ST-EUA: Spatio-temporal Edge User Allocation with Task Decomposition

Guobing Zou, Ya Liu, Zhen Qin, Jin Chen, Zhiwei Xu, Yanglan Gan*, Bofeng Zhang, Qiang He*

Abstract—Recently, edge user allocation (EUA) problem has received much attentions. It aims to appropriately allocate edge users to their nearby edge servers. Existing EUA approaches suffer from a series of limitations. First, considering users' service requests only as a whole, they neglect the fact that in many cases a service request may be partitioned into multiple tasks to be performed by different edge servers. Second, the impact of the spatial distance between edge users and servers on users' quality of experience is not properly considered. Third, the temporal dynamics of users' service requests has not been fully considered. To overcome these limitations systematically, this paper focuses on the problem of spatio-temporal edge user allocation with task decomposition (ST-EUA). We first formulate the ST-EUA problem. Then, we transform ST-EUA problem as an optimization problem with multiple objectives and global constraints and prove its $\mathcal{NP}$-hardness. To tackle the ST-EUA problem effectively and efficiently, we propose a novel genetic algorithm-based heuristic approach called GA-ST, aiming to maximize users' overall QoE while minimizing the cost of task migration in different time slots. Extensive experiments are conducted on two widely-used real-world datasets to evaluate the performance of our approach. The results demonstrate that GA-ST significantly outperforms state-of-the-art approaches in finding approximate solutions in terms of the trade-off among multiple metrics.

Index Terms—Spatio-temporal EUA, Task decomposition, Quality of Experience, Migration cost, Service request.

1 Introduction

With the rapid development of Internet of Things (IoT) and wireless communication technologies, the world has witnessed a surge in the number of mobile and IoT devices (referred to as *end-devices* hereafter), including mobile phones, wearables, sensors and a wide range of IoT devices. It is estimated that the internet will connect up to 32 billion end-devices by 2023 according to Ericsson's Mobility Report [1]. Due to the limited computing and storage resources of end-devices, they often need to access various services deployed on remote cloud centers, which generates tremendous network traffic and contributes to network congestion significantly. In the meantime, this cloud computing paradigm is challenged by many services demanding low latency, especially those that require real-time interactions with users, such as autopilot, virtual reality (VR) and augmented reality (AR).

To tackle this issue, edge computing has emerged as a novel distributed computing paradigm as an extension of the cloud computing paradigm. It allows different kinds of resources, such as CPU, RAM, Storage, Bandwidth, etc., to be provided by edge servers deployed at the network edge within users' close geographic proximity. In edge computing, edge servers are deployed at cellular base stations and cover different geographical areas. Adjacent edge servers' coverages usually intersect to avoid blank areas not covered by any edge server. Under the *coverage constraint* [2], users can connect to nearby edge servers - those that cover the users - via radio access and submit service requests to them for processing. In this way, the central cloud is not required to process all the service requests. This significantly reduces the service latency as well as the network traffic over the backhaul network.

Although edge computing has emerged as a key 5G enabling technology for realizing the IoT visions [3], it still faces many new and critical challenges. Edge user allocation problem (EUA) is as one of those challenges and has received a lot of attentions in recent years [2], [4]. In the edge computing environment, service vendors hire resources on edge servers to provide edge users with different kinds of services, such as IoT services, AI services and mobile services. Under the *capacity constraint* [4], EUA aims to help service vendors allocate their users to edge servers properly so as to cost-effectively utilize the diverse resources hired on edge servers. Existing EUA approaches try to achieve a specific optimization goal, e.g., to minimize the number of edge servers needed [2], [5], to maximize the user satisfaction measured by their Quality of Experience (QoE) [6], [7], or to increase the overall ratio of allocated users [4], [6], [7], [8].

However, existing approaches suffer from a series of limitations. First, most EUA approaches model the problem as a static global optimization problem. However, the real-world edge computing environment features temporal dynamics, in particular the variations in the distributions of users and their requests over time. Second, existing EUA approaches neglect the spatial feature of the EUA problem. They do not consider the distance between edge servers and users which impacts the wireless signal transmission

- *G. Zou, Y. Liu, Z. Qin, J. Chen, Z. Xu, B. Zhang are with the School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China.*
  E-mail: gbzou@shu.edu.cn, ambersoul@shu.edu.cn, zhenqin@shu.edu.cn, cj1125@shu.edu.cn, zhiweixu@shu.edu.cn, bfzhang@shu.edu.cn.
- *Y. Gan is with the School of Computer Science and Technology, Donghua University, Shanghai 201620, China.*
  E-mail: ylgan@dhu.edu.cn.
- *Q. He is with the Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne, Australia.*
  E-mail: qhe@swin.edu.au.
  **Corresponding authors*

between them and consequently users' data rates [9]. Third, existing approaches always treat a user request as a whole and fail to accommodate scenarios where user requests may be partitioned for processing by different edge servers [10].

In real-world scenarios, from the service provider's perspective, cost-effective EUA aims to maximize its users' overall QoE, taking the above issues into consideration. This problem is referred to as the Spatio-Temporal user Allocation (ST-EUA) problem. Expanding beyond our previous work [9], [10], we take into account the temporal dynamics into account and try to address the above issues systematically. To solve the $\mathcal{NP}$-hard ST-EUA problem effectively and efficiently, this paper proposes a novel genetic algorithm (GA) based approach named GA-ST, with the aim to find an approximate solution efficiently. Specifically, when allocating users to their nearby edge servers with task decomposition, GA-ST considers the spatial distance between edge servers and users, the temporal variations in the distribution of users and their requests, and the unique constraints in the edge computing environment, including the coverage and capacity constraint. To our best knowledge, this paper is the first attempt to tackle the spatio-temporal EUA problem with task decomposition. The main contributions of this paper are summarized as follows.

