to minimize the number of unallocated users. To drive the app vendor to allocate as many users as possible to edge servers, the cost incurred by failing to allocate a user is modelled to be always greater than the cost incurred when the user is allocated to an edge server. Otherwise, it will simply choose not to allocate any users at all to bring the system cost down to zero.

5 PROBLEM FORMULATION

We model the NOMA-EUA problem as a mixed-integer constrained optimization problem as follows:

$$\min_{\{\mathbf{a}, \mathbf{p}\}} \sum_{i=1}^N \left(\eta_1 M_{\mathbf{a}}(u_i) + \eta_2 \sum_{j=1}^M \sum_{k=1}^V \mathbf{a}_{j,i}^k \mathbf{p}_{j,i}^k \right) \quad (13a)$$

$$\text{s.t. } \sum_{i=1}^N \sum_{t=1}^{|\mathcal{T}|} \mathbf{a}_{j,i}^t (1 - f_{j,i}^t) w_i^t \leq Q_j^t, \forall s_j \in \mathcal{S} \quad (13b)$$

$$\sum_{j=1}^M \sum_{k=1}^V \mathbf{a}_{j,i}^k d_{j,i} \leq R_j, \forall u_i \in \mathcal{U} \quad (13c)$$

$$\sum_{j=1}^M \sum_{k=1}^V \mathbf{a}_{j,i}^k \leq 1, \forall u_i \in \mathcal{U} \quad (13d)$$

$$\mathbf{a}_{j,i}^k r_{j,i}^k \geq \mathbf{a}_{j,i}^k r_{min}, \forall s_j \in \mathcal{S}, \forall c_j^k \in \mathcal{C}_j, \forall u_i \in \mathcal{U} \quad (13e)$$

$$\Theta(c_j^k), \forall s_j \in \mathcal{S} \quad (13f)$$

$$\sum_{k=1}^V \sum_{i=1}^N \mathbf{a}_{j,i}^k \mathbf{p}_{j,i}^k \leq P_j, \forall s_j \in \mathcal{S} \quad (13g)$$

$$\mathbf{a}_{j,i}^k \in \{0, 1\}, \forall s_j \in \mathcal{S}, \forall c_j^k \in \mathcal{C}_j, \forall u_i \in \mathcal{U} \quad (13h)$$

$$\mathbf{p}_{j,i}^k \in \mathbb{R}_{\geq 0}, \forall s_j \in \mathcal{S}, \forall c_j^k \in \mathcal{C}_j, \forall u_i \in \mathcal{U} \quad (13i)$$

where \mathbf{a} and \mathbf{p} are the user and power allocation strategies, respectively. Optimization objective (13a) minimizes the total system cost, i.e., the computing resource cost modeled in Section 4.5 and the cost of the total transmit power allocated to all users. η_1 and η_2 ($\eta_1 + \eta_2 = 1$) are the weight parameters that indicates the relative importance of the normalized computing resource cost and normalized transmit power cost, respectively. Computing resource constraint (13b) ensures that the aggregated computing resource consumption of all the users in an edge server does not exceed its computing capacity. Proximity constraint (13c) ensures that an edge server can only serve users within its coverage area. Constraint (13d) indicates that any user can only be either unallocated, or be allocated to one channel of an edge server. Constraint (13e) ensures a minimum app-specific data rate r_{min} for each allocated user. In a dynamic scenario where multiple time slots are considered, this constraint could be relaxed as long as the time-average data rate satisfies the requirement. For example, the data rate of a user may be lower than r_{min} in a time slot and higher than r_{min} in the next time slot. This enables a more flexible allocation of power. Constraint (13f) enforces the optimal decoding order stated in Section 4.3, which allows any user to successfully decode the signals of weaker users on the same channel. Constraint (13g) ensures that all the users allocated to an edge server do not use more transmit power than the edge server's maximum power capacity. Constraints (13h) and (13i) indicate the possible values of user allocation decisions $\mathbf{a}_{j,i}^k$ and transmit power decisions $\mathbf{p}_{j,i}^k$.

The optimization problem above can be proved to be NP-hard by showing that its subproblem (Section 5.1) is NP-hard. Considering the dynamic channel conditions associated with different edge servers, the NOMA-EUA problem becomes even more complicated. To solve it efficiently, we decompose it into two subproblems: 1) a user allocation problem, and 2) a power allocation problem.

The user allocation problem (Section 5.1) will be solved first to allocate users to channels in edge servers. At this stage, we aim to fully utilize the computing resources on edge servers by minimizing the computing resource cost. In the meantime, we attempt to minimize user interference to save on transmit power. Since transmit power has not been allocated to users yet, a fixed and identical transmit power is temporarily assumed for all the users so that we can approximate the intra-cell and inter-cell interference experienced by the users. This method will increase the likelihood of low interference in general. When all the users have been allocated to edge servers, transmit power will be allocated to them (Section 5.2) to fulfill their minimum data rate requirement with minimum transmit power.

5.1 User Allocation Problem

The user allocation problem can be modelled as follows:

$$\begin{aligned} \min_{\{\mathbf{a}\}} C_{\mathbf{a},\mathbf{p}} &\triangleq \sum_{i=1}^N (\eta_1 M_{\mathbf{a}}(u_i) + \eta_2 I_{\mathbf{a},\mathbf{p}}(u_i)) \\ \text{s.t.} & \quad (13\text{b}), (13\text{c}), (13\text{d}), (13\text{h}) \end{aligned} \quad (14)$$

where in optimization objective (14), we add a new term $I_{\mathbf{a},\mathbf{p}}(u_i)$, i.e., the interference cost, to replace the transmit power cost shown in (13a). As transmit power is determined by the channel condition, we can minimize the transmit power by proactively selecting a pair of edge server and channel that is most likely to incur the lowest interference to all the users involved. Note that transmit power and interference are inter-dependent. The interference experienced by a user is influenced by its channel gain and the transmit power allocated to other users; and the transmit power of a user is determined based on the interference it experiences. To deal with this, we fix one parameter while optimizing the other. Specifically, at this stage, all the users are assumed to have identical and fixed transmit power. Given a fixed transmit power, one can pick an edge server and a channel so that later on, when a proper power allocation mechanism kicks in, it will incur low interference for all the users.

When η_1 is higher than η_2 , our proposed user allocation approach, miUA, would prioritize saving computing resource costs. When η_2 is higher than η_1 , it would prioritize minimizing interference and thus saving the transmit power. Weights η_1 and η_2 can be adjusted domain-specifically to fulfil the app vendor's needs. For example, if the transmit power cost in a specific region is too high, η_1 can be lowered. Given a user allocation strategy \mathbf{a} and a power allocation strategy \mathbf{p} , which is fixed for now, the interference-plus-noise experienced by a user u_i , denoted by $I_{\mathbf{a},\mathbf{p}}(u_i)$, is defined as:

$$I_{\mathbf{a},\mathbf{p}}(u_i) = \begin{cases} |h_{j,i}^k|^2 \sum_{q=i+1}^{\lfloor \mathcal{U} \rfloor} \mathbf{p}_{j,q}^k + I_{j,i}^k + \sigma^2, & \text{if } \sum_{c_j^k \in \mathcal{C}_j} \mathbf{a}_{j,i}^k = 1 \\ \varepsilon I_{max}, & \text{if } \sum_{s_j \in \mathcal{S}} \sum_{c_j^k \in \mathcal{C}_j} \mathbf{a}_{j,i}^k = 0 \end{cases} \quad (15)$$

where $\varepsilon > 1$ is the weight specified by the app vendor that indicates the severity of the penalty when the user is unallocated, I_{max} is the maximum interference-plus-noise that a user could experience. It is formulated in this way so that the interference cost of unallocated users is always greater than

the interference cost of allocated users, thus driving app vendors to allocate users to edge servers. The computing resource cost has also been formulated by following this methodology in Section 4.5. We can prove the NP-hardness of this problem by reducing the NP-complete PARTITION problem [37] to a special case of its corresponding decision version. The proof can be found in Appendix B.

Note that in the implementation of our proposed algorithm, $M_{\mathbf{a}}(u_i)$ and $I_{\mathbf{a},\mathbf{p}}(u_i)$ will be min-max normalized. The possible minimum and maximum values of computing resource and interference costs can easily be calculated based on the given edge server information in real-world scenarios, i.e. edge server computing resource capacity, available channels, edge server locations, and minimum user data rate requirement. Constraints related to power and data rate, including (13e), (13f), (13g), and (13i) are not considered in this subproblem because they do not contribute to the optimization of computing resources. These constraints will be enforced through power allocation.

5.2 Power Allocation Problem

Once users have been allocated to channels in edge servers by solving the user allocation problem formulated above, we start allocating transmit power to users. The power allocation problem is expressed as follows:

$$\begin{aligned} \min_{\{\mathbf{p}\}} & \sum_{i=1}^N \sum_{j=1}^M \sum_{k=1}^V \mathbf{a}_{j,i}^k \mathbf{p}_{j,i}^k \\ \text{s.t.} & \quad (13\text{e}), (13\text{f}), (13\text{g}), (13\text{i}) \end{aligned} \quad (16)$$

The main objective of this subproblem is to allocate to users as little transmit power as possible while satisfying the user's minimum data rate requirement, SIC decoding order constraint, and power capacity constraint. A data rate higher than what is required for accessing an app is not necessary for most, if not all, apps.

6 USER ALLOCATION

In this section, we present a game-theoretic approach employed by miUA to effectively and efficiently solve the user allocation problem introduced in Section 5.1. The power allocation problem introduced in Section 5.2 will be solved in Section H. Over the years, game theory has been shown to be a versatile method for solving NP-hard problems in MEC systems [16], [29], [31]. In this study, players are simulated to make allocation decisions individually, pursuing to achieve objective (14). The game is decentralized by design and can alleviate the computational overhead that occurs by a centralized optimal solution.

6.1 Game Formulation and Properties

Our game aims to determine a user allocation strategy \mathbf{a} , which consists of the allocation decisions for all the users. Those decisions are made to pursue the app vendor's objective (14) by obeying the rules of the game. Let $\mathbf{a}_{-i} = (\mathbf{a}_1, \dots, \mathbf{a}_{i-1}, \mathbf{a}_{i+1}, \dots, \mathbf{a}_N)$ denote the user allocation strategy except the decision for user u_i . Given other users' decisions \mathbf{a}_{-i} , each individual user u_i will try to find a decision so that the total system cost is minimized.

The user allocation problem is modeled as a game $\Gamma = (\mathcal{U}, \{\mathcal{A}_i\}_{u_i \in \mathcal{U}}, \{C_{\mathbf{a},\mathbf{p}}(\mathbf{a}_i)\}_{u_i \in \mathcal{U}})$, where \mathcal{U} is the set of users

(players), \mathcal{A}_i is the set of all possible allocation strategies for each user u_i , and $C_{a,p}(\mathbf{a}_i)$ is the cost function that measures the cost incurred by user u_i 's decision $\mathbf{a}_i = (s_j, c_j^k)$, the lower the better, $C_{a,p}(\mathbf{a}_i) = \sum_{u_q \in \mathcal{U}_i \cup \mathcal{U}_j} (\eta_1 M_a(u_q) + \eta_2 I_{a,p}(u_q))$, where \mathcal{U}_i is the set of users allocated to server s_i , which is the server to which user u_i was allocated (if any) before it is allocated to server s_j . Both \mathcal{U}_i and \mathcal{U}_j must be considered because switching a user from a server s_i to another server s_j impacts the computing resources and inter-cell/inter-cell interference of the users in server s_i and s_j .

Next, we show that there exists at least one Nash equilibrium in the game, which is a stable state of the game where the system cost cannot be further lowered by changing the decision for any individual users unilaterally.

Definition 3. (Nash Equilibrium) A user allocation strategy $\mathbf{a}^* = (\mathbf{a}_1^*, \dots, \mathbf{a}_N^*)$ is a Nash equilibrium when no user can unilaterally change its decision to further lower the system cost:

$$C_{a_{-i},p}(\mathbf{a}_i^*) \leq C_{a_{-i},p}(\mathbf{a}_i), \forall \mathbf{a}_i \in \mathcal{A}_i, \forall u_i \in \mathcal{U} \quad (17)$$

Lemma 1 guarantees that if there exists a Nash equilibrium, the decisions for all the users will finally constitute an allocation strategy that is a Nash equilibrium through finite iterations in the game.

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

Proof: Please refer to Appendix C. \square

A potential game always admits at least one Nash equilibrium [38]. To prove the existence of a Nash equilibrium in our user allocation game, we need to show that it is a potential game, which is defined as follows.

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

$$C_{a_{-i},p}(\mathbf{a}_i) > C_{a_{-i},p}(\mathbf{a}'_i) \Leftrightarrow \phi_{a_{-i}}(\mathbf{a}_i) > \phi_{a_{-i}}(\mathbf{a}'_i) \quad (18)$$

where $u_i \in \mathcal{U}$, $\mathbf{a}_i, \mathbf{a}'_i \in \mathcal{A}_i$ and $\mathbf{a}_{-i} \in \prod_{q \neq i} \mathcal{A}_q$.

The following theorem proves that our user allocation game is an ordinal potential game.

Theorem 1. The formulated user allocation game Γ is an ordinal potential game with the potential function:

$$\begin{aligned} \phi(\mathbf{a}) = & \sum_{u_i \in \mathcal{U}} \eta_1 \sum_{t \in \mathcal{T}} (\tau^t w_i^t \log_{z_i^t}(|\mathcal{U}_j|)) \mathbb{1}_{\sum_{c_j^k \in \mathcal{C}_j} a_{j,i}^k = 1} \\ & + \sum_{u_i \in \mathcal{U}} \eta_2 h_{j,i}^k \sum_{q=i+1}^{|U_j^k|} p_{j,i}^k \mathbb{1}_{a_q = a_i} \mathbb{1}_{\sum_{c_j^k \in \mathcal{C}_j} a_{j,i}^k = 1} \\ & + \sum_{u_i \in \mathcal{U}} \eta_2 \sum_{s_i \in \mathcal{S} \setminus \{s_j\}} (|h_{i,i}^k|^2 p_i^k) \mathbb{1}_{\sum_{c_j^k \in \mathcal{C}_j} a_{j,i}^k = 1} \\ & + \sum_{u_i \in \mathcal{U}} (\epsilon \sum_{t \in \mathcal{T}} (\tau^t w_{max}^t) + \epsilon I_{max}) \mathbb{1}_{\sum_{s_j \in \mathcal{S}} \sum_{c_j^k \in \mathcal{C}_j} a_{j,i}^k = 0} \end{aligned} \quad (19)$$

where $\mathbb{1}_{condition}$ is an indicator function that returns 1 if the condition is true, and 0 otherwise.

Proof: Please refer to Appendix D. \square

6.2 Decentralized User Allocation Algorithm

To solve the user allocation game formulated above, we propose a multi-tenancy and interference-aware user allocation algorithm (miUA). It is an iterative and decentralized algorithm that follows a class of strategy updating rules called *best response dynamics* [39], which is an evolutionary process involving a finite number of iterations. In each iteration, the decision for each user is determined by its best responses (the allocation decisions that incur the lowest system costs) to the decisions for other users made in the previous iteration. This is a decentralized process where edge servers run the algorithm in parallel and coordinate the iterations of the game via messaging synchronization, similar to [16], [29], [31]. Due to the *Finite Improvement Property* of ordinal potential games, it is guaranteed that this process will eventually converge to a Nash equilibrium [38].

Algorithm 1 miUA

Input: \mathcal{S}, \mathcal{U} , and other parameters
Output: user allocation strategy \mathbf{a}

- 1: **initialization:**
- 2: every user u_i is initially unallocated, $\mathbf{a}_i = (s_j, c_j^k) = (0, 0), \forall u_i \in \mathcal{U}$.
- 3: **end initialization**
- 4: **repeat**
- 5: Get the current system cost $C_{a,p}$
- 6: **for** each user $u_i \in \mathcal{U}$ **do**
- 7: **for** each neighbor server $s_j \in \mathcal{S}_i$ of user u_i **do**
- 8: **for** each channel $c_j^k \in \mathcal{C}_j$ on server s_j **do**
- 9: Calculate $C_{a',p}(\mathbf{a}'_i)$ – the new cost if user u_i is allocated to channel c_j^k in server s_j
- 10: **end for**
- 11: **end for**
- 12: From all possible decisions \mathbf{a}'_i above, find one with the lowest cost $C_{a',p}(\mathbf{a}'_i)$
- 13: **if** $\mathbf{a}'_i \neq \mathbf{a}_i$ and $C_{a',p} < C_{a,p}$ **then**
- 14: Contend \mathbf{a}'_i for the decision update
- 15: **end if**
- 16: **end for**
- 17: Find user u_i , whose decision update \mathbf{a}'_i incurs system cost $C_{a',p}$ that is the lowest among all users \mathcal{U}
- 18: Apply decision \mathbf{a}'_i
- 19: **until** no decision updates needed for any users
- 20: Allocate transmit power to users

Given all the required parameters, Algorithm 1 allocates users so that the total system cost (14) is minimized. During the allocation, every user is assumed to have a fixed and identical transmit power p initially. Once all the users have been allocated, each user's transmit power will be adjusted to meet the data rate requirement. This will be discussed in Section H. With regards to the user allocation, no user is allocated initially (Lines 1-3). After that, decisions are updated for users iteratively (Lines 5-16) such that the system cost incurred in the next iteration is lower than the previous iteration. The updated decision for user u_i is denoted as \mathbf{a}'_i , which incurs a new system cost $C_{a',p}$. Once all the decision updates are submitted, the decision that incurs the lowest system cost (Line 17) will be chosen and applied to the corresponding user (Line 18). The allocation strategy

\mathbf{a} will also be updated accordingly. Note that the allocation strategy \mathbf{a} at this stage is not final and can be updated in the consequent iterations. After this iteration, users that are not affected by the updated allocation strategy are not required to alter their current decisions. The decisions for users affected by the latest allocation strategy update need to be updated. For example, say users u_1 and u_2 both want to be allocated to the same channel in the same edge server. After user u_2 is allocated to this server in the current iteration of the game, this server is now exhausted of computing resources and insufficient to server user u_1 . Consequently, the decision for user u_1 needs to be updated with a new pair of edge server and channel.

The procedure of finding the best allocation decision for each user (Lines 5-16) will now be discussed in more detail. In each iteration, the best allocation decision for each user u_i is determined by going through every channel in every u_i 's neighbor edge server (Lines 7-11). The pair of channel and edge server that incurs the lowest system cost, which can be calculated by (14) (Line 9), is selected for user u_i (Line 12). Next, if user u_i is not already allocated to this channel in this edge server, and the new system cost is lower than the current system cost, this pair of channel and edge server will be submitted for the decision update opportunity (Lines 13-15). If selected (by the mechanism previously discussed, Line 17), the decision update for user u_i will be applied. Should a better allocation decision for user u_i be found in consequent iterations, it will again be submitted to be updated. It is important to note that the decision update process for each user (Lines 6-11) in an iteration can be executed in parallel because the process for a user does not rely on other users' processes. Moreover, for each individual user, the search for the best pair of channel and edge server (Lines 7-11) can also be parallelized.

After the user allocation process completes, Algorithm 2 (Appendix H, adopted in [35], [40]) is executed to allocate transmit power to users (Line 20). Its objective is to ensure data rates for all the users at a minimum overall transmit power cost. It is in line with the power allocation problem formulated in Section 5.2. This is a decentralized and iterative algorithm that has been shown to eventually converge to the global optimal power allocation solution, if one exists [35], [40], [41]. Please note that other power allocation algorithms can also be employed instead of Algorithm 2 to achieve different optimization objectives, e.g., maximum overall data rate, maximum energy efficiency, etc., without impacting the correctness of miUA.

6.3 Performance Analysis

6.3.1 Convergence Analysis

The convergence time of miUA is evaluated by the number of iterations T taken to find a Nash equilibrium. Theorem 2 proves the upper bound of T .

Theorem 2. *The convergence time of Algorithm 1 is upper bounded by:*

$$T \leq \frac{N(\epsilon \sum_{t \in \mathcal{T}} (r^t w_{max}^t) + \epsilon I_{max})}{|h_{min}|^2 p} \quad (20)$$

where $|h_{min}|^2$ is the minimum channel gain that a user could experience, and p is the default transmit power allocated to users

during the user allocation process.

Proof: Please refer to Appendix E. \square

6.3.2 Complexity Analysis

In this section, we analyze the worst-case time complexity of miUA (Algorithm 1). Its *sequential* time complexity is $\mathcal{O}(T_{max} N^3 M V \log N)$. T_{max} is the maximum number of iterations taken by miUA to find a Nash equilibrium, which has been found in Theorem 2. In each iteration, a user (there is a maximum of N users) finds the best allocation decision by enumerating each channel in every neighbor edge server (Lines 7-11 of Algorithm 1) to find a channel in an edge server that produces the lowest system cost and contend for the decision update (Line 14 of Algorithm 1). The maximum number of options that this user might have to go through is M neighbor servers $\times V$ channels. In practice, the number of a user's neighbor edge servers is much less than M . In our experiments, a user has at most 4 neighbor edge servers. For each of these options, system cost calculation requires calculating the computing resource cost and interference cost for N users, each costing $N \log N$. It is dominated by the calculation of intra-cell interference, which involves the sorting of users by their channel conditions. Calculating computing resource cost and inter-cell interference is less expensive. Thus, miUA costs $\mathcal{O}(T_{max} N^3 M V \log N)$ if executed sequentially, which is very computationally expensive. Fortunately, the system cost calculation for an option for a user is independent of the calculation for other options. Thus, miUA can be executed in parallel, reducing the time complexity to approximately $\frac{\mathcal{O}(T_{max} N^3 M V \log N)}{\rho}$, where ρ is the total number of edge servers, or processing threads to be specific, running miUA collectively. Our experiments in Section 7.3.2 empirically demonstrate the efficiency of miUA when executed in parallel.

6.3.3 Performance Bounds

Theorem 3 analyzes the lower and upper bounds on the number of allocated users in each edge server.