- We formally formulate the ST-EUA problem, model it as a constrained optimization problem, and prove its $\mathcal{NP}$-hardness.
- To solve the ST-EUA problem effectively and efficiently, we propose a novel GA-based heuristic approach named GA-ST to find sub-optimal solutions.
- Extensive experiments are conducted on two widely-used real-world datasets to evaluate the performance of GA-ST. The effectiveness and efficiency of GA-ST are demonstrated with a comparison against four baselines, four state-of-the-art approaches and four representative GA-based approaches. The experiment results show that GA-ST significantly outperforms all these competing approaches.

The remainder of the paper is organized as follows. Section 2 provides a motivating example for this research. Section 3 formulates the ST-EUA problem. Section 4 models the ST-EUA problem as a constrained optimization problem, proves the $\mathcal{NP}$-hardness of the problem, and presents GA-ST in detail. Section 5 experimentally evaluates GA-ST. Section 6 reviews the related work. Section 7 concludes the paper and points out the future work.

2 Motivating Example

Fig. 1 presents an example spatio-temporal EUA scenario. There are five users, denoted by $\{u_1, u_2, u_3, u_4, u_5\}$, three edge servers, denoted by $\{s_1, s_2, s_3\}$, and nine tasks, denoted by $\{a_1, a_2, \ldots, a_9\}$. These tasks submitted by users can be processed by services deployed on the edge servers covering them. Please note that the computing resources and storage resources on edge server are limited. Each user has a list of tasks and each task may require different amounts of computing resources.

Task Decomposition. Existing studies of EUA [2], [6], [9], [11] simply assume that a user's computing resource demand can either be fully fulfilled by a single edge server

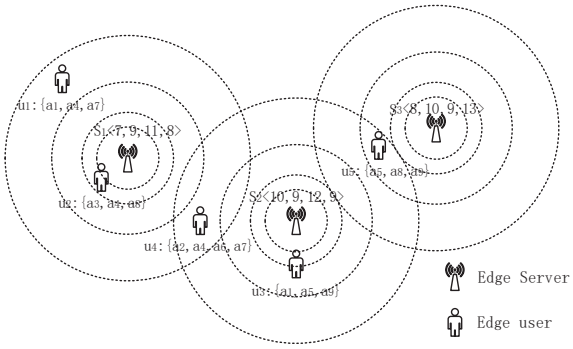


Fig. 1. A motivating example of an ST-EUA problem.

or cannot be fulfilled at all. However, in real-world scenarios, a user request may be be partitioned into multiple tasks that can be performed by different edge servers. Let us assume that u_1 and u_2 are fully satisfied by edge server s_1, while u_3 and u_5's tasks are assigned to edge server s_2, where the resources demanded by each task is $\langle 1, 1, 1, 1\rangle$. As a result, the remaining resources on s_1 or s_2 cannot fulfil the request of u_4. Therefore, current EUA approaches will have to allocate u_4 to the remote cloud instead of any edge servers. If a user service request can be partitioned into a set of tasks, it can be processed by multiple edge servers. Thus, a_2 can be assigned to s_1 and task $\{a_4, a_6, a_7\}$ can be assigned to s_2. In this way, the total amount of resources required by s_1 to perform the submitted tasks is $\langle 7, 7, 7, 7\rangle$. It does not exceed s_1's capacity of $\langle 7, 9, 11, 8\rangle$. In the meantime, s_2 also has abundant resources to process the tasks assigned to it.

Distance Awareness. Existing EUA approaches [2], [6], [11] assume that each edge server has a specific coverage radius and the users covered by the same edge server have the same data rate. However, when a user communicates with an edge server, the wireless transmission between them follows a slow attenuation pattern [12]. That is, the wireless signal strength relies on the distance between the user and the edge server, the closer the stronger. For example, u_1, u_2 and u_4 are covered by s_1. Since u_2 is closer to s_1, it receives a stronger wireless signal than u_1 and u_4. This translates to a higher data rate. On the contrary, being far away from s_1, u_1 and u_4 may receive an unsatisfactory data rate, which lowers their QoE. Thus, geographical distance between users and edge servers must be considered in EUA to maximize users' overall QoE.

Temporal Dynamics. Existing studies [2], [6], [9], [11] treat an EUA problem as a static global optimization problem and aim at finding an optimal or near-optimal solution. However, they are ineffective in dynamic EUA scenarios because they cannot handle the variations in the distribution of users and their service requests over time. Take u_1 as an example. It submits one service request $\{a_1, a_4, a_7\}$ in time slot t_1, one service request $\{a_1, a_4\}$ in t_2 and one service request $\{a_1, a_2, a_4\}$ in t_3. Such service demand dynamics must be considered in the EUA problem to achieve the service provider's optimization objective.

3 System Model

3.1 Task Decomposition and Capacity Constraint

Given a finite set of m edge servers $S = \{s_1, s_2, \ldots, s_m\}$, and n users $U_t = \{u_1, u_2, \ldots, u_n\}$ in a time slot t, a service

TABLE 1
Notations

Notation	Description
$B = \{CPU, RAM, storage, bandwidth\}$	a set of computing resource types
$T = \{t_1, t_2, \ldots, t_p\}$	a set of time slots
$U_t = \{u_1, u_2, \ldots, u_n\}$	a set of edge users in time slot t
$S = \{s_1, s_2, \ldots, s_m\}$	a set of edge servers
$A_t = \{a_1, a_2, \ldots, a_q\}$	a set of tasks decomposed from users' service requests in t
$D_t = \{d_{11}, d_{12}, \ldots, d_{nm}\}$	a set of geographical distance in t
d_{ij}	geographical distance between u_i and s_j in D_t
$f(d_{ij})$	attenuation coefficient caused by distance d_{ij}
C_j^t	available capacity of edge server s_j in t
$A_t(u_i)$	a set of tasks u_i needs in a service request in t
$w_k = \langle w_k^1, w_k^2, \ldots, w_k^d \rangle$	computing resources demanded for the task a_k
$A_t(s_j)$	a set of tasks allocated to s_j in t
$r_{a_k}^t(u_i)$	A boolean indicator of whether u_i's task a_k is reallocated in t
$l_{a_k}^t(u_i)$	migration cost of user u_i' task a_k in t
W_i^t	computing resources that u_i is assigned in t

request submitted by a user can be decomposed as a set of independent tasks, defined as follows.