Theorem 3. *Let $num_j(\mathbf{a})$ be the number of users allocated to edge server s_j given a user allocation strategy \mathbf{a} , $num_j(\mathbf{a})$ satisfies:*

$$\begin{aligned} (num_j(\mathbf{a}) + 1)(1 + \log_2(num_j(\mathbf{a}))) &\geq \left\lfloor \frac{Q_j^t}{w_{max}} \right\rfloor \\ num_j(\mathbf{a})(1 + \log_2(num_j(\mathbf{a}))) &\leq \left\lfloor \frac{Q_j^t}{w_{min}} \right\rfloor \end{aligned}$$

Proof: Please refer to Appendix F. \square

Next, we analyze the theoretical optimality of miUA using its Price of Anarchy (PoA), which is a critical optimality indicator for game-theoretic approaches [16], [28], [29], [31]. It shows the ratio between the worst Nash equilibrium and the optimal solution [38]. The PoA is defined as: $PoA = \frac{\max_{\mathbf{a} \in \mathcal{A}} C_{\mathbf{a}, \mathbf{p}}}{C_{\mathbf{a}^{OPT}, \mathbf{p}^{OPT}}}$, where \mathcal{A} is the set of all the user allocation strategies that lead to Nash equilibria, and $(\mathbf{a}^{OPT}, \mathbf{p}^{OPT})$ is the optimal allocation strategy.

Theorem 4. In the user allocation game Γ , the PoA in terms of the system cost satisfies:

$$1 \leq \text{PoA} \leq \frac{\sum_{u_i \in U} (\eta_1 \epsilon \sum_{t \in \mathcal{T}} \tau^t w_{max}^t + \eta_2 \epsilon I_{max})}{\sum_{u_i \in U} \left(\eta_1 \sum_{t \in \mathcal{T}} \tau^t (1 + \log_{z_i^t}(\text{num}_j^*(a^{OPT}))) w_i^t + \eta_2 \sigma^2 \right)} \quad (21)$$

Proof: Please refer to Appendix G. \square

7 PERFORMANCE EVALUATION

We perform a series of experiments to evaluate miUA against three baseline approaches and two state-of-the-art approaches.

7.1 Performance Benchmark

We compare miUA against five representative approaches:

- **Optimal:** To find optimal solutions to the mixed-integer programming NOMA-EUA problem introduced in Section 5, we use IBM ILOG CP Optimizer solver². Due to the sophistication of the problem, this approach can only be executed in small-scale experiments.
- **Serving Channel Gain-based Subchannel Allocation (SCG-SA)** [42]: This approach solves the channel and power allocation problem in a NOMA-based cellular network to increase the energy efficiency. For each channel, SCG-SA ranks users based on their channel conditions. Users with stronger channel conditions are allocated first because they consume less transmit power. SCG-SA is designed to operate in a pure cellular network without edge servers, and thus does not consider the heterogeneity of edge servers with varying computing resources. Also, SCG-SA works on users that are already allocated to edge servers. Therefore, before allocating users to channels and allocating transmit power to users with SCG-SA, we employ a state-of-the-art user allocation approach (EUAGame) to allocate users to edge servers.
- **EUAGame** [16]: This approach allocates maximum users at the minimum computing resource cost by leveraging the multi-tenancy feature. However, it does not consider the communication/networking aspect. Thus, in the experiments, after users have been allocated to edge servers, we first allocate users to channels randomly and then perform a fixed power allocation where the total available transmit power of a base station is fully allocated to users based on their channel conditions. Assuming the users allocated to edge server s_j (\mathcal{U}_j) are indexed by order of channel condition, where users u_1 and $u_{|\mathcal{U}_j|}$ have the best and worst channel conditions, respectively, the transmit power allocated to user $u_i \in \mathcal{U}_j$ is then $P_j i / \sum_{x=1}^{|\mathcal{U}_j|} x$.
- **NearestUA:** This is a naive baseline approach that allocates each user to their nearest edge server that has sufficient computing resources. The rationale behind

2. www.ibm.com/au-en/analytcs/cplex-cp-optimizer

TABLE 1: Experiment settings

Cell radius (R_j)	289m
Inter-site distance	500m
Minimum distance between user and edge server	35m
Large-scale path loss model	$128.1 + 37.6 \log_{10}(d_{j,i})$ dB
Base station's maximum transmit power (P_j)	46dBm
Thermal noise density	-174dBm/Hz
System bandwidth (B)	10MHz

this approach is that a short distance between a user and an edge server usually results in a strong channel condition. After that, each user is allocated to the channel with the fewest users. Then, this approach employs Algorithm 2 to allocate transmit power to users.

- **Random:** This baseline approach assigns each user to a random channel in a random edge server that has enough computing capacity. This approach also employs Algorithm 2 to allocate transmit power to users.

7.2 Experiment Settings

The experiment settings are compliant with the existing LTE specifications [43] and summarized in Table 1. Edge servers' available computing resources Q_j are randomly generated based on a normal distribution $\mathcal{N}(\mu, 10^2)$, where μ is the average capacity of each resource type in $\mathcal{T} = \{\text{CPU, GPU, RAM, storage}\}$, and the standard deviation is 5. We set the weight parameters $\eta_1 = \eta_2 = 0.5$. Users are randomly distributed within the coverage of those edge servers by following a uniform distribution. Users' required data rate r_{min} is set at 2Mbps. We assume that users have three possible levels of normalized computing resource requirements, $w_i \in \{< 1, 2, 1, 1 >, < 2, 1, 2, 2 >, < 3, 3, 2, 2 >\}$. We have conducted experiments with other resource requirements, which produce similar results, so we pick those three levels as the representatives. Each user's computing resource requirement is randomly selected from those three levels.

We carry out seven sets of experiments (Table 2). In each experiment set, we vary one parameter while fixing the others. The first three sets (Sets #1s, #2s, and #3s) are small-scale experiments since Optimal is extremely time-consuming and infeasible to run in large-scale scenarios (Sets #1, #2, #3, #4). To evaluate miUA, we compare the normalized system costs (computing resource and transmit power costs, the lower the better) achieved by all the approaches. We also break it down by comparing the number of allocated users and the total transmit power consumed by all users. We do not explicitly present the result of computing resource cost since it can be deduced from the number of allocated users, which is a more intuitive metric. The convergence time of miUA is also evaluated, which is an important machine-independent efficiency indicator for game-theoretical approaches [16], [28], [29], [44], [45].

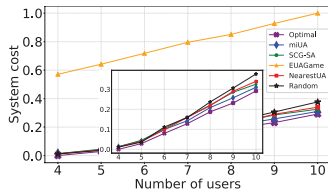


Fig. 2: System cost vs. number of users (Set #1s).

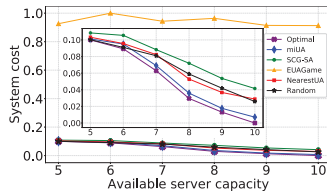


Fig. 3: System cost vs. number of users (Set #2s).

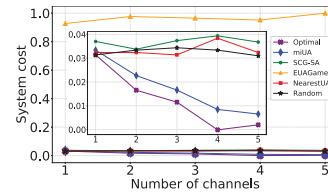


Fig. 4: System cost vs. number of users (Set #3s).

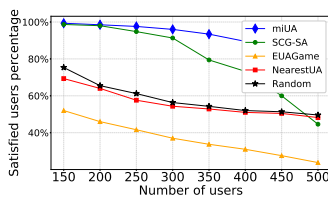


Fig. 5: Percentage of satisfied users vs. number of users (Set #1).

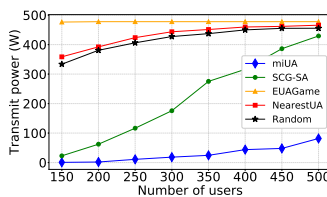


Fig. 6: Total transmit power vs. number of users (Set #1).

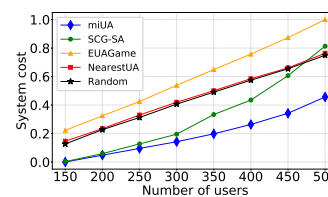


Fig. 7: System cost vs. number of users (Set #1).

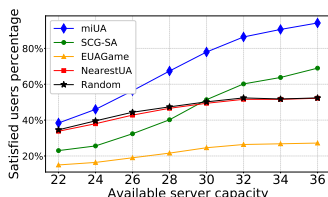


Fig. 8: Percentage of satisfied users vs. available server capacity (Set #2).

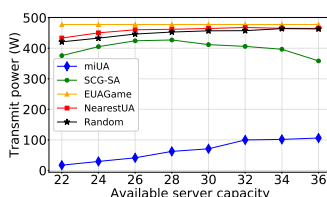


Fig. 9: Total transmit power vs. available server capacity (Set #2).

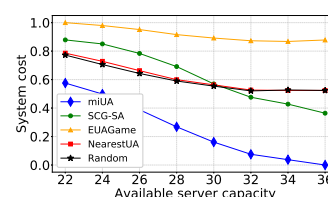


Fig. 10: System cost vs. available server capacity (Set #2).

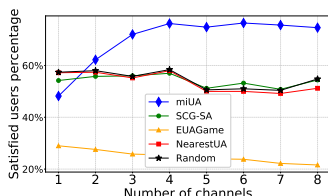


Fig. 11: Percentage of satisfied users vs. number of channels (Set #3).

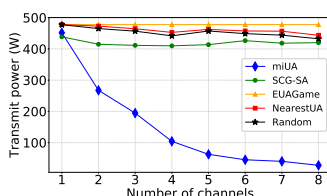


Fig. 12: Total transmit power vs. number of channels (Set #3).

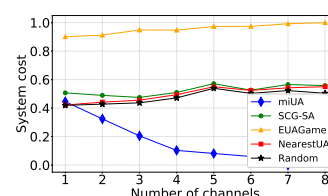


Fig. 13: System cost vs. number of channels (Set #3).

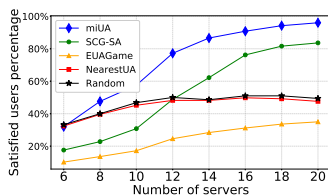


Fig. 14: Percentage of satisfied users vs. number of edge servers (Set #4).

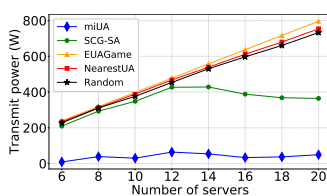


Fig. 15: Total transmit power vs. number of edge servers (Set #4).

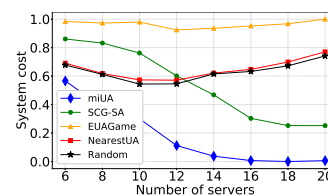


Fig. 16: System cost vs. number of edge servers (Set #4).

TABLE 2: Experiment sets

	Number of users N	Edge server capacity μ	Number of channels V	Number of edge servers M
Set #1s	4, ..., 10	8	4	2
Set #2s	10	5, ..., 10	4	2
Set #3s	10	8	1, ..., 5	2
Set #1	150, ..., 500	30	5	12
Set #2	500	22, ..., 36	5	12
Set #3	500	30	1, ..., 8	12
Set #4	500	30	5	6, ..., 20

7.3 Experimental Results

In general, miUA is able to allocate the most users at the lowest system cost in all the experiment sets, followed by SCG-GA. In particular, the total transmit power required by miUA is remarkably lower than any other approach. The performances of NearestUA and Random are comparable across all the parameter settings and all the experiments. EUAGame exhibits the worst performance, which shows that fixed power allocation is not suitable in NOMA-based MEC systems. The experimental results will be discussed in the following sections with more details.

7.3.1 Effectiveness

Experiment Sets #1s, #2s, and #3s (Figs 2, 3, and 4): In these small-scale experiments, the performance difference between the approaches is insignificant, except for EUAGame. Nevertheless, when zooming in the graphs (please refer to the subgraph within each figure), we can still notice that Optimal is obviously the best, closely followed by miUA. EUAGame always uses maximum transmit power, leading to extremely high system cost compared with the other approaches.

Experiment Set #1: We gradually increase the number of users from 150 to 500. As the number of users increases, the percentage of satisfied users by all the approaches decreases (Fig. 5). Since edge servers' capacities are fixed, adding more users to the system will quickly exhaust edge servers' computing resources, thus increasing the number of users that cannot be allocated to any edge servers or do not meet the minimum data rate requirement. Note that the *satisfied users* reported here (Fig.s 5, 8, 11, and 14) are the users who are allocated to edge servers *and* achieve the required minimum data rate. More details on the users, including those who are allocated but do not receive a satisfactory data rate, are shown in Fig.s 17, 18, 19, and 20.

Under all parameter settings, miUA is able to satisfy the most users. When there are not many users, SCG-SA is able to satisfy almost as many users as miUA. However, the difference between those two approaches grows noticeably larger as the number of users increases. EUAGame seems to be the worst because although it could allocate a good number of users to edge servers (even better than miUA in some cases), many of them do not achieve a satisfactory data rate (Fig. 17). This phenomenon also occurs in all the other experiment sets as depicted in Fig.s 17, 18, 19, and 20. Despite serving the most users among all the approaches, miUA consumes the least overall transmit power, lower than all other approaches by vast margins (Fig. 6). SCG-

SA starts with a relatively good performance. However, its performance degrades rapidly as the number of users increases. This shows that even though all the approaches (except EUAGame) employ a state-of-the-art power allocation mechanism, miUA still outperforms them by actively minimizing interference during the process of allocating users to edge servers and channels, saving a significant amount of transmit power. In overall, the system cost of miUA is the lowest (Fig. 7).

Experiment Set #2: We vary the average computing resource capacity μ available on edge servers from 22 to 36. As the computing resources become more abundant, edge servers can accommodate more users, leading to an increasing trend in the percentage of users allocated and satisfied by all the approaches (Fig.s 8 and 18). It can be seen that miUA vastly outperforms other approaches. In the beginning when the server capacity is limited, NearestUA and Random seem to outperform SCG-GA, which is no longer the case once the computing capacity increases thanks to the state-of-the-art server and channel selection methods employed by SCG-GA. miUA is again the approach that consumes the least power (Fig. 9) and incurs the lowest system cost (Fig. 10).

Experiment Set #3: We vary the number of channels in each base station from 1 to 8. In general, the performances of SCG-SA, EUAGame, NearestUA, and Random remain fairly constant (Fig. 13). In contrast, miUA is able to fully utilize the multi-channel characteristic. Given more channels, fewer users have to compete with each other, lowering the intra-cell and inter-cell interference in the system. As the number of channels increases, more users can achieve the minimum data rate required (Fig.s 11 and 19) with remarkable transmit power savings (Fig. 12). When the number of channels becomes adequately large (from 5 channels onward), the number of satisfied users and the overall transmit power start to converge without noticeable improvement. The reason is that all the allocated users have already achieved a sufficient data rate and the computing resources on edge servers are too exhausted to accommodate more users (Fig. 19).

Experiment Set #4: The number of edge servers is varied from 6 to 20. Extra edge servers immediately increase the total amount of computing and communication resources available in the system. As a result, more users can be satisfied (Fig.s 14 and 20). EUAGame, NearestUA, and Random fail to take advantage of this so they only achieve a very minimal improvement in the number of satisfied users. Even without a noticeable increase in the number of satisfied users, the overall transmit power consumed by those three approaches still continues to increase roughly linearly with the number of edge servers (Fig. 15). miUA and SCG-SA can effectively leverage the increase in edge servers and show a clear improvement in the number of satisfied users. When the number of edge servers increases, users are less likely to be allocated to the same channel in general. This significantly reduces the intra-cell and inter-cell interference. As a result, miUA and SCG-SA do not require much more overall transmit power (Fig. 15) even when the number of satisfied users increases (Fig. 14). In general, the system cost achieved by miUA is far lower than all the other approaches (Fig. 16).

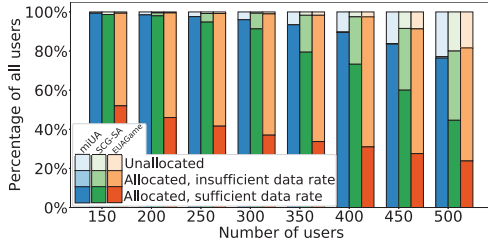


Fig. 17: Details of allocated users (Set #1).

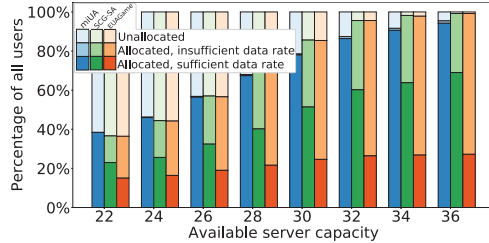


Fig. 18: Details of allocated users (Set #2).

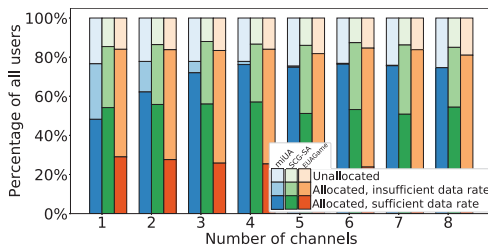


Fig. 19: Details of allocated users (Set #3).

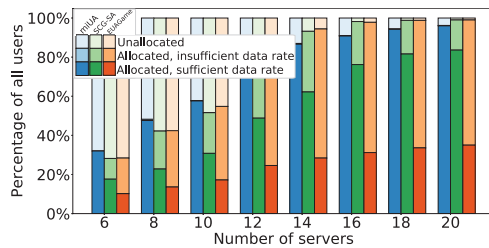


Fig. 20: Details of allocated users (Set #4).

7.3.2 Efficiency

Discussion: As shown by the four sets of experiments, miUA clearly outperforms all the other approaches by significant margins. It achieves the highest number of satisfied users, i.e., those allocated to edge servers with their data rate requirements fulfilled while saving a significant amount of transmit power. SCG-SA, NearestUA, and Random are beaten by miUA even though they all employ the same power allocation method that aims to minimize the overall transmit power while satisfying the data rate requirement. This demonstrates the importance of allocating individual users to appropriate edge servers and channels. The fixed power allocation scheme that assigns full transmit power on a base station to users based on their channel conditions, which is employed by EUAGame, is obviously not suitable in NOMA-based MEC systems. Evidently, the vast majority of users allocated by EUAGame do not receive a sufficient data rate (Figs 17, 18, 19, and 20).

To evaluate the efficiency of miUA, we measure its convergence time by the number of decision iterations it takes to reach a Nash equilibrium (Figs 21, 22, 23, and 24). This metric is widely used to evaluate the efficiency of game-theoretic approaches [16], [28], [29], [44], [45] due to its machine independence (different machines with different computing power might result in different CPU time taken to solve a problem). For reference, we also present the elapsed CPU time for all the approaches. The experiments are performed on a Linux machine equipped with 2 × Intel Gold 6140 18-core processors and 32GB RAM.

In all experiment sets, the complexity of the user allocation problem increases when the experiment parameter increases. Consequently, miUA gradually requires more iterations to reach a Nash equilibrium. Under the largest

experiment setting (20 edge servers, 5 channels per server, and 500 users), it takes almost 700 iterations, which is quite reasonable for the scale of the problem. With the increase in the number of iterations, the elapsed CPU time also increases (with the exception of experiment Set #3). Because of the complex calculation of intra-cell interference, inter-cell interference, and system cost, miUA is noticeably more computationally expensive than the other approaches and takes much more CPU time. However, it still falls within the acceptable range since based on our experiments, the greatest time taken by miUA is 80ms (in Set #2). In Set #3 when the number of channels increases, miUA gradually runs quicker despite the increasing number of iterations. This happens because when the number of channels is small, more users are allocated to each channel on average. This increases the complexity of the sorting of the users on each channel by their channel conditions, which is computationally demanding.

8 CONCLUSION AND FUTURE WORK

In this work, the edge user allocation problem in a downlink non-orthogonal multiple access (NOMA)-based mobile edge computing (MEC) system is investigated. To tackle this NP-hard problem, we model it as a potential game with the objective to maximize the number of allocated users at the minimum computing resource and transmit power costs while satisfying their data rate requirements. By jointly considering both the communication and computation aspects of a NOMA-based MEC system, miUA, our decentralized game-theoretic approach, greatly outperforms all the other approaches, being able to serve the most users with the least transmit power. Our experiments highlight the significance of incorporating wireless interference into the user

13

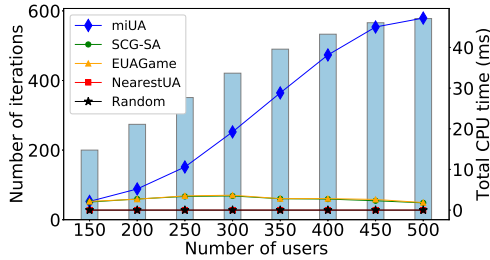


Fig. 21: Number of decision iterations and elapsed CPU time vs. number of users (Set #1).

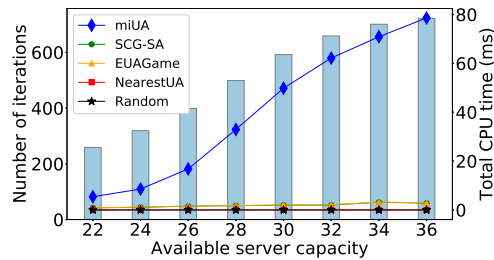


Fig. 22: Number of decision iterations and elapsed CPU time vs. Available server capacity (Set #2).

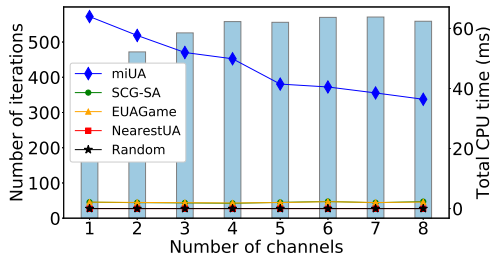


Fig. 23: Number of decision iterations and elapsed CPU time vs. number of channels (Set #3).

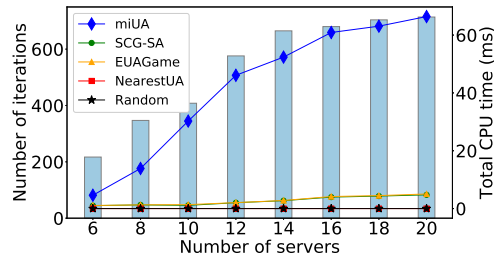


Fig. 24: Number of decision iterations and elapsed CPU time vs. number of edge servers (Set #4).

allocation approach in a NOMA-based MEC system. We also theoretically analyze the optimality and convergence of our proposed approach.