Definition 1. *(Task Decomposition) A user's service request r is composed of a set of independent tasks $A_t(u_i) = \{a_1, a_2, \ldots\}$. Each task a_k can be processed by an edge server $s_j \in S$ covering the user.*

The total amount of resources required by all the tasks allocated to an edge server must not exceed its capacity available in that time slot. Otherwise, the edge server will be overloaded, causing performance degradation and even service disruptions.

Definition 2. *(Capacity Constraint) Given an edge server s_j and users within its coverage area in time slot t, denoted as $U_c^t = \{u_c^1, u_c^2, ...\}$, let us denote the tasks allocated to each edge server $s_j \in S$ as $A_t(s_j) = \left\{a_{s_j}^1, a_{s_j}^2, ...\right\}$. The total amount of resources required by these tasks must not exceed s_j's resources available in time slot t:*

$$\sum_{a_{s_j}^k \in A_t(s_j)} w_k \leq C_j^t, \forall s_j \in S \tag{1}$$

Take Fig. 1 for example. The resource required by each task is $\langle 1, 1, 1, 1 \rangle$, and the total amount of resources required by u_1 and u_2 is $\langle 6, 6, 6, 6 \rangle$. It does not exceed s_1's available capacity $\langle 7, 9, 11, 8 \rangle$.

3.2 Distance-aware QoE Model

In this study, we measure a user's QoE in the same way as [6], which depends on the Quality of the Service (QoS) delivered to the user. According to [13], a user's QoS is non-linearly correlated with its QoE. Generally, it starts to increase slowly at first, then speeds up, and finally converges. The correlation between QoS and QoE is application-specific. In many studies, the correlation between QoE and QoS is modeled with the sigmoid function [6]:

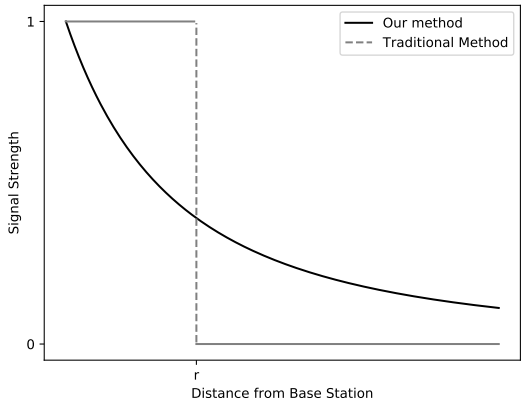


Fig. 2. Quantitative correlation between distance and signal strength.

$$E_i^0 = \frac{L}{1 + e^{-\alpha(x_i - \beta)}} \tag{2}$$

where L is the maximum QoE value, β is a domain-specific parameter that controls the QoE growth, and α is the growth rate of the QoE level, i.e., how significant the change from the minimum to the maximum QoE is; E_i^0 represents the QoE level given user u_i's QoS level W_i^t, and $x_i = \frac{W_i^t}{|B|}$. There is $E_i^0 = 0$ if u_i is not allocated to any edge servers.

In a general EUA scenario, a user's QoE relies on the computing resources allocated to it from the edge servers, e.g., CPU, RAM, storage and data rate [6]. To measure the data rate received by a user, the attenuation of data rate in wireless transmission must be taken into account. The Free Space Path Loss (FSPL) [12] model is part of the IEEE 802.11 standard, where the received power is impacted by many factors, such as transmit power, antenna gain and distance between the transmitter and the receiver. Accordingly, the received power decreases with the square of the distance between the user and the edge server. The signal power attenuation in a free space can be calculated as:

$$f(d) = \frac{P_r}{P_t} = G_t G_r (\frac{\lambda}{4\pi d})^2 \tag{3}$$

where P_r and P_t are the received power and transmitted power, respectively; G_r and G_t are the receiver antenna gain and transmitter antenna gain, respectively; λ is the wavelength and d is the distance between transmitter and receiver; G_r and G_t are commonly set to 1.

In the edge computing environment, edge servers are attached to base stations. The transmitted power and antenna gain of the base stations are ensured and controlled by network operators. Thus, the distance is the key to finding an proper solution to the ST-EUA problem. As illustrated in Fig. 2, a user's signal strength is attenuated by the increasing distance from the edge server. According to Eq. (3), the attenuation coefficient can be calculated as:

$$f(d_{ij}) = (\frac{\xi}{d_{ij}})^2 \tag{4}$$

where $f(d_{ij})$ is the attenuation coefficient for the communication between u_i and s_j.

Given the above attenuation coefficient, a user's data rate

and QoE can be calculated. As a result, we measure a user's QoE in an ST-EUA scenario, where the tasks of a service request may be allocated to multiple edge servers as follows:

$$E_i = \frac{W_i^t}{\sum_{a_k \in A_t(u_i)} w_k} f(d_{ij}) E_i^0 \tag{5}$$

where $\frac{W_i^t}{\sum_{a_k \in A_t(u_i)} w_k}$ is the ratio of assigned computing resources over the total computing resources required for fulfilling u_i's task requests in time slot t, $f(d_{ij})$ is the attenuation coefficient caused by the distance d_{ij} and E_i^0 is QoE calculated based on the corresponding QoS.

3.3 Migration Cost with Temporal Feature

In edge computing, when users move out from the signal range of their allocated edge servers, the service is interrupted. Thus, reallocation is required through service migration between the source edge server and the destination edge server. Generally, service migration cost consists of data transmission time and service restart time on the destination edge server. When performing service migration for a user, data transmission, e.g., memory state data, application image data, etc, is inevitable. Moreover, different network topologies and communication systems (e.g., WiFi, LTE-U, 4G and 5G) result in various transmission delays and processing costs. Therefore, the time cost incurred by data transmission from the source edge server to the destination edge server must be considered in service migration in ST-EUA.

In an ST-EUA problem, under the *proximity constraint*, the variations in user distribution and users' service requests in different time slots may result in significant network performance degradation, dramatic drop in QoS, and even service interruptions. Therefore, services must be migrated from source edge servers to destination edge servers to ensure service continuity for users if those services are unavailable on the destination edge servers. Similar to [14], [15], [16], we focus on the service migration in different time slots, calculating the service migration cost for a user in a time slot as follows.

Definition 3. *(Migration Cost) Given a set of time slots $T = \{t_1, t_2, ..., t_p\}$ and a set of users $U_t = \{u_1, u_2, \dots, u_n\}$ in time slot t, each user $u_i \in U_t$ submits a service request r_i. Let $R_t = \{r_1, r_2, \dots, r_n\}$ denote all the service requests in time slot t. Each user's service request is composed of a set of tasks $A_t(u_i) = \{a_1, a_2, \dots\}$. Here, the cost $l_{a_k}^t(u_i)$ incurred by migration task $a_k \in A_t(u_i)$ is defined as the span between the time when the service migration starts and the time when the migrated service starts on the destination edge server.*

Here, data transmission delay for service migration from the source edge server to the destination edge server can be calculated as follows:

$$l_{mig} = z_{a_k}^t(u_i) / v_{a_k}^t(u_i) \tag{6}$$

where $z_{a_k}^t(u_i)$ indicates the size of the data that needs to be transferred to the target edge server during service migration for u_i's task a_k in time slot t, and $v_{a_k}^t(u_i)$ is the transmission rate from the source edge server to the destination edge server.

Moreover, the service needs to be started on the target server to respond to user u_i's task request, which will also bring a certain time cost. Therefore, the total delay caused by the service migration to the user u_i is calculated as follows:

$$l_{a_k}^t(u_i) = l_{com} \frac{w_k}{C_j^t} + l_{mig} \tag{7}$$

where $l_{a_k}^t(u_i)$ is the total delay, l_{com} is the delay caused by request processing on the destination edge server, w_k is the amount of the requested resources, C_j^t is the capacity of the destination edge server, and l_{mig} is the delay caused by service migration.

3.4 ST-EUA Problem

Definition 4. *(Spatio-temporal User Allocation Problem) An ST-EUA problem can be defined as a six tuple $ST-EUA =< T, U_t, S, D_t, A_t, W_t >$, where*

(1) $T = \{t_1, t_2, \dots, t_p\}$ is a set of time slots;
(2) $U_t = \{u_1, u_2, ..., u_n\}$ is a set of users in time slot t and each user has a service request composed by a set of tasks;
(3) $S = \{s_1, s_2, ..., s_m\}$ is a set of edge servers;
(4) $D_t = \{d_{11}, d_{12}, \dots, d_{nm}\}$ is a set of distances between users and edge servers in time slot t;
(5) $A_t = \{a_1, a_2, ..., a_q\}$ is a set of tasks decomposed from users' service requests in time slot t;
(6) $W_t = \{w_1, w_2, ..., w_q\}$ is a set of resources demanded by A_t.

4 Approach

4.1 ST-EUA Optimization Model

Given an ST-EUA problem ST-EUA=$\langle T, U_t, S, D_t, A_t, W_t \rangle$, there are two optimization objectives: (1) maximizing the users' overall QoE across multiple time slots, and (2) minimizing the average migration cost incurred by users' tasks in multiple time slots, while satisfying the proximity constraint and capacity constraint. The LGP model of the ST-EUA problem is as follows:

$$\max \frac{1}{|T|} \sum_{t \in T} \sum_{i=1}^{|U_t|} E_i \tag{8}$$

$$\min \frac{1}{|T|} \sum_{t \in T} \sum_{i=1}^{|U_t|} \sum_{k=1}^{|A_t(u_i)|} r_{a_k}^t(u_i) l_{a_k}^t(u_i) \tag{9}$$

s.t:

$$\sum_{i=1}^{|U_t|} \sum_{k=1}^{|A_t(u_i)|} w_k x_{i,j,k} \le C_j^t \tag{10}$$

$$\sum_{j=1}^{|S|} x_{i,j,k} \le 1 \tag{11}$$

where $x_{i,j,k}$ and $r_{a_k}^t(u_i)$ are two binary variables indicating that,

$$x_{i,j,k} = \begin{cases} 1, & \text{if } u_i's \text{ task } a_k \text{ is allocated to } s_j \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

$$r_{a_k}^t(u_i) = \begin{cases} 1, & \text{if } u_i's \text{ task } a_k \text{ is reallocated} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The objective function (8) maximizes users' overall average QoE across all the time slots, where E_i depends on the

ratio of the assigned computing resources W_i^t for u_i over the total computing resources requested by u_i. The objective function (9) minimizes the overall average migration cost caused by users' tasks reallocated in different time slots. Constraint (10) makes sure that the aggregate resource demands of all users' tasks allocated to an edge server must not exceed its available upper bound capacity in time slot t. Constraint (11) ensures that each task can be allocated to at most one edge server.

This optimization model takes into account the characteristics of edge computing. First, unlike cloud servers that have access to virtually unlimited computing resources in the cloud center, edge servers' computing resources are restricted due to its physical size limit [17], [18]. In the open edge computing environment, service providers may hire computing resources on the edge servers to serve their users in the same area. It causes competition among service providers, where the total resources required serving the users allocated to an edge server must not exceed its available capacity. Second, in the cloud computing environment, there often is no hard requirement for the time taken to process a task. Tasks can be scheduled to be processed on slow cloud servers as long as the main optimization objective is fulfilled, e.g., system throughput or reliability. However, low service latency is a fundamental requirement in the edge computing environment. Thus, task allocation optimization must not ignore the time constraint. It is often modelled as an objective or a constraint in the optimization model. Third, it is often assumed that a task can be allocated to any servers for processing in the cloud computing environment. However, in the edge computing environment, a user's tasks can only be processed by an edge server covering that user. Thus, tasks cannot be allocated arbitrarily.

4.2 ST-EUA Complexity Analysis

4.2.1 ST-EUA Hardness

Based on the LGP model built in Section 4, we now prove that an ST-EUA problem is $\mathcal{NP}$-hard. First, we introduce the Knapsack problem, which is a classic $\mathcal{NP}$-hard problem.