The future work has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

ACKNOWLEDGMENTS

This research is partly funded by Australian Research Council Discovery Project grants (DP170101932, DP180100212 and DP200102491), and Laureate Fellowship FL190100035. Qiang He is the corresponding author of this paper.

REFERENCES

- [1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.
- [2] "Ericsson Mobility Report," *Ericsson, Stockholm*, 2019. [Online]. Available: www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf
- [3] Y. Saito, Y. Kishiyama, A. Benjebbour, T. Nakamura, A. Li, and K. Higuchi, "Non-orthogonal multiple access (NOMA) for cellular future radio access," in *Proceedings of IEEE Vehicular Technology Conference*. IEEE, 2013, pp. 1–5.
- [4] A. Benjebbour, Y. Saito, Y. Kishiyama, A. Li, A. Harada, and T. Nakamura, "Concept and practical considerations of non-orthogonal multiple access (NOMA) for future radio access," in *Proceedings of International Symposium on Intelligent Signal Processing and Communication Systems*. IEEE, 2013, pp. 770–774.
- [5] L. Dai, B. Wang, Z. Ding, Z. Wang, S. Chen, and L. Hanzo, "A survey of non-orthogonal multiple access for 5G," *IEEE communications surveys & tutorials*, vol. 20, no. 3, pp. 2294–2323, 2018.

- [6] L. Dai, B. Wang, Y. Yuan, S. Han, I. Chih-Lin, and Z. Wang, "Non-orthogonal multiple access for 5G: solutions, challenges, opportunities, and future research trends," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 74–81, 2015.
- [7] M. Vaezi, G. A. A. Baduge, Y. Liu, A. Arafa, F. Fang, and Z. Ding, "Interplay between NOMA and other emerging technologies: A survey," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 900–919, 2019.
- [8] Z. Ding, P. Fan, and H. V. Poor, "Impact of non-orthogonal multiple access on the offloading of mobile edge computing," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 375–390, 2018.
- [9] —, "Impact of user pairing on 5G nonorthogonal multiple-access downlink transmissions," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6010–6023, 2015.
- [10] K. Wang, Y. Liu, Z. Ding, A. Nallanathan, and M. Peng, "User association and power allocation for multi-cell non-orthogonal multiple access networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5284–5298, 2019.
- [11] F. Guo, H. Lu, D. Zhu, and H. Wu, "Interference-aware user grouping strategy in NOMA systems with QoS constraints," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 1378–1386.
- [12] H. V. Nguyen, V.-D. Nguyen, O. A. Dobre, D. N. Nguyen, E. Dutkiewicz, and O.-S. Shin, "Joint power control and user association for NOMA-based full-duplex systems," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 8037–8055, 2019.
- [13] M. W. Baidas, Z. Bahbahani, and E. Alsusa, "A matching-theoretic approach to user-association and channel assignment in downlink multi-cell noma networks," in *Proceedings of International Conference on Communications and Networking (ComNet)*. IEEE, 2018, pp. 1–8.
- [14] C. Xu, G. Zheng, and L. Tang, "Energy-aware user association for noma-based mobile edge computing using matching-coalition game," *IEEE Access*, vol. 8, pp. 61943–61955, 2020.
- [15] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, H. Liu, Y. Qin, and P. Chen, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proceedings*

- of *IEEE International Conference on Web Services*. IEEE, 2019, pp. 91–98.
- [16] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, “A game-theoretical approach for user allocation in edge computing environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.
- [17] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, “Optimal edge user allocation in edge computing with variable sized vector bin packing,” in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2018, pp. 230–245.
- [18] G. Cui, Q. He, F. Chen, H. Jin, and Y. Yang, “Trading off between multi-tenancy and interference: A service user allocation game,” *IEEE Transactions on Services Computing*, 2020, doi: 10.1109/TSC.2020.3028760.
- [19] P. Lai, Q. He, J. Grundy, F. Chen, M. Abdelrazek, J. Hosking, J. Grundy, and Y. Yang, “Cost-effective app user allocation in an edge computing environment,” *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020, doi: 10.1109/TCC.2020.3001570.
- [20] G. Cui, Q. He, F. Chen, Y. Zhang, H. Jin, and Y. Yang, “Interference-aware game-theoretic device allocation for mobile edge computing,” *IEEE Transactions on Mobile Computing*, 2021, doi: 10.1109/TMC.2021.3064063.
- [21] D. Sabella, V. Sukhomlinov, L. Trang, S. Kekki, P. Paglierani, R. Rossbach, X. Li, Y. Fang, D. Druta, F. Giust *et al.*, “Developing software for multi-access edge computing,” *ETSI white paper*, vol. 20, pp. 1–38, 2019. [Online]. Available: www.etsi.org/images/files/ETSIWhitePapers/etsi_wp20ed2_MEC_SoftwareDevelopment.pdf
- [22] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, “MEC in 5G networks,” *ETSI white paper*, vol. 28, pp. 1–28, 2018. [Online]. Available: www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf
- [23] F. Chong and G. Carraro, “Architecture strategies for catching the long tail,” *MSDN Library, Microsoft Corporation*, pp. 9–10, 2006.
- [24] B. P. Rimal and M. Maier, “Workflow scheduling in multi-tenant cloud computing environments,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 290–304, 2016.
- [25] S. Srikantaiah, A. Kansal, and F. Zhao, “Energy aware consolidation for cloud computing,” in *Proceedings of HotPower Workshop on Power Aware Computing and Systems*. USENIX, 2008, pp. 10–15.
- [26] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, “Multi-hop cooperative computation offloading for industrial IoT–edge-cloud computing environments,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.
- [27] Z. Xu, L. Zhou, S. C.-K. Chau, W. Liang, Q. Xia, and P. Zhou, “Collaborate or Separate? Distributed Service Caching in Mobile Edge Clouds,” in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2020, pp. 2066–2075.
- [28] 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.
- [29] 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.
- [30] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang, and S. Ou, “Dynamic resource scheduling in mobile edge cloud with cloud radio access network,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2429–2445, 2018.
- [31] 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.
- [32] S. R. Islam, N. Avazov, O. A. Dobre, and K.-S. Kwak, “Power-domain non-orthogonal multiple access (NOMA) in 5G systems: Potentials and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 721–742, 2016.
- [33] J. Choi, “Minimum power multicast beamforming with superposition coding for multiresolution broadcast and application to NOMA systems,” *IEEE Transactions on Communications*, vol. 63, no. 3, pp. 791–800, 2015.
- [34] Z. Ding, Z. Yang, P. Fan, and H. V. Poor, “On the performance of non-orthogonal multiple access in 5G systems with randomly deployed users,” *IEEE Signal Processing Letters*, vol. 21, no. 12, pp. 1501–1505, 2014.
- [35] Z. Yang, C. Pan, W. Xu, Y. Pan, M. Chen, and M. Elkashlan, “Power control for multi-cell networks with non-orthogonal multiple access,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 927–942, 2017.
- [36] G. Velkoski, M. Simjanoska, S. Ristov, and M. Gusev, “CPU utilization in a multitenant cloud,” in *Eurocon 2013*. IEEE, 2013, pp. 242–249.
- [37] M. R. Garey and D. S. Johnson, *Computers and intractability*. Freeman San Francisco, 1979, vol. 174.
- [38] D. Monderer and L. S. Shapley, “Potential games,” *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996.
- [39] M. J. Osborne and A. Rubinstein, *A course in game theory*. MIT Press, 1994.
- [40] Y. Fu, Y. Chen, and C. W. Sung, “Distributed power control for the downlink of multi-cell noma systems,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 6207–6220, 2017.
- [41] R. D. Yates, “A framework for uplink power control in cellular radio systems,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1341–1347, 1995.
- [42] H. Zeng, X. Zhu, Y. Jiang, Z. Wei, and T. Wang, “A green coordinated multi-cell noma system with fuzzy logic based multicriterion user mode selection and resource allocation,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 3, pp. 480–495, 2019.
- [43] E. 3GPP, “LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Frequency (RF) requirements for LTE Pico Node B (3GPP TR 36.931 version 9.0.0 Release 9),” Tech. Rep., 2011.
- [44] 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.
- [45] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, “A cyclic game for joint cooperation and competition of edge resource allocation,” in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2019, pp. 503–513.

The author biographies has been removed since the biographies have no close connection with the thesis. Please refer to the original paper online for a full, unedited version.

**APPENDIX A
KEY NOTATIONS**

Key notations used in this article are summarized in Table 3.

TABLE 3: Key Notations

Symbol	Description
$\mathbf{a}_{j,i}^k$	allocation decision on whether user u_i will be allocated to the k -th channel of edge server s_j
B_j^k	bandwidth of channel c_j^k in edge server s_j
C_j	the set of channels $c_j^k, k \in \{1, 2, \dots, V\}$, in edge server s_j
$C_{a,p}$	total system cost given a user allocation strategy \mathbf{a} and a power allocation strategy \mathbf{p}
$C_{a,p}(\mathbf{a}_i)$	system cost incurred by user u_i 's decision \mathbf{a}_i
$d_{j,i}$	distance between user u_i and edge server s_j
$f_{j,i}^t$	the utilization of computing resource type t on edge server s_j when user u_i is being allocated
$ h_{j,i}^k ^2$	channel gain of user u_i on channel c_j^k of edge server s_j
$I_{j,i}^k$	inter-cell interference experienced by user u_i on c_j^k
$I_{a,p}(u_i)$	interference cost incurred by user u_i given a user allocation strategy \mathbf{a} and power allocation strategy \mathbf{p}
$M_a(u_i)$	computing resource cost incurred by user u_i given an allocation strategy \mathbf{a}
P_j	maximum transmit power (power capacity) of edge server s_j
$\mathbf{p}_{j,i}^k$	allocation decision on the amount of transmit power allocated to user u_i on the k -th channel of edge server s_j
p_j^k	transmit power of edge server s_j on channel c_j^k
Q_j	computing capacity of edge server s_j . Q_j is a $ \mathcal{T} $ -dimensional vector
R_j	cell radius of edge server s_j
$r_{j,i}^k$	achievable data rate of user u_i on channel c_j^k
\mathcal{S}	the set of edge servers $s_j, j \in \{1, 2, \dots, M\}$
\mathcal{S}_i	set of user u_i 's neighbor edge servers
\mathcal{T}	the set of computing resource types, or computing capacity dimensions. $\mathcal{T} = \{\text{CPU, RAM, storage, ...}\}$
\mathcal{U}	the set of users $u_i, i \in \{1, 2, \dots, N\}$
U_j	set of users allocated to edge server s_j
U_j^k	set of users allocated to edge server s_j on channel c_j^k
w_i	computing resource requirement of user u_i . w_i is a $ \mathcal{T} $ -dimensional vector
z_i^t	computation task size and computing resource type dependent parameter used in resource utilization model
$\Theta(c_j^k)$	SIC decoding order of users on channel c_j^k
r_{min}	minimum user data rate requirement
τ^t	a weight indicating app vendor's priority of saving computing resource type $t \in \mathcal{T}$
η_1, η_2	weights of computing resource cost and interference cost in the whole system cost
ϵ, ε	weights indicating the severity of the penalty on computing resource cost and interference cost when a user is unallocated

**APPENDIX B
PROOF OF NP-HARDNESS OF NOMA-EUA**

Proof: The user allocation problem modeled in Section 5.1 can be proved to be NP-hard by proving that its associated decision version is NP-complete. Its decision version is defined as follows:

Given a set of all the users $\mathcal{U} = \{u_1, \dots, u_N\}$ and a set of all the channels in all edge server $\mathcal{C} = \{C_j | s_j \in \mathcal{S}\}$, for each positive number M , determine whether there exists a

partition of $\mathcal{U}' \subseteq \mathcal{U}$ into $\mathcal{C}' \subseteq \mathcal{C}$ with a cost (computing resource and interference costs (14)) lower than M , while satisfying constraints (13b) and (13c). By repeatedly solving the decision problem with all feasible combination of users and channels, it is possible to find the user allocation that incurs the lowest cost (14).

First, we show that the EUA problem is NP. We can easily validate a solution of the EUA problem in polynomial time – ensuring that each user is allocated to at most one channel in an edge server, and the aggregated computing resources consumed by all the users in an edge server does not exceed its capacity. This problem is thus in NP class.

Then, the EUA problem can be proven to be NP-hard by reducing the NP-complete PARTITION problem [37] to a special case of the user allocation decision problem. The PARTITION problem is defined as follows: Given a finite series of non-negative integers $\mathcal{W} = (w_1, w_2, \dots, w_N)$, determine if there exists a subset $\mathcal{S} \subseteq \{1, \dots, N\}$ such that $\sum_{i \in \mathcal{S}} w_i = \sum_{j \notin \mathcal{S}} w_j$.

Each user u_i can be either unallocated, or allocated to a channel in an edge server. For any instance $\mathcal{W} = (w_1, w_2, \dots, w_N)$ of PARTITION, construct the following instance of the user allocation problem with N users, for each user u_i , there are two possible 2-dimensional computing resource requirements, $\langle w_i, 0 \rangle$ and $\langle 0, w_i \rangle$; and a number of identical servers whose size is $\langle Q, Q \rangle$, where $Q = \frac{\sum_{i=1}^N w_i}{2}$. Assume that all users can be served by any of those servers. Note that $\langle w_i, 0 \rangle \equiv \langle 0, w_i \rangle \equiv u_i$, which represents a user's computing resource requirement. Clearly, there is a solution to the user allocation problem which allocates N users to two servers if and only if there is a solution to the PARTITION problem. Therefore, this special case is NP-hard. This, combined with the fact that it is NP as shown above, proves that the associated decision version of the user allocation problem is NP-complete. Because an optimization problem is at least as hard as its decision version, the user allocation problem defined in Section 5.1 is thus NP-hard. Therefore, the NOMA-EUA problem is also NP-hard, which completes the proof. \square

**APPENDIX C
PROOF OF LEMMA 1**

Proof: If the allocation decision \mathbf{a}_i^* made for user u_i is not the best decision in \mathcal{A}_i , there must be a better decision $\mathbf{a}_i \in \mathcal{A}_i$ that decreases the system cost, i.e., $C_{a^*,p}(\mathbf{a}_i^*) > C_{a^*,p}(\mathbf{a}_i)$. As a result, switching from \mathbf{a}_i^* to \mathbf{a}_i will lead to a lower system cost. This is in contradictory to (17), where no user in a Nash equilibrium can unilaterally lower the system cost. \square

**APPENDIX D
PROOF OF THEOREM 1**

Proof: For ease of exposition, here we assume all users have the same computing resource requirement w_i , i.e., $w_i = w, \forall u_i \in \mathcal{U}$, and have the same z_i^t , i.e., $z_i^t = z^t, \forall u_i \in \mathcal{U}$. Also note that during the user allocation process, all users are allocated an identical transmit power p . The existence of a Nash equilibrium where different users have different

computing resource requirements and computation task sizes are experimentally validated in Section 7.

Let us suppose the current decision $\mathbf{a}_i = (s_j, c_j^k)$ of user u_i and its updated decision $\mathbf{a}'_i = (s'_j, c_{j'}^{k'})$ (the decision made in the next iteration in Algorithm 1) satisfy $C_{a,p}(\mathbf{a}_i) \geq C_{a',p}(\mathbf{a}'_i)$. According to (12), (14), and (15), there are two possible cases: 1) u_i moves from an edge server to another edge server, and 2) u_i moves from being unallocated to being allocated. The case where user u_i switches to another channel in the same edge server is a special case of Case 1, where the computing resource cost remains unchanged.

Case 1: User u_i 's transition from an edge server to another edge server, $\sum_{c_j^k \in C_j} \mathbf{a}_{j,i}^k = 1$ and $\sum_{c_{j'}^{k'} \in C_{j'}} \mathbf{a}'_{j,i}{}^{k'} = 1$:

Condition $C_{a,p}(\mathbf{a}_i) \geq C_{a',p}(\mathbf{a}'_i)$ implies:

$$\begin{aligned} \sum_{u_q \in \mathcal{U}_j \cup \mathcal{U}_{j'}} (\eta_1 M_a(u_q) + \eta_2 I_{a,p}(u_q)) &\geq \sum_{u_q \in \mathcal{U}_j \cup \mathcal{U}_{j'}} (\eta_1 M_{a'}(u_q) \\ &\quad + \eta_2 I_{a',p}(u_q)) \\ \Leftrightarrow \sum_{u_q \in \mathcal{U}_j \cup \mathcal{U}_{j'}} \left(\eta_1 (M_a(u_q) - M_{a'}(u_q)) \right. \\ &\quad \left. + \eta_2 (I_{a,p}(u_q) - I_{a',p}(u_q)) \right) \geq 0 \end{aligned} \quad (22)$$

Based on (12), there is:

$$\begin{aligned} \sum_{u_q \in \mathcal{U}_j \cup \mathcal{U}_{j'}} \eta_1 (M_a(u_q) - M_{a'}(u_q)) \\ = \eta_1 \left(\sum_{u_i \in \mathcal{U}_j} \sum_{t \in \mathcal{T}} \tau^t \log_{z_t^i}(|\mathcal{U}_j|) w_i^t - \sum_{u_o \in \mathcal{U}_{j'}} \sum_{t \in \mathcal{T}} \tau^t \log_{z_o^t}(|\mathcal{U}_{j'}|) w_o^t \right) \end{aligned} \quad (23)$$

Let h denote the channel gain of a user on its assigned channel. Based on (15), there is:

$$\begin{aligned} \sum_{u_q \in \mathcal{U}_j \cup \mathcal{U}_{j'}} \eta_2 (I_{a,p}(u_q) - I_{a',p}(u_q)) \\ = \sum_{u_q \in \mathcal{U}_j \cup \mathcal{U}_{j'}} \eta_2 \left(h \sum_{b=q+1}^{|\mathcal{U}_j|} p - h \sum_{b'=q+1}^{|\mathcal{U}_{j'}|} p \right. \\ \left. + \sum_{s_l \in \mathcal{S}_q \setminus \{s_j\}} h p_l^k - \sum_{s_{l'} \in \mathcal{S}_{q'} \setminus \{s_{j'}\}} h p_{l'}^{k'} + \sigma^2 - \sigma^2 \right) \end{aligned} \quad (24)$$

Given (22), (23), and (24), we can see that $C_{a,p}(\mathbf{a}_i) \geq C_{a',p}(\mathbf{a}'_i)$ implies:

$$\begin{aligned} \eta_1 \left(\sum_{u_i \in \mathcal{U}_j} \sum_{t \in \mathcal{T}} \tau^t \log_{z_t^i}(|\mathcal{U}_j|) w_i^t - \sum_{u_o \in \mathcal{U}_{j'}} \sum_{t \in \mathcal{T}} \tau^t \log_{z_o^t}(|\mathcal{U}_{j'}|) w_o^t \right) \\ + \sum_{u_q \in \mathcal{U}_j \cup \mathcal{U}_{j'}} \eta_2 \left(h \sum_{b=q+1}^{|\mathcal{U}_j|} p - h \sum_{b'=q+1}^{|\mathcal{U}_{j'}|} p \right. \\ \left. + \sum_{s_l \in \mathcal{S}_q \setminus \{s_j\}} h p_l^k - \sum_{s_{l'} \in \mathcal{S}_{q'} \setminus \{s_{j'}\}} h p_{l'}^{k'} \right) \geq 0 \end{aligned} \quad (25)$$

Based on the definition of the potential game, we need to show that the cost decrease (both computing resource and interference costs) caused by a user u_i updating its current decision \mathbf{a}_i to a new decision \mathbf{a}'_i will result in a decrease in

the potential function (19). Given the potential function (19), there is:

$$\begin{aligned} \phi_{\mathbf{a}_i}(\mathbf{a}_i) - \phi_{\mathbf{a}_i}(\mathbf{a}'_i) \\ = \eta_1 \sum_{u_q \in \mathcal{U}_j} \sum_{t \in \mathcal{T}} \tau^t \left(\log_{z_t^i}(|\mathcal{U}_j|) w_q^t - \log_{z_t^i}(|\mathcal{U}_{j'}|) w_q^t \right) \\ + \eta_2 \sum_{u_q \in \mathcal{U}_j} \left(|h_{j,q}^k|^2 \sum_{b=q+1}^{|\mathcal{U}_j|} p_{j,q}^k + \sum_{s_l \in \mathcal{S}_q \setminus \{s_j\}} |h_{l,q}^k|^2 p_l^k + \sigma^2 \right) \\ - \eta_2 \sum_{u_q \in \mathcal{U}_{j'}} \left(|h_{j',q}^{k'}|^2 \sum_{b'=q'+1}^{|\mathcal{U}_{j'}|} p_{j',q}^{k'} \right. \\ \left. + \sum_{s_{l'} \in \mathcal{S}_{q'} \setminus \{s_{j'}\}} |h_{l',q'}^{k'}|^2 p_{l'}^{k'} + \sigma^2 \right) \end{aligned} \quad (26)$$

Based on (25), we can see $\phi_{\mathbf{a}_i}(\mathbf{a}_i) - \phi_{\mathbf{a}_i}(\mathbf{a}'_i) \geq 0$ when $C_{a,p}(\mathbf{a}_i) - C_{a',p}(\mathbf{a}'_i) \geq 0$.