Definition 5. *(Knapsack Problem) Given n items and their corresponding weights and values, the aim of a Knapsack problem is to select a group of items so that the overall values of the items are the highest within the total weight constraint W. It can be formally modeled as follows:*

$$\max \sum_{i=1}^{n} v_i x_i \tag{14}$$

s.t.:

$$\sum_{i=1}^{n} w_i x_i \leq W \tag{15}$$

$$x_i \in \{0, 1\} \tag{16}$$

where v_i and w_i are the value and the weight of item i, respectively, W is the total weight constraint on all the selected items and x_i is a binary variable indicating whether the i-th item is selected.

Theorem 1. *The $\mathcal{NP}$-hard Knapsack is reducible from the ST-EUA problem, i.e., $\leq_p$ ST-EUA. Thus, the ST-EUA problem is $\mathcal{NP}$-hard.*

Proof: Based on the definition of the Knapsack problem, we now prove that an ST-EUA problem is $\mathcal{NP}$-hard by reducing a Knapsack problem to a specialization of an ST-EUA problem.

Based on Equations (2) (3) (4) (5), the QoE of a user u_i allocated to an edge server s_j can be calculated as follows:

$$E_i = \frac{W_i^t}{\sum_{a_k \in A(u_i)} w_k} (\frac{\xi}{d_{ij}})^2 \frac{L}{1 + e^{-\alpha(x_i - \beta)}} \tag{17}$$

To do the proof, we make the following assumptions to constitute a special instance of the ST-EUA problem:

1. In a time slot t, each user u_i has a service request $A_t(u_i)$. For each task $a_k \in A_t(u_i)$, its requirements for different types of computing resources are the same, i.e., $w_k^1 = w_k^2 = \cdots = w_k^d$. Thus, the ratio of the resources available to a user to the total resources required by the user $\frac{W_i^t}{\sum_{a_k \in A_t(u_i)} w_k}$ in (17) is transformed into the ratio of the number of allocated tasks to the total number of tasks requested by a user.
2. The coverage of each edge server $s_j \in S$ is infinite, i.e., a task can be allocated to any of the edge servers in the area. In this way, the distance between a user and an edge server can be omitted in an ST-EUA problem. That is, the attenuation coefficient of a user's signal $(\frac{\xi}{d_{ij}})^2$ is reduced to a constant.
3. For any edge server s_j, its capacities in different resource dimensions are equal, i.e., $c_j^1 = c_j^2 = \cdots = c_j^d$.
4. In a time slot t, the migration cost $l_{a_k}^t(u_i)$ of any user u_i's task a_k is omitted. That is, the ST-EUA problem aims to maximize the users' overall QoE without considering the migration cost.

For the simplified special case, objective (8) can be projected to objective (14). Constraint (11) can be projected to constraint (16), because a task can be allocated to an edge server or it cannot be satisfied by any edge server. Moreover, constraint (10) can be projected to constraint (15), since the computing capacities of all the edge servers can be aggregated to obtain an overall resource upper bound. Consequently, there is a solution to the ST-EUA problem if and only if there is a solution to a corresponding knapsack problem. Thus, an ST-EUA problem is $\mathcal{NP}$-hard.

4.2.2 ST-EUA Challenges

Based on the definition of ST-EUA problem, it aims to assign edge servers for satisfying users' task requests in a specific area across multiple time slices, maximizing overall users' satisfaction and minimizing services' migration cost.

The main challenges are threefold. First, the allocation of edge users in each time slot is highly affected by previous multiple time slots. Over time, users may move across multiple edge servers, leading to service interruptions and QoE degradation. Service migration across edge servers incurs energy, bandwidth and transportation costs. To improve users' overall QoE, migration cost optimization is a key factor across multiple time slots that must be considered during the process of edge user allocation. Second, how to match feasible edge servers with tasks effectively to achieve high user satisfaction is a challenging issue. Each of these users may submit multiple task requests in a time slot. Since edge servers' computing capacities are constrained, they

Algorithm 1 : Genetic Algorithm-based Spatio-temporal user Allocation (GA-ST)

Input: An ST-EUA problem $\langle T, U_t, S, D_t, A_t, W_t \rangle$.
Output: An approximate solution Θ.
1: Set the parameters of $N_{pop}, N_{it}, P_{cr}, P_{mu}$ and N_{mu}
2: Initialize N_{pop} chromosomes as the solutions to the ST-EUA problem by heuristic information, $H = \{H_1, H_2, \ldots, H_{N_{pop}}\}$
3: Calculate the fitness score of each $H_i \in H$ by QoE and migration cost:
$f_{qoe}(H_i) = \sum_{j}^{|U_t|} E_j$
$f_{mc}(H_i) = \sum_{j=1}^{|U_t|} \sum_{k=1}^{|A_t(u_j)|} r^t_{a_k}(u_j) l^t_{a_k}(u_j)$
4: $H_{best} \leftarrow \underset{H_i \in H}{argmax}\, f_{qoe}(H_i), \underset{H_i \in H}{argmin}\, f_{mc}(H_i)$
5: **for** $i = 1 \rightarrow N_{it}$ **do**
6: Create an empty set H′ to save the populations of the next generation
7: **for** $j = 0 \rightarrow \frac{N_{pop}}{2}$ **do**
8: Select two chromosomes H_u,H_v by the fitness scores of user allocation strategy
9: **if** rand$(0,1)$[1] $\leq P_{cr}$ **then**
10: Crossover: $H_u \rightarrow H'_u$; $H_v \rightarrow H'_v$,
11: **if** rand(0,1) $\leq P_{mu}$ **then**
12: Mutation: randomly select N_{mu} loci to replace genes with candidate servers by distance-aware mutation probability p_k
13: **end if**
14: Put H'_u, H'_v to H'
15: **else**
16: Put H_u, H_v to H'
17: **end if**
18: **end for**
19: Calculate the fitness score of each $H'_i \in H'$ by QoE and migration cost:
$f_{qoe}(H'_i) = \sum_{j}^{|U_t|} E_j$
$f_{mc}(H'_i) = \sum_{j=1}^{|U_t|} \sum_{k=1}^{|A_t(u_j)|} r^t_{a_k}(u_j) l^t_{a_k}(u_j)$
20: $H_{best} \leftarrow \underset{H'_i \in H'}{argmax}\, f_{qoe}(H'_i), \underset{H'_i \in H'}{argmin}\, f_{mc}(H'_i)$
21: $H' \rightarrow H$
22: **end for**
23: **return** $\Theta = \{\theta_1, \theta_2, ..., \theta_p\}$ decoded from H_{best}