Case 2: User u_i 's transition from being unallocated to allocated, $\sum_{s_j \in \mathcal{S}} \sum_{c_j^k \in C_j} \mathbf{a}_{j,i}^k = 0$ and $\sum_{c_{j'}^{k'} \in C_{j'}} \mathbf{a}'_{j,i}{}^{k'} = 1$: We can use the similar argument in Case 1 to show that $C_{a,p}(\mathbf{a}_i) - C_{a',p}(\mathbf{a}'_i) \geq 0$ leads to $\phi_{\mathbf{a}_i}(\mathbf{a}_i) - \phi_{\mathbf{a}_i}(\mathbf{a}'_i) \geq 0$. Combined this with the result in Case 1, we can conclude that our user allocation game is an ordinal potential game. \square

APPENDIX E PROOF OF THEOREM 2

Proof: According to (19), we have:

$$0 \leq \phi(\mathbf{a}) \leq N \left(\epsilon \sum_{t \in \mathcal{T}} (\tau^t w_{max}^t) + \epsilon I_{max} \right) \quad (27)$$

because the term $\sum_{u_i \in \mathcal{U}} (\epsilon \sum_{t \in \mathcal{T}} (\tau^t w_{max}^t) + \epsilon I_{max}) \mathbb{1}_{\sum_{s_j \in \mathcal{S}} \sum_{c_j^k \in C_j} \mathbf{a}_{j,i}^k = 0}$ in the potential function $\phi(\mathbf{a})$ is always greater than the other terms combined.

In each iteration decision, a user $u_i \in \mathcal{U}$ updates its current decision \mathbf{a}_i to a new decision \mathbf{a}'_i to decrease the computing resource and interference costs, represented by $C_{a,p}(\mathbf{a}_i) - C_{a',p}(\mathbf{a}'_i)$. Next, we need to find the minimum possible value of $C_{a,p}(\mathbf{a}_i) - C_{a',p}(\mathbf{a}'_i)$, i.e., the minimum cost decrease that might occur when a user changes its decision.

We can easily see that the minimum cost decrease happens when a user's decision update meets all the following conditions:

- The user is already allocated to an edge server and does not wish to move to another server. Thus, the computing resource cost remains unchanged when \mathbf{a}_i is updated to \mathbf{a}'_i .
- The user only wishes to switch to another channel to lower its intra-cell interference. The minimum improvement of intra-cell interference occurs when the new channel has one user fewer than the current channel. The intra-cell interference incurred by this one user is $|h_{min}|^2 p$.
- The user is located in the middle of its current cell. In other words, it is not covered by any other edge server. Thus, this user does not suffer from inter-cell interference.

When the above conditions are satisfied, the minimum cost decrease is then $|h_{min}|^2 p$. As a result, miUA will terminate by driving the potential function to a minimum point (Nash equilibrium) within at most $\frac{N(\epsilon \sum_{t \in \mathcal{T}} (\tau^t w_{max}^t) + \epsilon I_{max})}{|h_{min}|^2 p}$ iterations. \square

APPENDIX F PROOF OF THEOREM 3

Proof: For ease of exposition, let $\tau^t = 1, \forall t \in \mathcal{T}$, and $z_i^t = z, \forall t \in \mathcal{T}, \forall u_i \in \mathcal{U}$. For a user u_i to be allocated to edge server s_j , server s_j must have sufficient computing resources. This can be expressed by:

$$\sum_{u_q \in \mathcal{U}_j \setminus \{u_i\}} (1 - f_{j,q}) w_q + (1 - f_{j,i}) w_i \leq Q_j \quad (28)$$

where the left-hand side of the inequality is the total resource consumption of all existing users in edge server s_j and user u_i . First, we find the upper bound on the number of allocated users. Let $num_j^*(\mathbf{a}^{OPT}) = |\mathcal{U}_j|$ be the maximum number of allocated users on edge server s_j produced by a centralized optimal solution \mathbf{a}^{OPT} . Then, (28) implies:

$$\begin{aligned} & (num_j^*(\mathbf{a}^{OPT}) - 1)(1 + \log_z(num_j^*(\mathbf{a}^{OPT}))) w_{min} \\ & \quad + (1 + \log_z(num_j^*(\mathbf{a}^{OPT}))) w_{min} \leq Q_j \\ \Leftrightarrow & num_j^*(\mathbf{a}^{OPT})(1 + \log_z(num_j^*(\mathbf{a}^{OPT}))) \leq \left\lfloor \frac{Q_j}{w_{min}} \right\rfloor \end{aligned} \quad (29)$$

Now we find the lower bound on the number of allocated users. For an arbitrary Nash equilibrium \mathbf{a} , there is at least one unallocated user u_i (if this user was allocated too then this would have been an optimal solution, here we investigate the case where the solution is sub-optimal). Since this is a Nash equilibrium, user u_i cannot select any server. This leads to:

$$\begin{aligned} & \sum_{u_q \in \mathcal{U}_j} (1 - f_{j,q}) w_q + (1 - f_{j,i}) w_i \geq Q_j \\ \Leftrightarrow & num_j(\mathbf{a})(1 + \log_z(num_j(\mathbf{a}))) w_{max} \\ & \quad + (1 + \log_z(num_j(\mathbf{a}))) w_{max} \geq Q_j \\ \Leftrightarrow & (num_j(\mathbf{a}) + 1)(1 + \log_z(num_j(\mathbf{a}))) \geq \left\lfloor \frac{Q_j}{w_{max}} \right\rfloor \end{aligned} \quad (30)$$

The combination of (29) and (30) completes the proof. \square

APPENDIX G PROOF OF THEOREM 4

Proof: Case 1: For any allocation strategy (\mathbf{a}, \mathbf{p}) , its system cost is clearly always higher than the system cost of an optimal allocation strategy, i.e., $C_{\mathbf{a}, \mathbf{p}} \geq C_{\mathbf{a}^{OPT}, \mathbf{p}^{OPT}}$, hence $PoA \geq 1$.

Case 2: Next, we analyze the bounds on the system cost of an arbitrary Nash equilibrium and an optimal solution.

For an arbitrary Nash equilibrium (\mathbf{a}, \mathbf{p}) , its system cost $C_{\mathbf{a}, \mathbf{p}}$ always satisfies:

$$\begin{aligned} C_{\mathbf{a}, \mathbf{p}} & \leq \max_{\mathbf{a}, \mathbf{p} \in \mathcal{A}} \sum_{u_i \in \mathcal{U}} (\eta_1 M_{\mathbf{a}}(u_i) + \eta_2 I_{\mathbf{a}, \mathbf{p}}(u_i)) \\ & \stackrel{\dagger}{\leq} \sum_{u_i \in \mathcal{U}} (\eta_1 \epsilon \sum_{t \in \mathcal{T}} \tau^t w_{max}^t + \eta_2 \epsilon I_{max}) \end{aligned} \quad (31)$$

where the inequality \dagger is because of the computing resource and interference cost models (12) and (15).

For an optimal solution $(\mathbf{a}^{OPT}, \mathbf{p}^{OPT})$, its system cost $C_{\mathbf{a}^{OPT}, \mathbf{p}^{OPT}}$ always satisfies:

$$\begin{aligned} C_{\mathbf{a}^{OPT}, \mathbf{p}^{OPT}} & = \sum_{u_i \in \mathcal{U}} (\eta_1 M_{\mathbf{a}^{OPT}}(u_i) + \eta_2 I_{\mathbf{a}^{OPT}, \mathbf{p}^{OPT}}(u_i)) \\ & \stackrel{\ddagger}{\geq} \sum_{u_i \in \mathcal{U}} \left(\eta_1 \sum_{t \in \mathcal{T}} \tau^t (1 + \log_{z_i^t}(num_j^*(\mathbf{a}^{OPT}))) w_i^t + \eta_2 \sigma^2 \right) \end{aligned} \quad (32)$$

where $num_j^*(\mathbf{a}^{OPT})$ is the maximum number of allocated users on edge server s_j produced by an optimal solution \mathbf{a}^{OPT} . The inequality \ddagger is because of the computing resource and interference cost models (12) and (15), and the fact that $I_{\mathbf{a}^{OPT}, \mathbf{p}^{OPT}}(u_i) \geq \sigma^2$.

Since $C_{\mathbf{a}, \mathbf{p}} \geq C_{\mathbf{a}^{OPT}, \mathbf{p}^{OPT}}$, combined with (31) and (32), we have:

$$PoA \leq \frac{\sum_{u_i \in \mathcal{U}} (\eta_1 \epsilon \sum_{t \in \mathcal{T}} \tau^t w_{max}^t + \eta_2 \epsilon I_{max})}{\sum_{u_i \in \mathcal{U}} \left(\eta_1 \sum_{t \in \mathcal{T}} \tau^t (1 + \log_{z_i^t}(num_j^*(\mathbf{a}^{OPT}))) w_i^t + \eta_2 \sigma^2 \right)}$$

Combining Case 1 and Case 2 completes the proof. \square

APPENDIX H POWER ALLOCATION

After the user allocation is finished, miUA allocates transmit power to users to ensure their required data rate. The power allocation problem formulation can be found in Section 5.2.

H.1 Power Allocation Problem Transformation

Instead of allocating transmit power to each user individually, the power allocation problem (16) can be converted into a problem of finding the total transmit power of all the users on a channel (or in other words, the transmit power allocated to a channel). After that, the power allocated to each channel will be allocated to the users on that channel. The rationale behind this is that the transmit power required by a user is determined by the total transmit power of users allocated to its neighbor edge servers on the same channel, apart from the transmit power of users sharing the channel in the same server. Once the total transmit power of the users allocated to the neighbor edge servers on the same channel is found, we can find the transmit power for each individual users. Lemma 2 below defines the minimum transmit power required by a channel to satisfy the data rate requirement of the users allocated to that channel. We use $\mathbf{p}_{sc} = \{p_j^k | c_j^k \in \mathcal{C}_j, s_j \in \mathcal{S}\}$ to denote the channel transmit power allocation strategy.

Lemma 2. Given \mathcal{U}_j^k – the set of users allocated to channel c_j^k on edge server s_j , to ensure the required data rate r_{min} for all those users, the total transmit power p_j^k of those users, i.e., the transmit power allocated to channel c_j^k , must satisfy:

$$p_j^k \geq \sum_{i=1}^{|\mathcal{U}_j^k|} \left(2^{\frac{r_{min}}{B_j^k}} - 1\right) \left(2^{\frac{r_{min}}{B_j^k}} - 1\right)^{i-1} H_{j,i}^k \triangleq y_j^k(\mathbf{p}_{sc})$$

Proof: Please refer to Appendix I. \square

According to [35], [41], the optimal solution to finding the total transmit power of all the users allocated to a channel can be found by solving the following problem, which is transformed from problem (16).

$$\min_{\{p_j^k\}} \sum_{j=1}^M \sum_{k=1}^V p_j^k \quad (33a)$$

$$\text{s.t. } p_j^k \geq \sum_{i=1}^{|\mathcal{U}_j^k|} \left(2^{\frac{r_{min}}{B_j^k}} - 1\right) \left(2^{\frac{r_{min}}{B_j^k}} - 1\right)^{i-1} H_{j,i}^k \triangleq y_j^k(\mathbf{p}_{sc}), \quad \forall c_j^k \in \mathcal{C}_j, \forall s_j \in \mathcal{S} \quad (33b)$$

$$\Theta(c_j^k), \forall s_j \in \mathcal{S} \quad (33c)$$

$$\sum_{k=1}^V p_j^k \leq P_j, \forall s_j \in \mathcal{S} \quad (33d)$$

$$p_j^k \in \mathbb{R}_{\geq 0}, \forall c_j^k \in \mathcal{C}_j, \forall s_j \in \mathcal{S} \quad (33e)$$

where objective (33a) minimizes the total transmit power allocated to all the channels in all the edge servers. Constraint (33b) is retrieved from Lemma 2, which helps enforce constraint (13e). Constraint (33c) enforces the SIC decoding order on each channel. Constraint (33d) ensures that the total transmit power allocated to all the channels in an edge server does not exceed that edge server's power capacity. Constraint (33e) indicates the possible values of channel transmit power decision p_j^k .

H.2 Decentralized Power Allocation Algorithm

Similar to Algorithm 1, the power allocation algorithm below (Algorithm 2, adopted in [35], [40]) is also decentralized.

This is a two-stage algorithm. First, the BS's transmit power is allocated to each channel (Lines 1-11). After that, the transmit power allocated to each channel will be allocated to the users on that channel (Lines 12-19). In the first stage, initially, the transmit power of a base station is equally assigned to all channels in that base station (Lines 2-4). This is followed by an iterative and recursive process for updating the transmit power allocated to channels, which consists of a finite number of iterations (Lines 5-11). In each iteration, the transmit power allocated to each channel is updated based on the transmit power allocated to other channels, which may have been updated in the previous iteration (Line 8). Similar to [35], $y_j^k(\mathbf{p}_{sc})$ is a *standard interference function* [41] since it satisfies three criteria as follows: 1) Positivity: $y_j^k(\mathbf{p}_{sc}) > 0$, and 2) Monotocity: If $\mathbf{p}_{sc} > \mathbf{p}'_{sc}$ then $y_j^k(\mathbf{p}_{sc}) > y_j^k(\mathbf{p}'_{sc})$, and 3) Scalability: For all $\alpha > 1$, then $\alpha y_j^k(\mathbf{p}_{sc}) > y_j^k(\alpha \mathbf{p}_{sc})$. When $y_j^k(\mathbf{p}_{sc})$ is standard, Algorithm 2 is referred to as a *standard power control algorithm*, which will eventually converge to a unique

18

Algorithm 2 Decentralized power allocation algorithm

Input: $\mathcal{S}, \mathcal{U}, \mathbf{a}$, and other parameters

Output: power allocation strategy \mathbf{p}

```

1: Stage 1: allocating BS's transmit power to channels
2: initialization:
3:  $p_j^k = P_j/V, \forall s_j \in \mathcal{S}, \forall c_j^k \in \mathcal{C}_j$ 
4: end initialization
5: repeat
6:   for each server  $s_j \in \mathcal{S}$  do
7:     for each channel  $c_j^k \in \mathcal{C}_j$  in server  $s_j$  do
8:       Calculate  $p_j^k(\textit{iteration}) = y_j^k(\mathbf{p}_{sc}^{(\textit{iteration}-1)})$ 
9:     end for
10:  end for
11: until convergence
12: Stage 2: allocating channel's transmit power to users
13: for each server  $s_j \in \mathcal{S}$  do
14:   for each server  $s_j$ 's channel  $c_j^k \in \mathcal{C}_j$  do
15:     for  $u_i \in \mathcal{U}_j^k$  do  $\triangleright$  start from the weakest user
16:       Calculate  $\mathbf{p}_{j,i}^k = (2^{\frac{r_{min}}{B_j^k}} - 1) (\sum_{q=1}^{i-1} \mathbf{p}_{j,q}^k + H_{j,i}^k)$ 
17:     end for
18:   end for
19: end for
    
```

fixed point (the global optimal solution, if one exists) from any initial power allocation [35], [40], [41].

In the second stage, the transmit power allocated to channels will be allocated to the users on these channels (Lines 13-19). On each channel, the transmit power is allocated in the order of channel conditions, or the SIC decoding order (8). The user with the weakest channel condition is the first to be allocated transmit power using (9), where $r_{j,i}^k = r_{min}$ (Line 16). Transmit power is allocated to that user first because it is the first to decode the received signal without the need for SIC or the consideration of the power of the other users sharing the same channel.

Complexity Analysis. Similar to Algorithm 1, Algorithm 2 can also be executed in parallel since the calculation of $p_j^k(\textit{iteration})$ for each channel of each edge server (Line 8) is independent of the calculations for all the other channels. The worst-case time complexity of calculating $p_j^k(\textit{iteration})$ is $\mathcal{O}(N_c M_u)$, where N_c is the highest number of users allocated to a channel and M_u is the highest number of neighbor edge servers of a user. In practice, N_c and M_u are relatively small. According to [35], this algorithm usually converges in 20 iterations.

APPENDIX I PROOF OF LEMMA 2

Proof: Based on (6), and (13e), the minimum user data rate requirement for serving a user u_i , who is allocated to channel c_j^k on edge server s_j , can be expressed by:

$$r_{j,i}^k = B_j^k \log_2 \left(1 + \frac{\mathbf{p}_{j,i}^k}{\sum_{t=i+1}^{|\mathcal{U}_j^k|} \mathbf{p}_{j,t}^k + H_{j,i}^k} \right) \geq r_{min}$$

Therefore, to reach the minimum user data rate r_{min} , the minimum transmit power $\mathbf{p}_{j,i}^k$ allocated to user u_i must

satisfy:

$$\mathbf{p}_{j,i}^k = (2^{\frac{r_{\min}^k}{B_j^k}} - 1) \left(\sum_{t=i+1}^{|\mathcal{U}_j^k|} \mathbf{p}_{j,t}^k + H_{j,i}^k \right)$$

Next, we find the total transmit power allocated to all the users on a channel so that they all can receive the minimum user data rate. First, let us consider the case where channel c_j^k has two users u_1 and u_2 . Assume that u_1 has the weakest channel condition and u_2 has the strongest channel condition. By following SIC decoding order constraint (13f), the transmit power allocated to users u_1 , and u_2 is:

$$\mathbf{p}_{j,2}^k = (2^{\frac{r_{\min}^k}{B_j^k}} - 1) H_{j,2}^k, \quad \mathbf{p}_{j,1}^k = (2^{\frac{r_{\min}^k}{B_j^k}} - 1) (\mathbf{p}_{j,2}^k + H_{j,1}^k)$$

The total transmit power allocated to all the users on channel c_j^k is then:

$$p_j^k = \mathbf{p}_{j,1}^k + \mathbf{p}_{j,2}^k = H_{j,1}^k (2^{\frac{r_{\min}^k}{B_j^k}} - 1) + H_{j,2}^k \left((2^{\frac{r_{\min}^k}{B_j^k}} - 1) + (2^{\frac{r_{\min}^k}{B_j^k}} - 1)^2 \right)$$

When channel c_j^k has three users u_1 , u_2 , and u_3 , where u_1 has the weakest channel condition and u_3 has the strongest channel condition, the transmit power allocated to u_1 , u_2 , and u_3 is then:

$$\begin{aligned} \mathbf{p}_{j,3}^k &= (2^{\frac{r_{\min}^k}{B_j^k}} - 1) H_{j,3}^k, \quad \mathbf{p}_{j,2}^k = (2^{\frac{r_{\min}^k}{B_j^k}} - 1) (\mathbf{p}_{j,3}^k + H_{j,2}^k) \\ \mathbf{p}_{j,1}^k &= (2^{\frac{r_{\min}^k}{B_j^k}} - 1) (\mathbf{p}_{j,2}^k + \mathbf{p}_{j,3}^k + H_{j,1}^k) \end{aligned}$$

The total transmit power allocated to all the users on channel c_j^k is then:

$$\begin{aligned} p_j^k &= \mathbf{p}_{j,1}^k + \mathbf{p}_{j,2}^k + \mathbf{p}_{j,3}^k \\ &= H_{j,1}^k (2^{\frac{r_{\min}^k}{B_j^k}} - 1) + H_{j,2}^k \left((2^{\frac{r_{\min}^k}{B_j^k}} - 1) + (2^{\frac{r_{\min}^k}{B_j^k}} - 1)^2 \right) \\ &\quad + H_{j,3}^k \left((2^{\frac{r_{\min}^k}{B_j^k}} - 1) + 2(2^{\frac{r_{\min}^k}{B_j^k}} - 1)^2 + (2^{\frac{r_{\min}^k}{B_j^k}} - 1)^3 \right) \end{aligned}$$

Finally, using mathematical induction, the minimum total transmit power allocated to all the users on a general channel c_j^k is:

$$p_j^k = \sum_{i=1}^{|\mathcal{U}_j^k|} (2^{\frac{r_{\min}^k}{B_j^k}} - 1) (2^{\frac{r_{\min}^k}{B_j^k}} - 1)^{i-1} H_{j,i}^k$$

□

5.2 A Dynamic Scenario

In this section, we consider the scenario where users come and go randomly over time. We aim to maximize the allocation delay and intra-cell/inter-cell interference, inherently minimizing the transmit power and maximizing the energy efficiency of the MEC system, while keeping user data rates at a stable and satisfactory level. We adopt Lyapunov optimization to transform this long-term problem into a series of subproblems to be solved in individual time slots. To effectively and efficiently solve the subproblem mentioned above within each time slot, we model it as a potential game then introduce a decentralized user and power allocation algorithm to find Nash equilibria, utilizing the distributed nature of MEC. We theoretically and experimentally evaluate the proposed approach and show it to significantly outperform a baseline and three state-of-the-art approaches.

This section is presented in the form of a paper [76] that is under review by IEEE International Conference on Computer Communications (INFOCOM).

Online User and Power Allocation for Dynamic NOMA-based Mobile Edge Computing

Anonymous Author(s)

Abstract—In this work, we study online user allocation in mobile edge computing (MEC) networks powered by non-orthogonal multiple access. App vendors need to determine a proper wireless channel in a base station/edge server and sufficient transmit power for every user. We consider a stochastic MEC system where users arrive and depart over time. When an edge server runs out of computing resources, some users will have to wait until the resources become available again. This incurs an allocation delay cost. We aim to minimize the allocation delay and transmit power costs, increasing the energy efficiency of the MEC system. To achieve this objective while guaranteeing users' data rate requirements, we adopt the Lyapunov framework to transform this long-term optimization problem into a series of subproblems to be solved in every time slot. Unlike typical Lyapunov optimization frameworks designed to stabilize queuing systems, we aim to stabilize users' data rates over time. To solve the aforementioned subproblems efficiently, we present a distributed game theory-based approach. The proposed algorithm is theoretically and experimentally evaluated, and is demonstrated to outperform several baseline and state-of-the-art methods, highlighting the significance of systematic consideration for both computation and communication aspects of this problem.

I. INTRODUCTION

Mobile edge computing (MEC) has fueled the potential of latency-sensitive applications [1], [2] as edge servers can be installed at cellular base stations (BSs) in close distance to users. App vendors can rent computing resources in edge servers and host their services for their users to access. To facilitate the massive connectivity over 5G/6G networks, non-orthogonal multiple access (NOMA) is proposed [3]. Compared with traditional multi-access methods for wireless communication (e.g., OFDMA, TDMA, or CDMA), NOMA achieves greater spectral efficiency and user throughput performance by accommodating multiple users concurrently with the same frequency or time resources in the power or code domain [3]. Integrating NOMA into MEC systems will further promote latency-sensitive applications in the 5G/6G era.

The *edge user allocation (EUA)* problem has been investigated extensively in recent years as an offline problem [4]–[8]. MEC researchers have begun to study computation offloading with NOMA. However, the EUA problem in NOMA-based MEC still remains open. Here, we study an *online* EUA problem in downlink multi-channel multi-cell power-domain NOMA-based MEC systems. In power-domain NOMA, a frequency channel can serve multiple users simultaneously. An app vendor needs to select a suitable channel in a suitable

BS/edge server¹ with a sufficient amount of transmit power to serve each user and satisfy its data rate requirement. We study a highly stochastic time-slotted MEC system. In every time slot, there is a random number of user arrivals and departures.

When allocating users, app vendors have to incorporate two types of costs. Firstly, due to the heterogeneity and limitation of edge servers' computing resources [9], new users might have to wait until existing users depart the system and free up the occupied computing resources in edge servers. This incurs an *allocation delay cost*. Secondly, the *transmit power cost* must be minimized. With the above in mind, a minimum data rate requirement must be fulfilled for as many users as possible. In NOMA, the transmit powers allocated to different users are tightly coupled and must be considered in conjunction with each other. Two decisions are jointly made for each user: 1) user allocation, including BS/edge server and channel assignments; and 2) power allocation, so that the allocation delay and transmit power costs are minimized. Our key contributions include:

- We model the online EUA problem. To our best knowledge, this is the first study on online user and power allocation in stochastic multi-channel multi-cell NOMA-based MEC. We adopt Lyapunov optimization to transform this long-term problem into a series of subproblems to be solved in individual time slots. Unlike typical adoptions of Lyapunov optimization that model target systems as queuing systems, our approach aims to stabilize users' data rates over time.
- To effectively and efficiently solve the subproblem mentioned above within each time slot, we model it as a potential game then introduce a decentralized user and power allocation algorithm to find Nash equilibria, utilizing the distributed nature of MEC.
- We theoretically and experimentally evaluate the proposed approach and show it to significantly outperform various baseline and state-of-the-art approaches.

The rest of this paper is structured as follows. Relevant studies are reviewed in Section II. In Section III, we model the stochastic MEC system and formulate the problem. In Section IV, a Lyapunov optimization-based online user allocation method is presented. As part of this online algorithm, we introduce a game-theoretical approach in Section V. Our approach is experimentally evaluated in Section VI. Lastly, the paper is concluded in Section VII.

¹The terms "edge server" and "base station" (BS) will be used interchangeably.

II. RELATED WORK

This section has been removed to reduce repetition in this thesis. Please refer to the original paper online for a full, unedited version.

III. SYSTEM MODEL

A. System Description

Edge servers: An MEC system consists of M BSs denoted by $S = \{s_1, \dots, s_M\}$. The cell radius of BS $s_j \in S$ is rad_j . The set of K channels in each BS s_j is denoted by $C_j = \{c_j^1, \dots, c_j^K\}$. We divide the total bandwidth B of each BS s_j equally into all channels C_j . Each channel $c_j^k \in C_j$ has a bandwidth $B_j^k = B/K$. Each BS s_j has an edge server installed, whose computing capacity is denoted by an $|\mathcal{R}|$ -dimensional vector $R_j = (R_j^r)$ with dimension R_j^r being the capacity of resource type $r \in \mathcal{R} = \{\text{CPU, memory, storage, \dots}\}$.

Mobile users: This system's operational timeline is represented by a series of equal-length time slots t . Let $\mathcal{U}(t)$ be the set of newly-arrived users u_i in time slot t , and \mathcal{U} be the set of all the current users in the system. We use an $|\mathcal{R}|$ -dimensional vector $w_i = (w_i^r)$, $r \in \mathcal{R}$, to denote the required amount of computing resources to accommodate user u_i in an edge server. The distance from user u_i to BS s_j is $d_{j,i}$. We use $S_i = \{s_j \in S | d_{j,i} \leq rad_j\}$, $\forall u_i \in \mathcal{U}(t)$, to denote the set of user u_i 's neighbor BSs, i.e., BSs that cover user u_i . A user's session (the time a user uses the application, or is served by an edge server) is unknown at any time and represented by a number of time slots. For every user u_i , we need to make two decisions as follows.

User Allocation Decision. $\mathbf{a}_{j,i}^k(t) = \{0, 1\}$ denotes the decision variable for user u_i in time slot t . $\mathbf{a}_{j,i}^k(t) = 1$ if user u_i is assigned to BS s_j on channel c_j^k in time slot t ; otherwise $\mathbf{a}_{j,i}^k(t) = 0$. Let $\mathbf{a}(t) = \{\mathbf{a}_i(t) | u_i \in \mathcal{U}(t)\}$ represent the user allocation strategy comprised of the decisions for all the users in time slot t . $\mathbf{a}_i(t) \triangleq (s_j, c_j^k)$, where $\mathbf{a}_{j,i}^k(t) = 1$, indicates the BS and channel that serve user u_i in time slot t . We let $\mathbf{a}_i(t) \triangleq (0, 0)$ when user u_i is unallocated.

Power Allocation Decision. $\mathbf{p}_i(t)$ indicates user u_i 's allocated transmit power in time slot t . Let $\mathbf{p}(t) = \{\mathbf{p}_i(t) | u_i \in \mathcal{U}\}$ represent the power allocation strategy comprised of the power allocation decisions for all the current users in the system in time slot t .

Let $\mathcal{U}_j(t) = \{u_i \in \mathcal{U} | \sum_{k=1}^K \mathbf{a}_{j,i}^k(t) = 1\}$, be the set of users allocated to BS s_j in time slot t , and $\mathcal{U}_j^k(t) = \{u_i \in \mathcal{U} | \mathbf{a}_{j,i}^k(t) = 1\}$, be the set of users allocated to channel c_j^k in BS s_j in time slot t , $\mathcal{U}_j^k(t) \subseteq \mathcal{U}_j(t)$. A user cannot be allocated to multiple channels or BSs in a time slot.

B. Allocation Delay Model

When user u_i is assigned to server s_j that has insufficient computing resources, u_i will be put in a waiting list Q_j waiting to be served. Once one or more existing users have left and free up the occupied resources, server s_j can start serving the users in the waiting list on a first-come, first-served basis. Given edge server s_j 's computing capacity, we can easily calculate

N_j , the maximum number of simultaneous users it can serve in a time slot. We use ℓ to represent the expected length of a user session, which can be approximated based on historical data in practice. Edge server s_j 's service rate can then be calculated by N_j/ℓ . Let $n_j(t)$ be the number of users being served by server s_j and $Q_j(t)$ be the length of server s_j 's waiting list in time slot t . The allocation delay cost of a newly-arrived user u_i in server s_j , or the number of time slots that u_i has to wait until being served by s_j , can be estimated by:

$$M_i(\mathbf{a}(t)) = \begin{cases} \frac{[n_j(t) - N_j + Q_j(t) + 1]_+}{N_j/\ell} & \text{if } \sum_{c_j^k \in C_j} \mathbf{a}_{j,i}^k(t) = 1 \\ M_{max} & \text{if } \sum_{s_j \in S} \sum_{c_j^k \in C_j} \mathbf{a}_{j,i}^k(t) = 0 \end{cases} \quad (1)$$

where M_{max} is the penalty when user u_i is unallocated, which can be any arbitrarily large number. An unallocated user always incurs a greater allocation delay cost than an allocated user to drive app vendors into allocating users.

C. Interference Model

1) Signal Model: Based on NOMA scheme, a BS transmits a superposition-coded signal to everyone on a channel [3]. In downlink transmissions, users employ SIC to decode the received superposed signal. Without loss of generality, suppose that all users $\mathcal{U}_j^k(t)$ on channel c_j^k are sorted by their channel conditions: $u_1, u_2, \dots, u_{|\mathcal{U}_j^k(t)|}$, where u_1 has the weakest channel condition and $u_{|\mathcal{U}_j^k(t)|}$ has the strongest channel condition. User u_1 , being the weakest user in $\mathcal{U}_j^k(t)$, decodes the signal without performing SIC. User u_1 's decoded component is then subtracted from the superposed signal. The subsequent user in $\mathcal{U}_j^k(t)$, i.e., user u_2 , can decode the received signal without interference from user u_1 . Following this principle, the signal received by user $u_i \in \mathcal{U}_j^k(t)$ on channel c_j^k in BS s_j in time slot t has a signal-to-interference-plus-noise ratio $\gamma_i(t)$ of:

$$\gamma_i(t) = \frac{|h_{j,i}^k|^2 \mathbf{p}_i(t)}{|h_{j,i}^k|^2 \sum_{q=i+1}^{|\mathcal{U}_j^k(t)|} \mathbf{p}_q(t) + I_{j,i}^k(t) + \sigma^2} \quad (2)$$

where $|h_{j,i}^k|^2$ is user u_i 's channel gain on channel c_j^k , $|h_{j,i}^k|^2 \sum_{q=i+1}^{|\mathcal{U}_j^k(t)|} \mathbf{p}_q(t)$ is the intra-cell interference experienced by user u_i (caused by those sharing the same channel with user u_i), $I_{j,i}^k(t) = \sum_{s_l \in S_i \setminus \{s_j\}} |h_{l,i}^k|^2 p_l^k(t)$ is the inter-cell interference experienced by user u_i (caused by users in u_i 's neighbor BSs), and σ^2 is the additive white Gaussian noise. Taking into account the factors that affect a channel's condition, the SIC decoding order of users on channel c_j^k in time slot t must follow:

$$\Theta_j^k(t) \triangleq \frac{I_{j,1}^k(t) + \sigma^2}{|h_{j,1}^k|^2} \geq \dots \geq \frac{I_{j,|\mathcal{U}_j^k(t)|}^k(t) + \sigma^2}{|h_{j,|\mathcal{U}_j^k(t)|}^k|^2} \quad (3)$$

where weaker users (high inter-cell interference and low channel gain) decode before stronger users. According to [10], $\Theta_j^k(t)$ is optimal for efficiently improving each individual user's data rate. When $\Theta_j^k(t)$ is followed, user u_i 's achievable data rate $r_i(t)$ in time slot t is then: $r_i(t) = B_j^k \log_2(1 + \gamma_i(t))$.

2) *Interference Cost Model*: Given allocation strategies $\mathbf{a}(t)$ and $\mathbf{p}(t)$, the interference-plus-noise $I_i(\mathbf{a}(t), \mathbf{p}(t))$ experienced by user u_i can be measured by:

$$I_i(\mathbf{a}(t), \mathbf{p}(t)) = \begin{cases} |h_{j,i}^k|^2 \sum_{q=i+1}^{|\mathcal{U}_j^k(t)|} \mathbf{p}_q(t) + I_{j,i}^k(t) + \sigma^2 & \text{if } \sum_{c_j^k \in \mathcal{C}_j} \mathbf{a}_{j,i}^k(t) = 1 \\ I_{max} & \text{if } \sum_{s_j \in \mathcal{S}} \sum_{c_j^k \in \mathcal{C}_j} \mathbf{a}_{j,i}^k(t) = 0 \end{cases} \quad (4)$$

where I_{max} is the highest interference-plus-noise that a user would receive if allocated to a channel. I_{max} can also be any arbitrarily large number. It acts as a penalty for unallocated users to drive app vendors into allocating users, similar to how the allocation delay cost is formulated in Section III-B.

D. Problem Formulation

We formulate the online EUA problem as follows.

$$(\mathbf{P1}) \min \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left\{ \sum_{u_i \in \mathcal{U}} (\eta_1 M_i(\mathbf{a}(t)) + \eta_2 I_i(\mathbf{a}(t), \mathbf{p}(t))) \right\}$$

$$\text{s.t. } \sum_{i=1}^{|\mathcal{U}_j|} \sum_{r=1}^{|\mathcal{R}|} \sum_{k=1}^K \mathbf{a}_{j,i}^k(t) w_i^r \leq R_j^r, \forall s_j \in \mathcal{S}, \forall t \quad (5a)$$

$$\sum_{j=1}^M \sum_{k=1}^K \mathbf{a}_{j,i}^k(t) d_{j,i} \leq rad_j, \forall u_i \in \mathcal{U}, \forall t \quad (5b)$$

$$\Theta_j^k(t), \forall s_j \in \mathcal{S}, \forall c_j^k \in \mathcal{C}_j, \forall t \quad (5c)$$

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \{ r_i(t) \} \geq r_{req}, \forall u_i \in \mathcal{U} \quad (5d)$$

$$\sum_{j=1}^M \sum_{k=1}^K \mathbf{a}_{j,i}^k(t) = 1, \forall u_i \in \mathcal{U} \quad (5e)$$

$$\sum_{k=1}^K \sum_{i=1}^{|\mathcal{U}|} \mathbf{a}_{j,i}^k(t) \mathbf{p}_i(t) \leq P_j, \forall s_j \in \mathcal{S}, \forall t \quad (5f)$$

$$\mathbf{a}_{j,i}^k(t) \in \{0, 1\}, \forall s_j \in \mathcal{S}, \forall u_i \in \mathcal{U}, \forall c_j^k \in \mathcal{C}_j \quad (5g)$$

$$\mathbf{p}_i(t) \in \mathbb{R}_{\geq 0}, \forall u_i \in \mathcal{U}, \forall t \quad (5h)$$

where η_1 and η_2 ($\eta_1 + \eta_2 = 1$) are the weights that indicate the importance of allocation delay and interference costs. As the MEC system is highly stochastic, optimizing the long-term system performance is more beneficial than optimizing the short-term, spontaneous system performance. The objective is to minimize the time-average expectation of system cost, which consists of allocation delay costs and interference costs over multiple time slots. Constraints (5a) and (5b) ensure that an edge server/BS does not accommodate users outside its computing capacity and cell coverage, respectively. Constraint (5c) enforces the SIC decoding order (3). Constraint (5d) ensures a long-term minimum data rate requirement r_{req} for every user. Constraint (5e) makes sure that a user is not allocated to multiple channels or BSs in a time slot. Constraint

(5f) ensures that the total power assigned to all users in a BS does not exceed the BS's maximum power allowance at any time. Constraints (5g) and (5h) define the acceptable values of $\mathbf{a}_{j,i}^k(t)$ and $\mathbf{p}_i(t)$.

IV. ONLINE USER AND POWER ALLOCATION WITH LYAPUNOV OPTIMIZATION

We introduce a Lyapunov optimization-based algorithm to solve problem **P1**.

A. Problem Transformation with Lyapunov Optimization

To meet the long-term data rate requirement (5d), i.e., to stabilize users' average data rate over time, we introduce a concept called *accumulated data rate* for each user u_i , which is defined by:

$$D_i(t+1) = \max\{D_i(t) + r_{req} - r_i(t), 0\} \quad (6)$$

where $D_i(0) = 0, \forall u_i \in \mathcal{U}$. The accumulated data rate $D_i(t+1)$ of a user u_i represents its overdue data rate accumulated over t time slots relative to the data rate requirement r_{req} . Its value increases if the user's data rate in the previous time slot $r_i(t)$ decreases and vice versa. This can be used to adjust the user and power allocation strategies to stabilize users' average data rate over time as enforced by (5d). To fulfill the long-term data rate requirement, $D_i(t)$ must be stabilized, or *mean rate stable* [11]: $\lim_{t \rightarrow \infty} \frac{\mathbb{E}\{D_i(t)\}}{t} = 0$. Based on Eq. (6), we define a quadratic Lyapunov function: $L(D(t)) \triangleq \frac{1}{2} \sum_{u_i \in \mathcal{U}} D_i(t)^2$, where $D(t) \triangleq \{D_i(t), \forall u_i \in \mathcal{U}\}$. We can see that $L(D(t))$ is high when there is at least one user with a high accumulated data rate $D_i(t)$, and $L(D(t))$ is low when the accumulated data rate of every user is small, representing a stable state. We then define a *conditional Lyapunov drift* to observe how the Lyapunov function changes between two consecutive time slots: $\Delta(D(t)) \triangleq \mathbb{E}\{L(D(t+1)) - L(D(t)) | D(t)\}$. We minimize the system cost while stabilizing users' average data rate. By incorporating the system cost into the Lyapunov drift above, our optimization objective can be fulfilled without violating the data rate constraints. This can be achieved via a *drift-plus-penalty*:

$$\Delta(D(t)) + V \mathbb{E} \left\{ \sum_{u_i \in \mathcal{U}} (\eta_1 M_i(\mathbf{a}(t)) + \eta_2 I_i(\mathbf{a}(t), \mathbf{p}(t))) | D(t) \right\}$$

where $V > 0$ is a parameter which adjusts the relative importance of the system cost to the accumulated user data rate. Depending on the application context, app vendors can flexibly regulate the trade-off between time-average system cost and accumulated data rate by changing the value of V . For instance, they can increase V to relax the user data rate requirement and put more emphasis on minimizing the system cost. Under the Lyapunov optimization scheme, we pursue the optimization objective in **P1** by minimizing the supreme bound of the above drift-plus-penalty.

Lemma 1. *Given any user and power allocation strategy in any time slot, the drift-plus-penalty is bounded by:*

$$\begin{aligned} & \Delta(D(t)) + V \mathbb{E} \left\{ \sum_{u_i \in \mathcal{U}} (\eta_1 M_i(\mathbf{a}(t)) + \eta_2 I_i(\mathbf{a}(t), \mathbf{p}(t))) | D(t) \right\} \\ & \leq O + \sum_{u_i \in \mathcal{U}} \mathbb{E} \left\{ \frac{r_i^2(t)}{2} + D_i(t)(r_{req} - r_i(t)) - r_{req} r_i(t) \right. \\ & \quad \left. + V (\eta_1 M_i(\mathbf{a}(t)) + \eta_2 I_i(\mathbf{a}(t), \mathbf{p}(t))) | D(t) \right\} \end{aligned} \quad (7)$$

where $O = \frac{r_{req}^2}{2}$ is a finite constant.

Proof. See Appendix A. \square

Next, we propose OUAD (Algorithm 1), an Online User Allocation algorithm in Dynamic NOMA-based MEC systems, which formulates a user and power allocation strategy, $\mathbf{a}(t)$ and $\mathbf{p}(t)$, to lower the supreme bound of the drift-plus-penalty (7) in every time slot. Employing the concept of minimizing an expectation opportunistically [11], we can accomplish this by solving problem **P2** defined below.

$$\begin{aligned} (\mathbf{P2}) \min_{\mathbf{a}(t), \mathbf{p}(t)} & \sum_{u_i \in \mathcal{U}} \left(\frac{r_i^2(t)}{2} + D_i(t)(r_{req} - r_i(t)) - r_{req} r_i(t) \right. \\ & \left. + V (\eta_1 M_i(\mathbf{a}(t)) + \eta_2 I_i(\mathbf{a}(t), \mathbf{p}(t))) \right) \quad (8) \\ \text{s.t.} & \text{ (5a), (5b), (5c), (5e), (5f), (5g), (5h)} \end{aligned}$$

Algorithm 1 OUAD

- 1: **Input:** $S, V, \eta_1, \eta_2, r_{req}$
 - 2: **Output:** user and power allocation decisions $\mathbf{a}_{j,s}^k(t), \mathbf{p}_i(t), \forall t, \forall s_j \in S, \forall u_i \in \mathcal{U}$
 - 3: **for** every time slot t **do**
 - 4: Observe newly-arrived users $\mathcal{U}(t)$ and each current user's accumulated data rate $D_i(t), \forall u_i \in \mathcal{U}$
 - 5: Determine $\mathbf{a}(t), \mathbf{p}(t)$ by solving **P2**
 - 6: Update users' accumulated data rate $D_i(t+1), \forall u_i \in \mathcal{U}$, according to Eq. (6)
 - 7: Update $Q_j(t+1), \forall s_j \in S$
 - 8: **end for**
-

In every time slot, newly-arrived users are assigned to BSs/edge servers with sufficient transmit power (Line 5 of Algorithm 1) based on the observed user arrivals and users' accumulated data rates (Line 4). Users that are assigned to an exhausted edge server will be put on a waiting list for that edge server and wait until one or more existing users depart. A user leaving the system releases computing resources, which can then be used to accommodate the waiting users. Users' accumulated data rates and waiting lists for all the edge servers are updated after a user and power allocation strategy has been determined in each time slot (Lines 6-7). OUAD works without prior knowledge of future user arrivals, user departures, or statistics of the user distribution. This online algorithm allocates every user as soon as they arrive and thus

can accommodate user mobility. When a user moves out of its associated BS's cell coverage, we will treat it as a new user and reallocate it to another BS in the next time slot. Employing the same technique as Theorem 4.2 in [11], we can see that OUAD achieves an $[\mathcal{O}(1/V), \mathcal{O}(V)]$ trade-off between time-average system cost and accumulated data rate.

Problem **P2** is NP-hard because its subproblem is already NP-hard, which we will show later. It is hard to be solved optimally within a time slot. To find a near-optimal solution efficiently, we break it down into a user allocation problem (Stage #1 - Section IV-B) and a power allocation problem (Stage #2 - Section IV-C). We first assign users to channels in BSs in Stage #1. Here, we seek a solution with a low allocation delay cost that is most likely to result in low interference in overall. We use the words "most likely" because we have not properly allocated transmit power to users yet, and thus the interference has not been evaluated. Once all the users are allocated to BSs, we will execute Stage #2 to adjust their transmit power to meet their data rate requirements in an energy-efficient manner.

B. User Allocation Problem

In this phase, we allocate every user to a channel in a BS by solving problem **P3** modeled below.

$$\begin{aligned} (\mathbf{P3}) \min_{\mathbf{a}(t)} & \sum_{u_i \in \mathcal{U}} \left(\frac{r_i^2(t)}{2} + D_i(t)(r_{req} - r_i(t)) - r_{req} r_i(t) \right. \\ & \left. + V (\eta_1 M_i(\mathbf{a}(t)) + \eta_2 I_i(\mathbf{a}(t), \mathbf{p}(t))) \right) \quad (9) \\ \text{s.t.} & \text{ (5a), (5b), (5e), (5g)} \end{aligned}$$

Interference and transmit power are highly interdependent. An app vendor allocates transmit power to users based on the inter- and intra-cell interference they experience, which is partly caused by other users' transmit power. To handle this interdependence, we fix one decision (power allocation decision) while choosing the other (user allocation decision). At this stage, all the users are assigned a default transmit power. This allows us to estimate the interference experienced by users and incorporate it into optimization objective (9). This increases the possibility of low interference when a proper power allocation algorithm is applied later on. Problem **P3** is NP-hard because a special case of it is a reduction of the NP-complete PARTITION problem [12]. The proof employs the same technique used in Appendix B of [4] so it will be omitted here due to limited space.