may not be able to process all the incoming service requests from within their coverage areas. Fortunately, a user may be covered by multiple edge servers and its tasks can be processed by any of these edge servers with an adequate processing capacity. Thus, a proper allocation strategy is needed to allocate service requests to appropriate edge servers. Third, the ST-EUA problem exhibits both temporal and spatial features. We must consider user mobility and service requests with task decomposition over time, as well as the distances between edge users and edge servers. Finding the optimal solution that fulfil all the constraints is not trivial, especially in large-scale scenarios.

4.3 Approximate Algorithm of ST-EUA

4.3.1 Overview of GA-ST Algorithm

The genetic algorithm (GA) encodes specific problems in a chromosome-like structure and simulates the evolution process by applying a fitness function to judge whether a chromosome is predominant. However, there are two issues when the traditional GA is directly employed to solve an ST-EUA problem, i.e., inappropriate fitness function and lack of heuristic information in a specific application scenario. To ensure the effectiveness of GA in solving the ST-EUA problem, we make the following improvements. First, we leverage the capacities of edge servers and the resource requirements of tasks when initializing the population to obtain better offspring. Second, we choose superior individuals to perform crossover and mutation operations in each iteration based on dual fitness functions, which balances the two optimization goals of overall users' satisfaction and services' migration cost. Finally, the distances between edge users and edge servers are applied as heuristic information in a mutation operation to find a suitable edge server for the replacement of a selected locus.

The pseudo code of GA-ST is presented in Algorithm 1. It begins with parameter settings (line 1), including the size of initial populations N_{pop}, the maximum iteration times N_{it}, the probability of crossover P_{cr} and mutation P_{mu}, and the mutation times at each iteration N_{mu}. Then, the initial chromosomes are generated by leveraging the heuristic information and put into the chromosome set H (line 2). Afterwards, we calculate the fitness score of each chromosome in H (line 3), and the chromosome with the best fitness score is selected as the optimal one (line 4). Subsequently, the pivotal iteration steps of generation update are performed (lines 7-18), including the operations of selection (line 8), crossover (lines 9-10) and mutation (lines 11-12). At the end of the iteration, the chromosomes for the next generation are generated according to the different situations (lines 14,16). After each round of generation update, the algorithm recalculates the fitness score of each chromosome and updates the best one (lines 19-20). Finally, Θ is returned as the approximate solution to an ST-EUA problem (line 23).

4.3.2 Generation Update of GA-ST

In GA-ST, a feasible solution to the ST-EUA problem is encoded as a chromosome. Each chromosome is comprised of a set of independent genes used to represent edge servers. The locus of a gene in a chromosome represents a task of a user. The steps of generation update in GA-ST are elaborated as follows.

Initialization of GA-ST: At the beginning, GA-ST generates N_{pop} chromosomes as the individuals of the initial populations. Instead of randomly generating a chromosome as a solution to the EUA problem, $H_i = \{s_1, s_2, \ldots, s_q\}$, heuristic information including the resources required for each task and edge servers' remaining capacities are taken into account to configure a set of more effective populations.

Fitness Evaluation of GA-ST: For a chromosome H_i, we first calculate the QoE for each user according to (2)-(5). Then, fitness $f_{qoe}(H_i)$ is measured as (18). Secondly, we calculate the migration cost for each user according to (6)(7), and the fitness $f_{mc}(H_i)$ is measured as in (19).

$$f_{qoe}(H_i) = \sum_{j=1}^{|U_t|} E_j \tag{18}$$

$$f_{mc}(H_i) = \sum_{j=1}^{|U_t|} \sum_{k=1}^{|A_t(u_j)|} r^t_{a_k}(u_j) l^t_{a_k}(u_j) \tag{19}$$

Selection of GA-ST: By leveraging the fitness of chromosomes, we choose those superior solutions to the ST-

[1] rand(0,1) generates a random real number ranged in [0,1].

EUA problem, while eliminating inferior ones. In GA-ST, the objective of selection is to transmit the chromosomes with high fitness scores directly or indirectly to the next generation by crossover and mutation operations. Here, we adopt the well-known *roulette wheel* method to perform the selection operation. The probability of a chromosome H_i with fitness $f_{qoe}(H_i)$ and $f_{mc}(H_i)$ to be selected out of the populations H are calculated as in (29) and (21), respectively.

$$P_{qoe}(H_i) = \frac{f_{qoe}(H_i)}{\sum_{j=1}^{|H|} f_{qoe}(H_j)} \tag{20}$$

$$P_{mc}(H_i) = \frac{exp(-f_{mc}(H_i))}{\sum_{j=1}^{|H|} exp(-f_{mc}(H_j))} \tag{21}$$

In the ST-EUA problem, we first select chromosomes according to P_{qoe}, because the users' overall QoE has much higher priority than migration cost in an ST-EUA problem. If $P_{qoe}(H_i) = P_{qoe}(H_j)$, we select chromosomes according to P_{mc}. Since we select two parent chromosomes to generate two children each time, the selection operation at each iteration is performed $N_{pop}/2$ times to generate the number of chromosomes in GA-ST.

Crossover of GA-ST: Given two selected chromosomes, they are reciprocally crossed to generate newly two children ones, aiming at improving the fitness scores of user allocation in the next generation. GA-ST adopts the two points crossover operation to exchange the partially allocated edge servers of two solutions to the ST-EUA problem. Let H_u and H_v be the two chromosomes selected. Given two points i and j, the results of crossover are obtained as in (22).

$$\begin{aligned} H'_u &= [H_{u(1:i-1)}, H_{v(i:j)}, H_{u(j+1:q)}] \\ H'_v &= [H_{v(1:i-1)}, H_{u(i:j)}, H_{v(j+1:q)}] \end{aligned} \tag{22}$$

where $H_{u(i:j)}$ intercepts the chromosome from the i-th to the j-th allocated edge servers of an ST-EUA solution vector encoded in H, and [ ] concatenates a set of vectors in order. Through the crossover operation, the advantageous parts of user allocation strategies encoded in the parent chromosomes may be merged into the children ones.