C. Power Allocation Problem

All the users are now assigned to channels in BSs once problem **P3** is solved. Next, we adjust their transmit power by solving problem **P4**, which is modeled by:

$$\begin{aligned} (\mathbf{P4}) \min_{\mathbf{p}(t)} & \sum_{u_i \in \mathcal{U}} \left(\frac{r_i^2(t)}{2} + D_i(t)(r_{req} - r_i(t)) - r_{req} r_i(t) \right. \\ & \left. + V \eta_2 I_i(\mathbf{a}(t), \mathbf{p}(t)) \right) \quad (10) \end{aligned}$$

s.t. (5c), (5f), (5h)

$M_i(\mathbf{a}(t))$ is now excluded in the objective because power allocation decisions have no impact on users' allocation delays. The transmit power of users that arrived in previous time slots will also be adjusted because their received interference might change when there are new users arriving at the system.

V. USER AND POWER ALLOCATION GAME

To effectively solve problem **P2** in an efficient manner in individual time slots, we first model it as a potential game. To find Nash equilibria in this game, we propose a decentralized two-stage algorithm, where each stage is responsible for solving a subproblem defined above (**P3** and **P4**).

A. Game Formulation and Properties

In each time slot t , an app vendor pursues objective (8) by finding a suitable user and power allocation strategy $\mathbf{a}(t)$ and $\mathbf{p}(t)$. The decision for each individual user u_i is determined based on other users' decisions $\mathbf{a}_{-i}(t)$ and $\mathbf{p}_{-i}(t)$.

The EUA problem in a time slot is modelled as a game $Z = (\mathcal{U}(t), \{\mathcal{A}_i(t), \mathcal{P}_i(t)\}_{u_i \in \mathcal{U}(t)}, \{C(\mathbf{a}_i(t), \mathbf{p}_i(t))\}_{u_i \in \mathcal{U}(t)})$, where $\mathcal{U}(t)$ is the set of players (users) arriving in time slot t , $\mathcal{A}_i(t)$ and $\mathcal{P}_i(t)$ are the sets of possible user and power allocation decisions available to each user u_i , and $C(\mathbf{a}_i(t), \mathbf{p}_i(t))$ is the system cost incurred by decisions $\mathbf{a}_i(t)$ and $\mathbf{p}_i(t)$ made for user u_i , the lower the better:

$$C(\mathbf{a}_i(t), \mathbf{p}_i(t)) = \sum_{u_i \in \mathcal{U}} \left(\frac{r_i^2(t)}{2} + D_i(t)(r_{req} - r_i(t)) - r_{req}r_i(t) + V(\eta_1 M_i(\mathbf{a}(t)) + \eta_2 I_i(\mathbf{a}(t), \mathbf{p}(t))) \right) \quad (11)$$

Next, we show that Z has at least one Nash equilibrium – a stable state of Z where an app vendor cannot lower the system cost any further by unilaterally changing the decision for any single user. In a Nash equilibrium $\mathbf{a}^*(t), \mathbf{p}^*(t)$, the allocation decision made for a user is the best response to the allocation decisions made for all other users [13]. This makes sure that if a Nash equilibrium exists, the decisions for all the users will automatically self-organize into a Nash equilibrium in finite iterations. In each iteration, every user proactively responds to the decisions made for all the other users to further reduce the system cost. A potential game (defined below) always admits one or more Nash equilibria [14]. The presence of a Nash equilibrium in Z can be confirmed by showing that Z is a potential game.

Definition 1. (Ordinal Potential Game) An ordinal potential game is a game that has a potential function $\phi(\mathbf{a}_i(t), \mathbf{p}_i(t))$ satisfying $C(\mathbf{a}_i(t), \mathbf{p}_i(t)) > C(\mathbf{a}'_i(t), \mathbf{p}'_i(t)) \Leftrightarrow \phi(\mathbf{a}_i(t), \mathbf{p}_i(t)) > \phi(\mathbf{a}'_i(t), \mathbf{p}'_i(t))$, where $\mathbf{a}_i(t), \mathbf{a}'_i(t) \in \mathcal{A}_i(t)$, and $\mathbf{p}_i(t), \mathbf{p}'_i(t) \in \mathcal{P}_i(t)$.

Theorem 1. Z is an ordinal potential game that has a potential function $\phi(\mathbf{a}_i(t), \mathbf{p}_i(t))$ defined as:

$$\phi(\mathbf{a}_i(t), \mathbf{p}_i(t))$$

$$\begin{aligned} &= \sum_{u_i \in \mathcal{U}} \left(\frac{r_i^2(t)}{2} + D_i(t)(r_{req} - r_i(t)) - r_{req}r_i(t) \right) \mathbb{1}_{\mathbf{a}_i(t) \neq (0,0)} \\ &+ \sum_{u_i \in \mathcal{U}} V \ell \eta_1 \frac{[n_j(t) - N_j + Q_j(t) + 1]_+}{N_j} \mathbb{1}_{\mathbf{a}_i(t) \neq (0,0)} \\ &+ \sum_{u_i \in \mathcal{U}} V \eta_2 |h_{j,i}^k|^2 \sum_{q=i+1}^{|\mathcal{U}_j^k(t)|} \mathbf{p}_{j,q}^k \mathbb{1}_{\mathbf{a}_i(t) \neq (0,0)} \\ &+ \sum_{u_i \in \mathcal{U}} V \eta_2 \sum_{s_i \in \mathcal{S}_i \setminus \{s_j\}} |h_{l,i}^k|^2 \mathbf{p}_l^k \mathbb{1}_{\mathbf{a}_i(t) \neq (0,0)} \\ &+ \sum_{u_i \in \mathcal{U}} (\eta_1 M_{max} + \eta_2 I_{max}) \mathbb{1}_{\mathbf{a}_i(t) = (0,0)} \end{aligned} \quad (12)$$

where the zero-one indicator function $\mathbb{1}_{condition}$ returns 1 if the condition is true, and 0 if the condition is false.

Proof: See Appendix B. ■

B. Algorithm Design

To find Nash equilibria in game Z , we present a distributed algorithm that adopts *best-response dynamics* [13], an iterative evolutionary procedure. In each iteration, we determine an allocation decision for every user by finding the best response to the decisions applied to other users. This decentralized procedure can be executed in parallel on edge servers, which coordinate the game through messaging synchronization [9], [15]. This procedure always converge to a Nash equilibrium thanks to *Finite Improvement Property* [14]. Our approach consists of two stages: Stage #1 (Algorithm 2) for solving problem **P3**, and Stage #2 (Algorithm 3) for solving problem **P4**.

Stage #1 (Algorithm 2): Algorithm 2 allocates every user to a channel in a BS. At this stage, every user is assigned a temporary default transmit power. Once Algorithm 2 completes, Algorithm 3 will adjust their transmit power to meet their data rate requirements. In Algorithm 2, all users are initially unallocated (Line 1). Subsequently, allocation decisions are updated and applied for every user iteratively (Lines 2-16), lowering the system cost after every iteration until it cannot be lowered any further. In each iteration, we determine the best decision $\mathbf{a}'_i(t)$ for each user u_i by iterating over all the channels in all of its neighbor BSs and select the channel that would generate the lowest system cost $C(\mathbf{a}'_i(t), \mathbf{p}(t))$ if user u_i is to be allocated to it (Lines 3-10). If $C(\mathbf{a}'_i(t), \mathbf{p}(t))$ is not lower than the current system cost, there is no need to update user u_i 's current decision. If the new decision $\mathbf{a}'_i(t)$ leads to a lower system cost, user u_i 's current decision will be updated with $\mathbf{a}'_i(t)$. The request for applying $\mathbf{a}'_i(t)$ will be submitted for the opportunity to be officially applied (Lines 11-13). Among all the requests for decision applying, the one with the lowest system cost will be officially applied (Line 15). The user, whose request for decision update is selected, now has a new allocation decision. Note that the allocation strategy in an iteration is not final; it may be amended in following iterations if a new allocation decision for a user is found. Users assigned to an exhausted edge server will be put

on a waiting list until one or more existing users depart the system and release the occupied computing resources.

Algorithm 2 Stage #1: User Allocation

Input: $\mathcal{S}, \mathcal{U}(t)$, fixed power allocation strategy $\mathbf{p}(t)$
Output: user allocation strategy $\mathbf{a}(t)$

- 1: $\mathbf{a}_i(t) = (0, 0), \forall u_i \in \mathcal{U}(t)$
- 2: **repeat**
- 3: Compute current system cost $C(\mathbf{a}(t), \mathbf{p}(t))$
- 4: **for** each user $u_i \in \mathcal{U}(t)$ **do**
- 5: **for** each neighbor BS $s_j \in \mathcal{S}_i$ of user u_i **do**
- 6: **for** each channel $c_j^k \in \mathcal{C}_j$ in BS s_j **do**
- 7: Compute $C(\mathbf{a}'_i(t), \mathbf{p}(t))$ – the new cost if user u_i is to be assigned to c_j^k
- 8: **end for**
- 9: **end for**
- 10: Among all feasible decisions $\mathbf{a}'_i(t)$ above, find one that incurs the lowest cost $C(\mathbf{a}'_i(t), \mathbf{p}(t))$
- 11: **if** $C(\mathbf{a}'_i(t), \mathbf{p}(t)) < C(\mathbf{a}(t), \mathbf{p}(t))$ **then**
- 12: Request to apply $\mathbf{a}'_i(t)$
- 13: **end if**
- 14: **end for**
- 15: Among all requests for applying decision update, apply the one with the lowest $C(\mathbf{a}'_i(t), \mathbf{p}(t))$
- 16: **until** decision updates not required for any users
- 17: Execute Stage #2

The process of updating decisions for all users (Lines 4–14) can be executed in parallel because the processes for different users are independent of each other. The search for the best decision for each user (Lines 5–9) can also be parallelized. After all the users are allocated, we execute Stage #2 (Algorithm 3) to adjust their transmit powers.

Stage #2 (Algorithm 3): Given the user allocation strategy found in Stage #1, Algorithm 3 adjusts the transmit power for all users. We consider a discrete power control scheme [16]–[18], where the transmit power of a user is selected from a set \mathcal{L} of discrete power levels. Discrete power control enables a simpler transmitter design than continuous power control and significantly reduces overhead incurred by information exchange among network nodes [17]. Initially, every user is allocated the lowest power level (Line 1). After that, for each user in each iteration, we iterate through every possible power level to find the one that incurs the lowest system cost $C(\mathbf{a}(t), \mathbf{p}'_i(t))$ (Lines 5–8). If $C(\mathbf{a}(t), \mathbf{p}'_i(t))$ is lower than the system cost incurred by the power allocation strategy found in the last iteration (calculated in Line 3), the request for updating this user's power will be submitted for a chance to be applied (Lines 9–11). Once all the users' requests for updating transmit power are submitted, the request with the lowest system cost will be officially applied in this iteration (Line 13). This iterative process terminates when we cannot update any user's transmit power to lower the system cost.

Algorithm 3 Stage #2: Power Allocation

Input: \mathcal{S}, \mathcal{U} , user allocation strategy $\mathbf{a}(t)$ found in Stage #1, a set \mathcal{L} of discrete power levels
Output: power allocation strategy $\mathbf{p}(t)$

- 1: Every user $u_i \in \mathcal{U}(t)$ is allocated the lowest power level
- 2: **repeat**
- 3: Compute current system cost $C(\mathbf{a}(t), \mathbf{p}(t))$
- 4: **for** each user $u_i \in \mathcal{U}$ **do**
- 5: **for** each power level $l \in \mathcal{L}$ **do**
- 6: Compute $C(\mathbf{a}(t), \mathbf{p}'_i(t))$ – the new cost if user u_i is given power level l
- 7: **end for**
- 8: Among all possible decisions $\mathbf{p}'_i(t)$ above, find one that incurs the lowest $C(\mathbf{a}(t), \mathbf{p}'_i(t))$
- 9: **if** $C(\mathbf{a}(t), \mathbf{p}'_i(t)) < C(\mathbf{a}(t), \mathbf{p}(t))$ **then**
- 10: Request to apply $\mathbf{p}'_i(t)$
- 11: **end if**
- 12: **end for**
- 13: Among all requests for applying decision update, apply the one that has lowest $C(\mathbf{a}(t), \mathbf{p}'_i(t))$
- 14: **until** decision updates not required for any users

C. Performance Analysis

We analyze the convergence time of Algorithms 2 and 3 by the number of iterations they take to reach a Nash equilibrium.

Theorem 2. *The convergence time of Algorithms 2 and 3 is upper bounded by:*

$$\frac{|\mathcal{U}|(\eta_1 M_{max} + \eta_2 I_{max})}{\frac{\sigma^2}{2} + D_i(t)(r_{req} - \tau) - r_{req}\tau + V\eta_2|h_{min}|^2 p} \quad (13)$$

where $|h_{min}|^2$ is the lowest possible channel gain of a user, p is the default transmit power in Algorithm 2 or the difference between two consecutive transmit power levels in Algorithm 3, and τ is the minimum change in data rate when an allocation decision is updated.

Proof: The proof is routine and similar to the proof of Theorem 2 in [4]. It is thus omitted due to limited space. ■

We then analyze the theoretical optimality of the solutions found by Algorithms 2 and 3 by examining the Price of Anarchy (PoA) in system cost. PoA, being the ratio of the the worst Nash equilibrium to the theoretical optimal strategy [14], is an important optimality indicator for game theory-based methods [9], [15]. We use $(\mathbf{a}^{opt}(t), \mathbf{p}^{opt}(t))$ to denote the optimal strategy. The system-cost PoA is then defined as $\frac{\max_{\mathbf{a}(t) \in \mathcal{A}(t), \mathbf{p}(t) \in \mathcal{A}(t)} C(\mathbf{a}(t), \mathbf{p}(t))}{C(\mathbf{a}^{opt}(t), \mathbf{p}^{opt}(t))}$.

Theorem 3. *The system cost PoA in game Z satisfies:*

$$1 \leq PoA \leq \frac{\sum_{u_i \in \mathcal{U}} (\eta_1 M_{max} + \eta_2 I_{max})}{\sum_{u_i \in \mathcal{U}} \left(\frac{r_{min}^2}{2} + D_{max}(r_{req} - r_{max}) - r_{req}r_{max} + V\eta_2\sigma^2 \right)} \quad (14)$$

where r_{min} and r_{max} are the possible minimum and maximum data rates of a user ($r_{min} < r_{req} < r_{max}$), D_{max} is the possible maximum accumulated data rate of a user.

Proof: The proof is routine and similar to the proof of Theorem 4 in [4] so it is omitted due to limited space. ■

VI. PERFORMANCE EVALUATION

A. Performance Benchmark

We evaluate OUAD against four representative approaches:

- *SUAC* [19]: This online user allocation approach minimizes the allocation delay cost in a time-slotted setting. It does not incorporate a proper power control scheme. Thus, to ensure the fairness of the comparison, after all users are allocated to BSs, we execute DPC-SPM, a state-of-the-art power control algorithm in NOMA [20], to assign minimum transmit power to users while meeting their data rate requirements.
- *SCG-SA* [21]: This approach aims to increase the energy efficiency while meeting user data rate requirements in NOMA-based cellular networks. SCG-SA only allocates users, who are presumably already allocated to BSs, to channels. It does not allocate users to BSs. Thus, we will allocate users to their nearest BSs. Subsequently, SCG-SA allocates users to channels based on a ranking of user channel conditions. This approach is designed without the consideration of time-slotted scenarios. In the experiments, we execute this approach once in every time slot. Unallocated users in a time slot are considered as new users in the next time slot.
- *miUA* [4]: This approach incorporates both inter- and intra-cell interference in NOMA-based MEC systems and also aims to maximize the energy efficiency. Similar to SCG-SA, miUA is designed without the consideration of time-slotted scenarios so we will execute this approach in every time slot. Unallocated users in a time slot are considered as new users in the next time slot.
- *Join Shortest Queue (JSQ)*: In each time slot, each user is assigned to the neighbor BS/server which has the shortest waiting list.

B. Experiment Settings

Our experiments are compliant with the LTE specifications [22] (Table I). We consider a 7-cell hexagon-layout network, corresponding to 7 BSs/edge servers, each with 6 communication channels. Each user requires four types of computing resources, $\mathcal{R} = \{\text{CPU, storage, RAM, GPU}\}$. We randomly generate edge servers' computing capacities using a normal distribution $\mathcal{N}(80, 20^2)$, where 80 is the average amount of each computing resource, and 20 is the standard deviation. All edge servers combined can serve $N = \sum_{s_j \in \mathcal{S}} N_j$ concurrent users. User arrivals are generated based on a Poisson process at rate $[0, \zeta N]$, where $\zeta \in \mathbb{R}$ controls the traffic intensity. They are uniformly distributed within BSs' coverage areas. w_i is randomly picked from three normalized levels $\{\langle 2, 2, 3, 3 \rangle, \langle 2, 2, 2, 1 \rangle, \langle 1, 1, 1, 2 \rangle\}$. The set \mathcal{L} of discrete transmit power levels is set to $\{-30\text{dBm}, -29\text{dBm}, \dots, 23\text{dBm}\}$. Each user session's length is uniformly drawn from 10 to 20 1-second time slots. We conduct three sets of experiments as summarized in Table II.

TABLE I: Experiment settings

BS maximum transmit power (P_j)	46dBm
Inter-site distance	500m
Cell radius (R_j)	289m
Minimum distance between user & BS	35m
Thermal noise density	-174dBm/Hz
Large-scale propagation model	$128.1 + 37.6 \log_{10}(d_{j,i})$ dB
System bandwidth (B)	10MHz

TABLE II: Experiment sets

	Traffic intensity ζ	Data rate requirement r_{req}	Control parameter V
Set #1	0.04, 0.041, ..., 0.049	0.5	5
Set #2	0.045	0.2, 0.3, ..., 0.7	5
Set #3	0.045	0.3, 0.5, 0.7	1, 2, ..., 10

C. Experiment Results

1) *Impact of Traffic Intensity (Set #1)*: In Set #1, we simulate different user arrival rates by varying the traffic intensity ζ . When ζ increases, the allocation delay experienced by a user on average (Fig. 1) gradually increases due to the rising number of users joining the system (for reference, there are around 20 new users in each time slot when $\zeta = 0.049$). SUAC, whose sole objective is to minimize the allocation delay, clearly achieves the lowest allocation delay among all the approaches, closely followed by JSQ and OUAD. miUA achieves the worst performance, incurring an average allocation delay twice higher than OUAD.

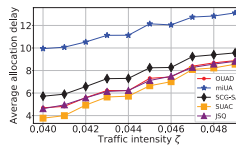


Fig. 1: Allocation delay vs. traffic intensity ζ (Set #1).

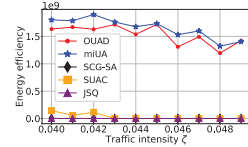


Fig. 2: Energy efficiency vs. traffic intensity ζ (Set #1).

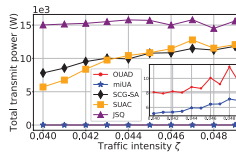


Fig. 3: Total transmit power vs. traffic intensity ζ (Set #1).

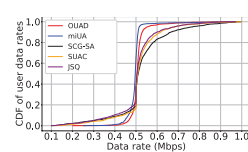


Fig. 4: CDF of all users' data rates (Set #1, $V = 0.045$).

The energy efficiency (Fig. 2) is measured by the ratio of the total data rate to the total power consumption. miUA is the most energy-efficient method since it solely focuses on minimizing inter- and intra-cell interference. However, its energy efficiency comes at the price of very long allocation delays (Fig. 1). OUAD achieves a much lower allocation delay while its energy efficiency is only slightly lower than miUA (even on par with miUA in some cases). OUAD and

miUA are remarkably more energy-efficient than the other three approaches. This shows the significance of considering interference in user allocation. Fig. 3 depicts the total transmit power required by all the approaches. We can see that SCG-SA, SUAC, and JSQ consume more power than OUAD and miUA by orders of magnitude. Fig. 4 illustrates the cumulative distribution function (CDF) of users' average data rate. A great portion of users allocated by SCG-SA, SUAC, and JSQ achieves either very low or very high data rates, largely deviating from the target data rate $r_{req} = 0.5$ Mbps. The average data rate of users allocated by OUAD is slightly higher than those allocated by miUA.

Fig. 5 visualizes the time efficiency under different traffic intensities. The line plot shows the average elapsed CPU time per time slot. The bar plot shows the number of decision iterations taken by Algorithms 2 and 3, which is a commonly used efficiency metric for game-theoretic approaches [9], [23] because of its machine independence (the time taken to solve a problem varies machine to machine). When traffic intensity ζ increases, Algorithms 2 and 3 require more iterations, consequently higher CPU time in total. OUAD is slightly faster than miUA. OUAD and miUA are the slowest due to the complexity of calculating user interference. Nevertheless, their completion time is still well within an acceptable range. In each time slot, OUAD allocates all new users within around 30ms – well below the duration of each time slot (1 second). This ensures that OUAD can be practically applied in MEC systems where low latency is mandated.

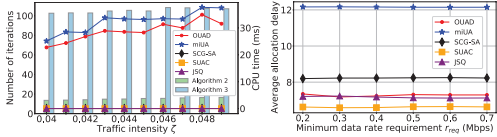


Fig. 5: Time efficiency vs. traffic intensity ζ (Set #1).

Fig. 6: Average allocation delay vs. data rate requirement r_{req} (Set #2).

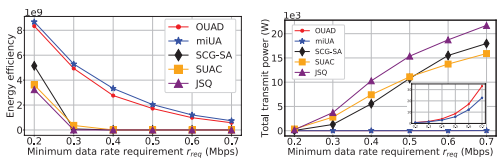


Fig. 7: Energy efficiency vs. data rate requirement r_{req} (Set #2).

Fig. 8: Total transmit power vs. data rate requirement r_{req} (Set #2).

2) *Impact of Minimum Data Rate Requirement (Set #2):* In Set #2, we vary the minimum data rate requirement r_{req} . Unsurprisingly, this does not have any impact on the allocation delay as shown in Fig. 6, which remains unchanged regardless of the changing r_{req} . Again, OUAD achieves a very low allocation delay, only marginally higher than SUAC, whose

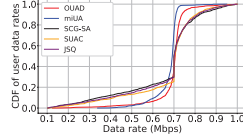


Fig. 9: CDF of all the users' data rates (Set #2, $r_{req} = 0.7$ Mbps).

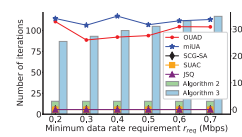


Fig. 10: Time efficiency vs. data rate requirement r_{req} (Set #2).

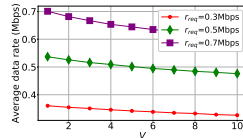


Fig. 11: Parameter V vs. users' average data rate by OUAD (Set #3).

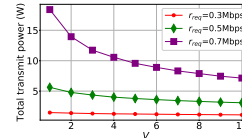


Fig. 12: Parameter V vs. total transmit power by OUAD (Set #3).

only goal is to lower the allocation delay cost. miUA is the most energy-efficient method (Fig. 7) at the cost of very high allocation delays (Fig. 6). OUAD's energy efficiency is very close to miUA, while its average allocation delay is much lower than miUA. In general, when r_{req} increases, all the approaches require more transmit power to serve users (Fig. 8). Their energy efficiency decreases accordingly. Fig. 9 depicts the CDF of the average data rate of all the users. Again, contrasted to OUAD and miUA, SCG-SA, SUAC, and JSQ largely deviate from the minimum data rate requirement $r_{req} = 0.7$ Mbps. They fail to deliver satisfactory data rates to a great number of users and meanwhile, they provide excessive data rates to an equally great number of users. This demonstrates an extremely inefficient use of transmit power.

Fig. 10 shows the time efficiency. Changing r_{req} does not affect Algorithm 2, thus its number of decision iterations remains the same in all settings. The change in r_{req} impacts only Algorithm 3. An increase in r_{req} increases the complexity of Algorithm 3, which now requires more iterations to converge to a Nash equilibrium. This leads to a slight increase in CPU time. Again, in each time slot, OUAD takes roughly 30ms to allocate all new users, which is within an acceptable range and well below the duration of each time slot.

3) *Impact of Parameter V (Set #3):* We examine how parameter V impacts the user average data rate achieved by OUAD under different data rate requirements r_{req} . Based on (8), a greater value of V results in a lower emphasis on the accumulated data rate (6). An increase in V lowers users' average data rate (Fig. 11), and consequently decreases the total transmit power required (Fig. 12). When V is very high, users' average data rate is even below the data rate requirement r_{req} since it now takes longer for users' data rates to converge to r_{req} . Because of the slow convergence, many users already left the system before their data rates can converge to a satisfactory level. This observation shows that app vendors can adjust V to flexibly control the trade-off between users'

average data rate and energy consumption or system cost depending on their app-specific requirements.

VII. CONCLUSION

We address the edge user allocation (EUA) problem in multi-channel multi-cell mobile edge computing (MEC) systems powered by power-domain NOMA. We incorporate the temporal dimension to accommodate random user arrivals and departures. The cost of allocation delay and transmit power is minimized, improving the energy efficiency while satisfying several constraints in NOMA-based MEC systems, including a long-term user data rate constraint. We propose OUAD, a Lyapunov- and game theory-based online user and power allocation algorithm, to allocate users and transmit power without any data of future user arrivals and departures. OUAD is shown to significantly outperform all the representative approaches through a series of experiments.

APPENDIX A PROOF OF LEMMA 1

Proof. We have: $\Delta(D(t)) = \mathbb{E}\{L(D(t+1)) - L(D(t)) | D(t)\}$

$$= \mathbb{E} \left\{ \sum_{u_i \in \mathcal{U}} \left(D_i(t)(r_{req} - r_i(t)) - r_{req} r_i(t) + \frac{r_{req}^2}{2} + \frac{r_i^2(t)}{2} \right) | D(t) \right\}$$

$$\leq O + \mathbb{E} \left\{ \sum_{u_i \in \mathcal{U}} \left(D_i(t)(r_{req} - r_i(t)) - r_{req} r_i(t) + \frac{r_i^2(t)}{2} \right) | D(t) \right\}$$

where $O = \frac{r_{req}^2}{2}$ is a constant. Adding $V(\eta_1 M_i(\mathbf{a}(t)) + \eta_2 L_i(\mathbf{a}(t), \mathbf{p}(t)))$ to both sides of this inequality finishes the proof. \square

APPENDIX B PROOF OF THEOREM 1

Proof. Suppose users u_i 's current decisions $\mathbf{a}_i(t) = (s_j, c_j^k)$, $\mathbf{p}_i(t)$, and its updated decisions $\mathbf{a}'_i(t) = (s_{j'}, c_{j'}^k)$, $\mathbf{p}'_i(t)$ (the decisions made in the next iteration in Algorithms 2 and 3) satisfy $C(\mathbf{a}_i(t), \mathbf{p}_i(t)) \geq C(\mathbf{a}'_i(t), \mathbf{p}'_i(t))$. Based on Eq.s (1) and (4), this theorem must be proven in two cases: 1) $\mathbf{a}_i(t) \neq (0, 0)$ and $\mathbf{a}'_i(t) \neq (0, 0)$, and 2) $\mathbf{a}_i(t) = (0, 0)$ and $\mathbf{a}'_i(t) \neq (0, 0)$. According to the definition of potential games, we need to demonstrate that the decrease in system cost caused by a user u_i updating its current decisions $\mathbf{a}_i(t), \mathbf{p}_i(t)$ to new decisions $\mathbf{a}'_i(t), \mathbf{p}'_i(t)$ will result in a decrease in the potential function (12). By arranging terms and using Eq.s (11), (1), (4), we can see $\phi(\mathbf{a}_i(t), \mathbf{p}_i(t)) - \phi(\mathbf{a}'_i(t), \mathbf{p}'_i(t)) \geq 0$ when $C(\mathbf{a}_i(t), \mathbf{p}_i(t)) - C(\mathbf{a}'_i(t), \mathbf{p}'_i(t)) \geq 0$, which completes the proof. The details are omitted due to limited space. \square

REFERENCES

- [1] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proceedings of INFOCOM*. IEEE, 2018, pp. 756–764.
- [2] L. Cheng and J. Wang, "Vitrac: Efficient tracking on the edge for commodity video surveillance systems," in *Proceedings of INFOCOM*. IEEE, 2018, pp. 1052–1060.
- [3] Y. Saito, Y. Kishiyama, A. Benjebbour, T. Nakamura, A. Li, and K. Higuchi, "Non-orthogonal multiple access (NOMA) for cellular future radio access," in *Proceedings of IEEE Vehicular Technology Conference*. IEEE, 2013, pp. 1–5.
- [4] P. Lai, Q. He, G. Cui, F. Chen, J. Grundy, M. Abdelrazek, J. G. Hosking, and Y. Yang, "Cost-effective user allocation in 5G NOMA-based mobile edge computing systems," *IEEE Transactions on Mobile Computing*, 2021, doi: 10.1109/TMC.2021.3077470.
- [5] G. Cui, Q. He, F. Chen, Y. Zhang, H. Jin, and Y. Yang, "Interference-aware game-theoretic device allocation for mobile edge computing," *IEEE Transactions on Mobile Computing*, 2021, doi: 10.1109/TMC.2021.3064063.
- [6] S. P. Panda, K. Ray, and A. Banerjee, "Dynamic edge user allocation with user specified QoS preferences," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2020, pp. 187–197.
- [7] P. Lai, Q. He, G. Cui, F. Chen, M. Abdelrazek, J. Grundy, J. Hosking, and Y. Yang, "Quality of experience-aware user allocation in edge computing systems: A potential game," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2020, pp. 223–233.
- [8] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2018, pp. 230–245.
- [9] 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.
- [10] K. Wang, Y. Liu, Z. Ding, A. Nallanathan, and M. Peng, "User association and power allocation for multi-cell non-orthogonal multiple access networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5284–5298, 2019.
- [11] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.
- [12] M. R. Garey and D. S. Johnson, *Computers and intractability*. Freeman San Francisco, 1979, vol. 174.
- [13] M. J. Osborne *et al.*, *An introduction to game theory*. Oxford university press New York, 2004, vol. 3, no. 3.
- [14] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996.
- [15] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.
- [16] L. Lei, D. Yuan, C. K. Ho, and S. Sun, "Power and channel allocation for non-orthogonal multiple access in 5G systems: Tractability and computation," *IEEE Transactions on Wireless Communications*, vol. 15, no. 12, pp. 8580–8594, 2016.
- [17] H. Zhang, L. Venturino, N. Prasad, P. Li, S. Rangarajan, and X. Wang, "Weighted sum-rate maximization in multi-cell networks via coordinated scheduling and discrete power control," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 6, pp. 1214–1224, 2011.
- [18] W.-J. Huang, Y.-W. Hong, and C.-C. J. Kuo, "Discrete power allocation for lifetime maximization in cooperative networks," in *Proceedings of IEEE Vehicular Technology Conference*. IEEE, 2007, pp. 581–585.
- [19] P. Lai, Q. He, X. Xia, F. Chen, M. Abdelrazek, J. Grundy, J. G. Hosking, and Y. Yang, "Dynamic user allocation in stochastic mobile edge computing systems," *IEEE Transactions on Services Computing*, 2021, doi: 10.1109/10.1109/TSC.2021.3063148.
- [20] Z. Yang, C. Pan, W. Xu, Y. Pan, M. Chen, and M. ElKashlan, "Power control for multi-cell networks with non-orthogonal multiple access," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 927–942, 2017.
- [21] H. Zeng, X. Zhu, Y. Jiang, Z. Wei, and T. Wang, "A green coordinated multi-cell noma system with fuzzy logic based multi-criterion user mode selection and resource allocation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 3, pp. 480–495, 2019.
- [22] E. 3GPP, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2 (3GPP TS 36.300 version 16.2.0 Release 16)," Tech. Rep., 2020.
- [23] J. Zhang, P. Lu, Z. Li, and J. Gan, "Distributed trip selection game for public bike system with crowdsourcing," in *Proceedings of INFOCOM*. IEEE, 2018, pp. 2717–2725.

Conclusions, Limitations, and Future Work

In this chapter, we conclude this thesis. In particular, Section 6.1 summarises our key contributions and their implications to the society. Section 6.2 discusses the limitations of our work and suggests the research directions that could be taken in the future.

6.1 Key Contributions and Conclusions

The problem of allocating users or mobile devices to cellular base stations is a mature and well-researched problem in conventional cellular networks. However, mobile edge computing (MEC) has arrived with many unique characteristics, namely the presence of computing resources (in edge servers deployed at base stations), new business requirements and pricing models, etc. MEC is a new computing paradigm with a lot of promising potentials, facilitating the availability of a wide range of ultra-low latency applications and services such as smart cities, Industry 4.0, vital health and infrastructure monitoring systems, as well as facial recognition, interactive VR/AR, video streaming, online gaming, just to name a few. By studying the edge user allocation (EUA) in MEC, we hope to accelerate the development of MEC, especially in the 5G/6G era, where the amount of network traffic and subscriptions are expected to grow exponentially.

In the EUA problem, we aim to help app vendors properly allocate their users to edge servers. Depending on the scenarios and application contexts, a number of optimization objectives must be achieved.

Next, we provide a summary of the scenarios that we have investigated, including our contributions:

- In Chapter 2, we have solved Research Problem 1 (Motivating Scenario 1 - Section 1.2.1). We first establish the foundation for the EUA problem and pave the way for future research on the EUA problem. We tackle a static scenario where users are relatively stationary. Our objective is to maximize the number of allocated users and minimize the number of edge servers needed to serve the allocated users. To begin with, we propose an optimal approach using Lexicographic Goal Programming technique. This has resulted in a publication [11], which was selected as the best research paper in the 16th International Conference on Service-Oriented Computing. Since its publication, this paper has attracted a tremendous attention from the MEC community worldwide. Later on, we propose a heuristic to solve this problem in an efficient manner [30].
- In Chapter 3, we have solved Research Problem 2 (Motivating Scenario 2 - Section 1.2.2). We tackle the scenario where users' QoS levels can be adjusted, as opposed to the scenario in Research Problem 1 where the QoS level of every user is fixed. In addition to determining an edge server for each user, the app vendor also needs to determine a QoS level for each user. The objective is to maximize the user satisfaction (measured by their QoE). We first propose an optimal approach and an efficient heuristic approach [29]. The proposed heuristic shows a degradation in the user QoE when the number of users is very high, thus we attempt to improve it with another heuristic [31]. We then noticed that this new heuristic still had some room for improvement in the user QoE. This led us to proposing a decentralized game-theoretical approach [32], whose performance is very close to optimal.
- In Chapter 4, we have solved Research Problem 3 (Motivating Scenario 3 - Section 1.2.3). We study a dynamic scenario where users come and go over time, incorporating the temporal dimension into the EUA problem. This chapter addresses a limitation of Research Problems 1 and 2, i.e., the lack of consideration of user dynamics. The new temporal dimension introduces a new type of cost that app vendors must take into account, which is the user queuing delay cost. To strike a controllable trade-off between the queuing delay cost and system throughput (measured by the number of users allocated to edge servers) over time, we propose an online algorithm [34] based on the Lyapunov optimization framework.
- In Chapter 5, we have solved Research Problem 4 (Motivating Scenario 4 - Section 1.2.4). We realize that wireless communication is an integral part of MEC as users are to be connected to base stations through wireless communication. The communication aspect should not be overlooked in the EUA problem. This is a major limitation of Research Problems 1, 2, and 3, which do not

incorporate the communication aspect of MEC. In this chapter, we address the EUA problem while considering some of the most important characteristics in wireless communication such as the presence of multiple communication channels in a base station, wireless interference, and achievable data rate. In particular, we investigate a non-orthogonal multiple access (NOMA)-based MEC system. NOMA is a new multi-access technique with the ability to facilitate the massive connectivity in 5G/6G networks. In addition to determining an edge server for each user, an app vendor also has to determine a proper channel in a base station/edge server and a sufficient amount of transmit power for each user to minimize the system cost, while ensuring users' data rates. In Section 5.1, we tackle a static scenario, where the system cost comprises of the computing resource and transmit power costs. To solve the EUA problem in this scenario, we propose a game-theoretical approach [33]. In Section 5.2, we tackle a dynamic scenario, where the system cost comprises of the allocation delay and transmit power costs. To solve the EUA problem in this scenario, we propose a Lyapunov optimization and game theory-based approach [76].

6.2 Limitations and Future Work

Being a new problem, the EUA problem can be investigated further from various aspects. In this section, we point out a few potential research directions.

So far, all of our research problems are addressed from the perspective of an app vendor. Looking at those problems from other stakeholders' perspectives, e.g., mobile infrastructure providers or users, other research problems might arise. One is how to ensure the collective benefit of various stakeholders.

Edge servers' performance degradation caused by the interference of workloads, or differentiated user workload patterns, may occur in some applications. This could potentially impact the performance and the available resources of an edge computing system, hence should be considered in future studies. Furthermore, it is possible that some users might not use the entire resources allocated to them during their runtime, leading to an under-utilized edge computing infrastructure. We can thus investigate the scenario where the runtime resource consumption might noticeably differ from the resources allocated to users during the allocation process.

In Chapter 3, we adopt a general QoS-to-QoE model to increase the generalizability of our research. In some domain-specific applications, this model might not be applicable. Thus, finer-grained QoE models

designed for those applications could be considered and incorporated in the future. It is possible that those models cannot be simply plugged into our current problem formulation because those domain-specific applications might involve more constraints. Therefore, the EUA problem might need to be reformulated in those cases.

The online approaches proposed in Chapters 4 and 5 are able to cope with user mobility by treating moving users as new users when they move outside the coverage of a base station/edge server. However, there might be some cases where moving users cannot be treated simply as new users. For example, a user might have session data on their current edge server. An app vendor needs to determine whether to migrate that data to the new edge server, which might incur an extra cost.

There exists a limitation in Chapter 4. In this chapter, the optimization problem in every time slot is solved optimally. If the size of the problem massively scales up and it cannot be solved within a time slot, the app vendor could employ a "divide-and-conquer" approach by dividing a big geographical area into smaller areas and run the algorithm in each of those areas separately. Alternatively, we can find a more efficient approach, such as game theory like we have employed in the dynamic scenario in Chapter 5.

In Chapter 5, we start to incorporate various aspects of wireless communication into the EUA problem. Apart from what we have studied, many other dimensions could be explored. As app vendors now have access to network information such as received signal, received power, throughput, neighbor cells, etc., new security and privacy issues might arise. One could also consider uplink transmissions, massive multiple-input multiple-output, etc.

Finally, all of our performance evaluations so far are experiment-based. Performance analysis of the all proposed solutions on a real-world testbed is highly desirable to validate our research with regards to the effectiveness, efficiency, and practicality.

References

- [1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *Proceedings of the 1st MCC Workshop on Mobile Cloud Computing*. ACM, 2012, pp. 13–16.
- [2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computing—A key technology towards 5G,” *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.
- [3] 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] 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.
- [5] Y. Zeng, Y. Huang, Z. Liu, and Y. Yang, “Joint online edge caching and load balancing for mobile data offloading in 5G networks,” in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2019, pp. 923–933.
- [6] X. Xia, F. Chen, Q. He, G. Cui, P. Lai, M. Abdelrazek, J. Grundy, and H. Jin, “Graph-based optimal data caching in edge computing,” in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2019, pp. 477–493.
- [7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
- [8] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, “Improving web sites performance using edge servers in fog computing architecture,” in *Proceedings of International Symposium on Service-Oriented System Engineering*. IEEE, 2013, pp. 320–323.
- [9] N. Takahashi, H. Tanaka, and R. Kawamura, “Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing,” in *Proceedings of International Conference on Mobile Cloud Computing, Services, and Engineering*. IEEE, 2015, pp. 233–234.
- [10] J. Xu, L. Chen, and P. Zhou, “Joint service caching and task offloading for mobile edge computing in dense networks,” in *Proceedings of International Conference on Computer Communications*. IEEE, 2018, pp. 207–215.

- [11] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2018, pp. 230–245.
- [12] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5G ultra-dense cellular networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, 2016.
- [13] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.
- [14] X. Cao, J. Zhang, and H. V. Poor, "An optimal auction mechanism for mobile edge caching," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2018, pp. 388–399.
- [15] "Ericsson Mobility Report," *Ericsson, Stockholm*, 2019. [Online]. Available: www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf
- [16] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, H. Liu, Y. Qin, and P. Chen, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proceedings of International Conference on Web Services*. IEEE, 2019, pp. 91–98.
- [17] Y. Gan, B. Zhang, and Q. He, "Td-eua: Task-decomposable edge user allocation with qoe optimization," in *Proceedings of International Conference on Service-Oriented Computing*, vol. 12571, no. 10. Springer Nature, 2020, p. 215.
- [18] Z. Xu, G. Zou, X. Xia, Y. Liu, Y. Gan, B. Zhang, and Q. He, "Distance-aware edge user allocation with qoe optimization," in *Proceedings of IEEE International Conference on Web Services*. IEEE, 2020, pp. 66–74.
- [19] T. Li, W. Niu, and C. Ji, "Edge user allocation by foa in edge computing environment," *Journal of Computational Science*, p. 101390, 2021.
- [20] F. Liu, B. Lv, J. Huang, and S. Ali, "Edge user allocation in overlap areas for mobile edge computing," *Mobile Networks and Applications*, pp. 1–11, 2021.
- [21] S. P. Panda, K. Ray, and A. Banerjee, "Dynamic edge user allocation with user specified QoS preferences," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2020, pp. 187–197.
- [22] C. Wu, Q. Peng, Y. Xia, Y. Ma, W. Zheng, H. Xie, S. Pang, F. Li, X. Fu, X. Li *et al.*, "Online user allocation in mobile edge computing environments: A decentralized reactive approach," *Journal of Systems Architecture*, vol. 113, p. 101904, 2021.
- [23] Q. Peng, Y. Xia, Y. Wang, C. Wu, W. Zheng, X. Luo, S. Panz, Y. Ma, and C. Jiang, "A decentralized collaborative approach to online edge user allocation in edge computing environments," in *Proceedings of IEEE International Conference on Web Services*. IEEE, 2020, pp. 294–301.
- [24] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.
- [25] G. Cui, Q. He, F. Chen, Y. Zhang, H. Jin, and Y. Yang, "Interference-aware game-theoretic device allocation for mobile edge computing," *IEEE Transactions on Mobile Computing*, 2021.
- [26] G. Cui, Q. He, X. Xia, P. Lai, F. Chen, T. Gu, and Y. Yang, "Interference-aware SaaS User Allocation Game for Edge Computing," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020, doi: 10.1109/TCC.2020.3008448.