Mutation of GA-ST: It is a crucial for GA to avoid early convergence. There are many conventional mutation operators such as Uniform and Gaussian. However, they have not taken full advantage of prior knowledge as heuristic information to improve the mutation operation, which influences the performance of GA-ST. In an ST-EUA problem, the distance between an edge user and an edge server impacts the user's data transmission rate, and consequently its QoE. As illustrated in Fig. 2, a user's signal strength is attenuated by the increasing distance from the edge server. Based on the prior knowledge of signal strength, distance-aware mutation operation is applied to GA-ST. It will increase the users' overall QoE across multiple time slots.

Fig. 3 illustrates the process of the improved distance-aware mutation operation with three steps. Step 1 is the selection of mutation locations, where N_{mu} loci are randomly chosen from a chromosome of user-task-server allocation strategy. Step 2 is the calculation of distance and probability. When a locus is selected, we search for candidate edge servers according to the corresponding user u_j. For each edge server, we calculate its distance from the user u_j.

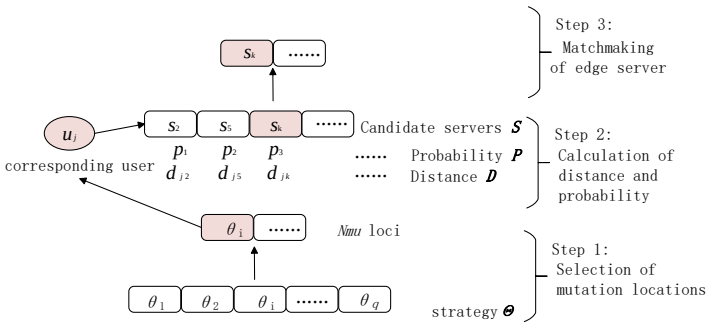


Fig. 3. The process of distance-aware mutation operation for ST-EUA problem.

Based on the results, we can estimate the probability of each edge server being selected. Step 3 is the matchmaking of edge server. It matches edge server s_k that is most likely to replace locus θ_i.

Specifically, given a solution to an ST-EUA problem $\Theta = \{\theta_1, \theta_2, \dots, \theta_q\}$, let us assume that a gene θ_i is chosen for mutation and a user u_j corresponds to the task at θ_i. On this condition, the candidate edge servers of u_j that meet the capacity and proximity constraints are $S(u_j) = \{s_1, s_2, \dots\}$, each of which is a feasibly mutated gene. By using $S(u_j)$, the distance set between u_j and candidate edge servers can be calculated as $D_t(u_j) = \{d_{j1}, d_{j2}, \dots\}$. Based on results, we measure the probability p_k of each candidate edge server s_k to be selected for the mutation operation, which is calculated by applying the softmax function to normalize the additive inverse of distance d_{jk} between u_j and a candidate edge server s_k as in (23).

$$p_k = \frac{exp(-d_{jk})}{\sum_{l=1}^{|S(u_j)|} exp(-d_{jl})} \tag{23}$$

where p_k denotes the probability of selecting s_k to replace gene θ_i. Thus, with the consideration of distance as heuristic information, the overall average QoE of a solution Θ can be improved after mutation operations.

5 Experiments

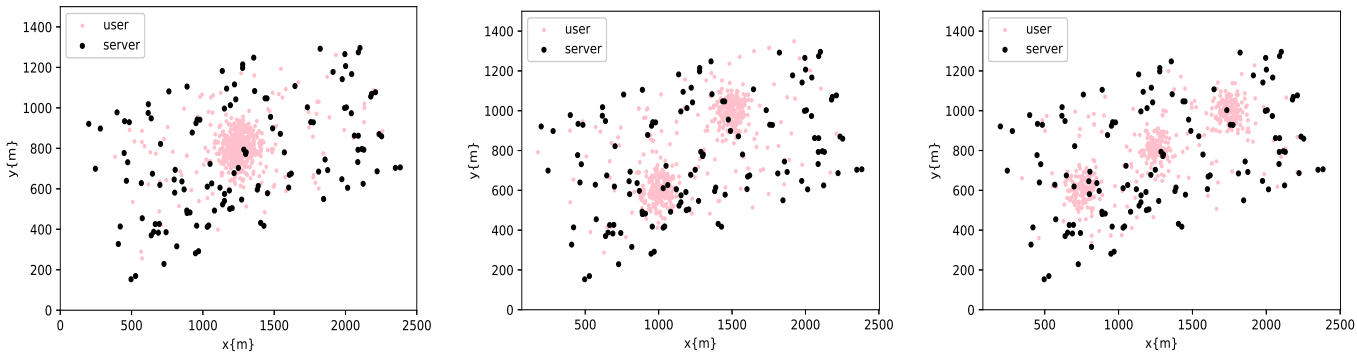
5.1 Experimental Setup and Datasets

To validate the performance of GA-ST, we evaluate its effectiveness and efficiency experimentally. All the experiments are performed on a machine equipped with an Intel(R) Xeon(R) Gold 6130 CPU@2 and 192GB RAM. The TD-EUA-O and TD-EUA-O-D approaches employ the Gurobi Optimizer to solve the ST-EUA problem. GA-ST and its variations are implemented in Python 3.7.4.