- [27] G. Cui, Q. He, F. Chen, H. Jin, and Y. Yang, "Trading off between User Coverage and Network Robustness for Edge Server Placement," pp. 1–1, 2020, doi: 10.1109/TCC.2020.3008440.
- [28] G. Zou, Y. Liu, Z. Qin, J. Chen, Z. Xu, Y. Gan, B. Zhang, and Q. He, "Td-eua: Task-decomposable edge user allocation with qoe optimization," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2020, pp. 215–231.
- [29] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *Proceedings of International Conference on Service-Oriented Computing*. Springer, 2019, pp. 86–101.
- [30] P. Lai, Q. He, J. Grundy, F. Chen, M. Abdelrazek, J. Hosking, J. Grundy, and Y. Yang, "Cost-effective app user allocation in an edge computing environment," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020, doi: 10.1109/TCC.2020.3001570.
- [31] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "QoE-aware user allocation in edge computing systems with dynamic QoS," *Future Generation Computer Systems*, vol. 112, pp. 684–694, 2020.
- [32] P. Lai, Q. He, G. Cui, F. Chen, M. Abdelrazek, J. Grundy, J. Hosking, and Y. Yang, "Quality of experience-aware user allocation in edge computing systems: A potential game," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2020, pp. 223–233.
- [33] P. Lai, Q. He, G. Cui, F. Chen, J. Grundy, M. Abdelrazek, J. G. Hosking, and Y. Yang, "Cost-Effective User Allocation in 5G NOMA-based Mobile Edge Computing Systems," *IEEE Transactions on Mobile Computing*, 2021, doi: 10.1109/TMC.2021.3077470.
- [34] P. Lai, Q. He, X. Xia, F. Chen, M. Abdelrazek, J. Grundy, J. G. Hosking, and Y. Yang, "Dynamic user allocation in stochastic mobile edge computing systems," *IEEE Transactions on Services Computing*, 2021, doi: 10.1109/10.1109/TSC.2021.3063148.
- [35] A. Lachat, J.-C. Gicquel, and J. Fournier, "How perception of ultra-high definition is modified by viewing distance and screen size," in *Image Quality and System Performance XII*, vol. 9396. International Society for Optics and Photonics, 2015, p. 93960Y.
- [36] M. Yannuzzi, F. van Lingen, A. Jain, O. L. Parellada, M. M. Flores, D. Carrera, J. L. Pérez, D. Montero, P. Chacin, A. Corsaro, and Others, "A new era for cities with fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 54–67, 2017.
- [37] 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, 2016.
- [38] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.
- [39] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.
- [40] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "MeFoRE: QoE based resource estimation at fog to enhance QoS in IoT," in *Proceedings of International Conference on Telecommunications*. IEEE, 2016, pp. 1–5.
- [41] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of Experience (QoE)-aware placement of applications in Fog computing environments," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, 2019.

- [42] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2015.
- [43] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C.-H. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Software: Practice and Experience*, vol. 50, no. 5, pp. 489–502, 2019.
- [44] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.
- [45] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Transactions on Cloud Computing*, 2018.
- [46] 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.
- [47] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2017, pp. 1281–1290.
- [48] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queuing system," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.
- [49] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 1459–1467.
- [50] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang, and S. Ou, "Dynamic resource scheduling in mobile edge cloud with cloud radio access network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2429–2445, 2018.
- [51] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 726–738, 2018.
- [52] F. Liu, Z. Zhou, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in saas clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2648–2658, 2013.
- [53] T. Hobfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE management for cloud applications," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 28–36, 2012.
- [54] Z. Su, Q. Xu, M. Fei, and M. Dong, "Game theoretic resource allocation in media cloud with mobile social users," *IEEE Transactions on Multimedia*, vol. 18, no. 8, pp. 1650–1660, 2016.
- [55] J. He, Y. Wen, J. Huang, and D. Wu, "On the cost-QoE tradeoff for cloud-based video streaming under Amazon EC2's pricing models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 669–680, 2013.
- [56] M. Chen, Y. Zhang, Y. Li, S. Mao, and V. C. Leung, "EMC: Emotion-aware mobile cloud computing in 5G," *IEEE Network*, vol. 29, no. 2, pp. 32–38, 2015.

- [57] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of Experience (QoE)-aware placement of applications in fog computing environments," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, 2019.
- [58] S.-T. Hong and H. Kim, "QoE-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds," in *Proceedings of International Conference on Sensing, Communication, and Networking*. IEEE, 2016, pp. 1–9.
- [59] D. Sabella, V. Sukhomlinov, L. Trang, S. Kekki, P. Paglierani, R. Rossbach, X. Li, Y. Fang, D. Druta, F. Giust *et al.*, "Developing software for multi-access edge computing," *ETSI white paper*, vol. 20, pp. 1–38, 2019. [Online]. Available: www.etsi.org/images/files/etsiwhitepapers/etsi_wp20ed2_mec_softwaredevelopment.pdf
- [60] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.
- [61] Z. Ding, P. Fan, and H. V. Poor, "Impact of non-orthogonal multiple access on the offloading of mobile edge computing," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 375–390, 2018.
- [62] F. Wang, J. Xu, and Z. Ding, "Optimized multiuser computation offloading with multi-antenna NOMA," in *Proceedings of IEEE Globecom Workshops*. IEEE, 2017, pp. 1–7.
- [63] Z. Hasan, H. Boostanimehr, and V. K. Bhargava, "Green cellular networks: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 524–540, 2011.
- [64] M. S. Ali, H. Tabassum, and E. Hossain, "Dynamic user clustering and power allocation for uplink and downlink non-orthogonal multiple access (NOMA) systems," *IEEE Access*, vol. 4, pp. 6325–6343, 2016.
- [65] Z. Ding, P. Fan, and H. V. Poor, "Impact of user pairing on 5G nonorthogonal multiple-access downlink transmissions," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6010–6023, 2015.
- [66] L. You, D. Yuan, L. Lei, S. Sun, S. Chatzinotas, and B. Ottersten, "Resource optimization with load coupling in multi-cell noma," *IEEE Transactions on Wireless Communications*, vol. 17, no. 7, pp. 4735–4749, 2018.
- [67] Z. Yang, C. Pan, W. Xu, Y. Pan, M. Chen, and M. ElKashlan, "Power control for multi-cell networks with non-orthogonal multiple access," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 927–942, 2017.
- [68] K. Wang, Y. Liu, Z. Ding, A. Nallanathan, and M. Peng, "User association and power allocation for multi-cell non-orthogonal multiple access networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5284–5298, 2019.
- [69] F. Guo, H. Lu, D. Zhu, and H. Wu, "Interference-aware user grouping strategy in NOMA systems with QoS constraints," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 1378–1386.
- [70] H. V. Nguyen, V.-D. Nguyen, O. A. Dobre, D. N. Nguyen, E. Dutkiewicz, and O.-S. Shin, "Joint power control and user association for NOMA-based full-duplex systems," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 8037–8055, 2019.

-
- [71] H. Zeng, X. Zhu, Y. Jiang, Z. Wei, and T. Wang, "A green coordinated multi-cell noma system with fuzzy logic based multi-criterion user mode selection and resource allocation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 3, pp. 480–495, 2019.
- [72] J. Cui, Y. Liu, Z. Ding, P. Fan, and A. Nallanathan, "QoE-based resource allocation for multi-cell NOMA networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 9, pp. 6160–6176, 2018.
- [73] L. Salaün, M. Coupechoux, and C. S. Chen, "Weighted sum-rate maximization in multi-carrier NOMA with cellular power constraint," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 451–459.
- [74] M. A. Sedaghat and R. R. Müller, "On user pairing in uplink NOMA," *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3474–3486, 2018.
- [75] X. Zhang, Q. Gao, C. Gong, and Z. Xu, "User grouping and power allocation for noma visible light communication multi-cell networks," *IEEE Communications Letters*, vol. 21, no. 4, pp. 777–780, 2016.
- [76] P. Lai, Q. He, F. Chen, J. Grundy, M. Abdelrazek, J. Hosking, and Y. Yang, "Online user allocation in dynamic noma-based mobile edge computing systems," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2022, in Review.

Appendix **A**

Authorship Statements



Swinburne Research

Authorship Indication Form

For HDR students

NOTE

This Authorship Indication form is a statement detailing the percentage of the contribution of each author in each submitted/published 'paper'. This form must be signed by each co-author and the Principal Supervisor. This form must be added to the publication of your final thesis as an appendix. Please fill out a separate form for each published paper to be included in your thesis.

DECLARATION

We hereby declare our contribution to the publication of the 'paper' entitled:

Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing

First Author

Name: Phu Lai

Signature: 

Percentage of contribution: 80 %

Date: 19 / 05 / 2021

Brief description of contribution to the 'paper' and your central responsibilities/role on project:

Formulated mathematical models, conducted experiments, analyzed results, and wrote manuscript

Second Author

Name: Qiang He

Signature: 

Percentage of contribution: 5 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Supervised project, proposed research direction, revised manuscript

Third Author

Name: Mohamed Abdelrazek

Signature: 

Percentage of contribution: 3 %

Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Fourth AuthorName: Feifei Chen Signature: Percentage of contribution: 3 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fifth AuthorName: John Hosking Signature: Percentage of contribution: 3 % Date: 18 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Sixth AuthorName: John Grundy Signature: Percentage of contribution: 3 % Date: 13/Aug 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Seventh AuthorName: Yun Yang Signature: Percentage of contribution: 3 % Date: 21 / 5 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Eighth Author

Name: _____ Signature: _____

Percentage of contribution: _____ % Date: ____ / ____ / _____

Brief description of your contribution to the 'paper':

Ninth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Tenth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Principal Supervisor:

Name: Qiang He Signature: Date: 20/05/2021

In the case of more than four authors please attach another sheet with the names, signatures and contribution of the authors.



Swinburne Research

Authorship Indication Form

For HDR students

NOTE

This Authorship Indication form is a statement detailing the percentage of the contribution of each author in each submitted/published 'paper'. This form must be signed by each co-author and the Principal Supervisor. This form must be added to the publication of your final thesis as an appendix. Please fill out a separate form for each published paper to be included in your thesis.

DECLARATION

We hereby declare our contribution to the publication of the 'paper' entitled:

Edge User Allocation with Dynamic Quality of Service

First Author

Name: Phu Lai

Signature: 

Percentage of contribution: 80 %

Date: 19 / 05 / 2021

Brief description of contribution to the 'paper' and your central responsibilities/role on project:

Formulated mathematical models, conducted experiments, analyzed results, and wrote manuscript

Second Author

Name: Qiang He

Signature: 

Percentage of contribution: 5 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Supervised project, proposed research direction, revised manuscript

Third Author

Name: Guangming Cui

Signature: 

Percentage of contribution: 2 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fourth AuthorName: Xiaoyu Xia Signature: Percentage of contribution: 2 % Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fifth AuthorName: Mohamed Abdelrazek Signature: Percentage of contribution: 2 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Sixth AuthorName: Feifei Chen Signature: Percentage of contribution: 2 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Seventh AuthorName: John Hosking Signature: Percentage of contribution: 2 % Date: 18 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Eighth AuthorName: John Grundy Signature: Percentage of contribution: 2 % Date: 13 / Aug / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Ninth AuthorName: Yun Yang Signature: Percentage of contribution: 3 %

Date: 21 / 5 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Tenth Author

Name: _____ Signature: _____

Percentage of contribution: _____ %

Date: __ / __ / ____

Brief description of your contribution to the 'paper':

Principal Supervisor:

Name: Qiang He Signature: Date: 20 / 05 / 2021

In the case of more than four authors please attach another sheet with the names, signatures and contribution of the authors.



Swinburne Research

Authorship Indication Form

For HDR students

NOTE

This Authorship Indication form is a statement detailing the percentage of the contribution of each author in each submitted/published 'paper'. This form must be signed by each co-author and the Principal Supervisor. This form must be added to the publication of your final thesis as an appendix. Please fill out a separate form for each published paper to be included in your thesis.

DECLARATION

We hereby declare our contribution to the publication of the 'paper' entitled:

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

First Author

Name: Phu Lai

Signature: 

Percentage of contribution: 80 %

Date: 19/05/2021

Brief description of contribution to the 'paper' and your central responsibilities/role on project:

Formulated mathematical models, conducted experiments, analyzed results, and wrote manuscript

Second Author

Name: Qiang He

Signature: 

Percentage of contribution: 5 %

Date: 20/05/2021

Brief description of your contribution to the 'paper':

Supervised project, proposed research direction, revised manuscript

Third Author

Name: Guangming Cui

Signature: 

Percentage of contribution: 2 %

Date: 20/05/2021

Brief description of your contribution to the 'paper':

Provided technical support

Fourth AuthorName: Feifei Chen Signature: Percentage of contribution: 2 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fifth AuthorName: Mohamed Abdelrazek Signature: Percentage of contribution: 2 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Sixth AuthorName: John Grundy Signature: Percentage of contribution: 3 % Date: 13/Aug 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Seventh AuthorName: John Hosking Signature: Percentage of contribution: 2 % Date: 18 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Eighth AuthorName: Yun Yang Signature: Percentage of contribution: 4 % Date: 21 / 5 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Ninth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Tenth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Principal Supervisor:

Name: Qiang He Signature: Date: 20/05/2021

In the case of more than four authors please attach another sheet with the names, signatures and contribution of the authors.



Swinburne Research

Authorship Indication Form

For HDR students

NOTE

This Authorship Indication form is a statement detailing the percentage of the contribution of each author in each submitted/published 'paper'. This form must be signed by each co-author and the Principal Supervisor. This form must be added to the publication of your final thesis as an appendix. Please fill out a separate form for each published paper to be included in your thesis.

DECLARATION

We hereby declare our contribution to the publication of the 'paper' entitled:

Cost-Effective App User Allocation in an Edge Computing Environment

First Author

Name: Phu Lai

Signature: 

Percentage of contribution: 80 %

Date: 19 / 05 / 2021

Brief description of contribution to the 'paper' and your central responsibilities/role on project:

Formulated mathematical models, conducted experiments, analyzed results, and wrote manuscript

Second Author

Name: Qiang He

Signature: 

Percentage of contribution: 5 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Supervised project, proposed research direction, revised manuscript

Third Author

Name: John Grundy

Signature: 

Percentage of contribution: 3 %

Date: 13 Aug/2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Fourth AuthorName: Feifei Chen Signature: Percentage of contribution: 3 % Date: 10/08 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fifth AuthorName: Mohamed Abdelrazek Signature: Percentage of contribution: 3 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Sixth AuthorName: John Hosking Signature: Percentage of contribution: 3 % Date: 18 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Seventh AuthorName: Yun Yang Signature: Percentage of contribution: 3 % Date: 21 / 5 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Eighth Author

Name: _____ Signature: _____

Percentage of contribution: _____ % Date: ____ / ____ / _____

Brief description of your contribution to the 'paper':

Ninth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Tenth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Principal Supervisor:

Name: Qiang He Signature: Date: 20/05/2021

In the case of more than four authors please attach another sheet with the names, signatures and contribution of the authors.



Swinburne Research

Authorship Indication Form

For HDR students

NOTE

This Authorship Indication form is a statement detailing the percentage of the contribution of each author in each submitted/published 'paper'. This form must be signed by each co-author and the Principal Supervisor. This form must be added to the publication of your final thesis as an appendix. Please fill out a separate form for each published paper to be included in your thesis.

DECLARATION

We hereby declare our contribution to the publication of the 'paper' entitled:

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

First Author

Name: Phu Lai

Signature: 

Percentage of contribution: 80 %

Date: 19 / 05 / 2021

Brief description of contribution to the 'paper' and your central responsibilities/role on project:

Formulated mathematical models, conducted experiments, analyzed results, and wrote manuscript

Second Author

Name: Qiang He

Signature: 

Percentage of contribution: 5 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Supervised project, proposed research direction, revised manuscript

Third Author

Name: Guangming Cui

Signature: 

Percentage of contribution: 2 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fourth AuthorName: Xiaoyu Xia Signature: Percentage of contribution: 2 % Date: 20 / 05 / 2021Brief description of your contribution to the 'paper':
Provided technical support**Fifth Author**Name: Mohamed Abdelrazek Signature: Percentage of contribution: 2 % Date: 10 / 08 / 2021Brief description of your contribution to the 'paper':
Co-supervised project, revised manuscript**Sixth Author**Name: Feifei Chen Signature: Percentage of contribution: 2 % Date: 10 / 08 / 2021Brief description of your contribution to the 'paper':
Provided technical support**Seventh Author**Name: John Hosking Signature: Percentage of contribution: 2 % Date: 18 / 08 / 2021Brief description of your contribution to the 'paper':
Co-supervised project, revised manuscript**Eighth Author**Name: John Grundy Signature: Percentage of contribution: 2 % Date: 13 Aug 2021Brief description of your contribution to the 'paper':
Co-supervised project, revised manuscript

Ninth AuthorName: Yun Yang Signature: Percentage of contribution: 3 %

Date: 21 / 5 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Tenth Author


Name: _____ Signature: _____

Percentage of contribution: _____ %

Date: __ / __ / ____

Brief description of your contribution to the 'paper':

Principal Supervisor:

Name: Qiang He Signature: Date: 20 / 05 / 2021

In the case of more than four authors please attach another sheet with the names, signatures and contribution of the authors.



Swinburne Research

Authorship Indication Form

For HDR students

NOTE

This Authorship Indication form is a statement detailing the percentage of the contribution of each author in each submitted/published 'paper'. This form must be signed by each co-author and the Principal Supervisor. This form must be added to the publication of your final thesis as an appendix. Please fill out a separate form for each published paper to be included in your thesis.

DECLARATION

We hereby declare our contribution to the publication of the 'paper' entitled:

Dynamic User Allocation in Stochastic Mobile Edge Computing Systems

First Author

Name: Phu Lai

Signature: 

Percentage of contribution: 80 %

Date: 19 / 05 / 2021

Brief description of contribution to the 'paper' and your central responsibilities/role on project:

Formulated mathematical models, conducted experiments, analyzed results, and wrote manuscript

Second Author

Name: Qiang He

Signature: 

Percentage of contribution: 5 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Supervised project, proposed research direction, revised manuscript

Third Author

Name: Xiaoyu Xia

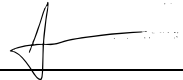
Signature: 

Percentage of contribution: 2 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fourth AuthorName: Feifei Chen Signature: Percentage of contribution: 2 % Date: 10/08/2021Brief description of your contribution to the 'paper':
Provided technical support**Fifth Author**Name: Mohamed Abdelrazek Signature: Percentage of contribution: 2 % Date: 10/08/2021Brief description of your contribution to the 'paper':
Co-supervised project, revised manuscript**Sixth Author**Name: John Grundy Signature: Percentage of contribution: 3 % Date: 13 Aug 2021Brief description of your contribution to the 'paper':
Co-supervised project, revised manuscript**Seventh Author**Name: John Hosking Signature: Percentage of contribution: 2 % Date: 18/08/2021Brief description of your contribution to the 'paper':
Co-supervised project, revised manuscript**Eighth Author**Name: Yun Yang Signature: Percentage of contribution: 4 % Date: 21/5/2021Brief description of your contribution to the 'paper':
Co-supervised project, revised manuscript

Ninth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Tenth Author


Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Principal Supervisor:

Name: Qiang He Signature: Date: 20/05/2021

In the case of more than four authors please attach another sheet with the names, signatures and contribution of the authors.



Swinburne Research

Authorship Indication Form

For HDR students**NOTE**

This Authorship Indication form is a statement detailing the percentage of the contribution of each author in each submitted/published 'paper'. This form must be signed by each co-author and the Principal Supervisor. This form must be added to the publication of your final thesis as an appendix. Please fill out a separate form for each published paper to be included in your thesis.

DECLARATION

We hereby declare our contribution to the publication of the 'paper' entitled:

Cost-Effective User Allocation in 5G NOMA-based Mobile Edge Computing Systems

First AuthorName: Phu LaiSignature: Percentage of contribution: 80 %Date: 19 / 05 / 2021

Brief description of contribution to the 'paper' and your central responsibilities/role on project:

Formulated mathematical models, conducted experiments, analyzed results, and wrote manuscript

Second AuthorName: Qiang HeSignature: Percentage of contribution: 5 %Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Supervised project, proposed research direction, revised manuscript

Third AuthorName: Guangming CuiSignature: Percentage of contribution: 2 %Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fourth AuthorName: Feifei Chen Signature: Percentage of contribution: 2 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fifth AuthorName: John Grundy Signature: Percentage of contribution: 3 % Date: 13 Aug/2021

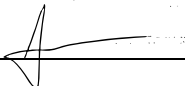
Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Sixth AuthorName: Mohamed Abdelrazek Signature: Percentage of contribution: 2 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Seventh AuthorName: John Hosking Signature: Percentage of contribution: 2 % Date: 18 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Eighth AuthorName: Yun Yang Signature: Percentage of contribution: 4 % Date: 21 / 5 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Ninth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Tenth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Principal Supervisor:

Name: Qiang He Signature: Date: 20/05/2021

In the case of more than four authors please attach another sheet with the names, signatures and contribution of the authors.



Swinburne Research

Authorship Indication Form

For HDR students

NOTE

This Authorship Indication form is a statement detailing the percentage of the contribution of each author in each submitted/published 'paper'. This form must be signed by each co-author and the Principal Supervisor. This form must be added to the publication of your final thesis as an appendix. Please fill out a separate form for each published paper to be included in your thesis.

DECLARATION

We hereby declare our contribution to the publication of the 'paper' entitled:

Online User and Power Allocation for Dynamic NOMA-based Mobile Edge Computing

First Author

Name: Phu Lai

Signature: 

Percentage of contribution: 80 %

Date: 19 / 05 / 2021

Brief description of contribution to the 'paper' and your central responsibilities/role on project:

Formulated mathematical models, conducted experiments, analyzed results, and wrote manuscript

Second Author

Name: Qiang He

Signature: 

Percentage of contribution: 5 %

Date: 20 / 05 / 2021

Brief description of your contribution to the 'paper':

Supervised project, proposed research direction, revised manuscript

Third Author

Name: Feifei Chen

Signature: 

Percentage of contribution: 3 %

Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Provided technical support

Fourth AuthorName: John Grundy Signature: Percentage of contribution: 3 % Date: 13/Aug 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Fifth AuthorName: Mohamed Abdelrazek Signature: Percentage of contribution: 3 % Date: 10 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Sixth AuthorName: John Hosking Signature: Percentage of contribution: 3 % Date: 18 / 08 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Seventh AuthorName: Yun Yang Signature: Percentage of contribution: 3 % Date: 21 / 5 / 2021

Brief description of your contribution to the 'paper':

Co-supervised project, revised manuscript

Eighth Author

Name: _____ Signature: _____

Percentage of contribution: _____ % Date: ____ / ____ / ____

Brief description of your contribution to the 'paper':

Ninth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Tenth Author

Name: _____ Signature: _____

Percentage of contribution: _____%

Date: __/__/____

Brief description of your contribution to the 'paper':

Principal Supervisor:

Name: Qiang He Signature: Date: 20/05/2021

In the case of more than four authors please attach another sheet with the names, signatures and contribution of the authors.