The experiments are conducted on two public benchmarking datasets that are widely used in research on edge computing, including the EUA dataset[2] and Shanghai-Telecom dataset[3]. The EUA dataset contains 125 edge servers (base stations) in the Melbourne central business district area in Australia. It has already been extensively applied as experiment datasets in a largy body of existing research [2], [6], [9], [11]. Following the Gaussian distribution $N(u, \sigma)$, 512 users are distributed in different ways in this area to simulate three kinds of real-world ST-EUA

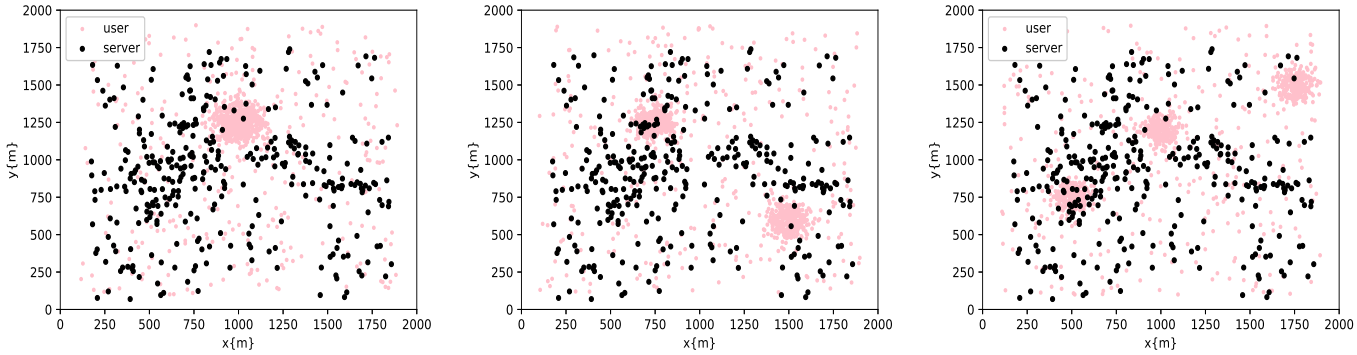
. [2]https://sites.google.com/site/heqiang/eua-respository https://github.com/swinedge/eua-dataset

. [3]http://www.sguangwang.com/TelecomDataset.html



(a) EUA Dataset 1 (b) EUA Dataset 2 (c) EUA Dataset 3

Fig. 4. Different distributions of users and servers on EUA dataset. Where (1),(2),(3) have one hot spot, two hot spot and three hot spot, respectively.



(a) Shanghai-Telecom Dataset 1 (b) Shanghai-Telecom Dataset 2 (c) Shanghai-Telecom Dataset 3

Fig. 5. Different distributions of users and servers on Shanghai-Telecom dataset. Where (1),(2),(3) have one hot spot, two hot spot and three hot spot, respectively.

scenarios with diverse user distributions, as illustrated in Fig. 4, where each black point represents an edge server and each orange point represents a user. The Shanghai-Telecom dataset contains 3,233 base stations within Shanghai, China. In the experiments, we choose those edge servers (base stations) of Lujiazui Finance and trade zone, and then generate users based on a Gaussian distribution $N(u, \sigma)$, where there are also three different distributions with 345 edge servers and 1,000 users, as illustrated in Fig. 5.

5.2 Competing Methods and Evaluation Metrics

To evaluate the performance of GA-ST, we compare it with twelve competing approaches, including four baseline approaches, four state-of-the-art approaches, and four GA-based approaches.

- **Random:** It randomly selects an available edge server to process users' tasks.
- **Greedy-distance:** It calculates the geographical distance between a user and an edge server, and selects the nearest edge server to perform the user's tasks.
- **Greedy-capacity:** It evaluates edge servers' available computing resources, and selects the edge server with the most remaining resources to process tasks.
- **Greedy-task:** It ranks the computing resources required for each task in descending order, then randomly selects an available edge server to process each task sequentially.
- **TD-EUA-O [10]:** It models the ST-EUA problem as a variant of the knapsack problem and finds an optimal solution that maximizes users' overall QoE.
- **TD-EUA-H [10]:** It finds a sub-optimal EUA solution by taking advantage of heuristic information on the computing resources required tasks and edge servers' available capacities.
- **TD-EUA-O-D:** It is a variation of TD-EUA-O, where the distance between a user and an edge server is integrated into the estimation on the QoE, when globally and optimally allocating each task to an appropriate edge server.
- **TD-EUA-H-D:** It is a variation of TD-EUA-H, where the distance between an user and an edge server is integrated into the QoE for finding an approximate solution to a user allocation problem.
- **GA:** This basic GA approach randomly selects edge servers to serve the tasks of users to generate initial populations.
- **GA-distance:** It is a variation of GA, where the Greedy-distance approach is employed to generate user allocation strategies as initial populations.
- **GA-capacity:** It is a variation of GA, where the Greedy-capacity approach is employed to generate user allocation strategies as initial populations.
- **GA-task:** It is a variation of GA, where the Greedy-task approach is used to generate user allocation strategies as initial populations.

In the experiments, we employ four widely-used evaluation metrics to compare and analyze the experiment results, two for effectiveness and two for efficiency.

- **QoE:** It is measured by users' overall QoE produced by the approach.

TABLE 2
Experimental results of user allocation among competing approaches on EUA datasets

Methods	EUA Dataset 1				EUA Dataset 2				EUA Dataset 3			
	QoE	AR	Cost	CPU Time	QoE	AR	Cost	CPU Time	QoE	AR	Cost	CPU Time
Random	1491	0.53	473	0.015	2012	0.65	612	0.018	2465	0.78	552	0.022
Greedy-distance	2459	0.53	136	0.016	3167	0.64	172	0.020	3261	0.78	161	0.023
Greedy-capacity	1335	0.55	425	0.045	1955	0.67	535	0.060	2282	0.81	504	0.069
Greedy-task	1434	0.51	402	0.028	1963	0.62	547	0.033	2247	0.74	552	0.036
TD-EUA-H	1391	0.52	349	0.056	1810	0.65	509	0.083	2270	0.81	556	0.099
TD-EUA-O	1704	0.68	-	6.378	2291	0.82	-	6.742	2589	0.93	-	6.183
TD-EUA-H-D	2731	0.50	155	0.054	3461	0.59	310	0.057	3636	0.72	201	0.066
TD-EUA-O-D	3839	0.56	-	2.386	5001	0.69	-	2.757	5092	0.82	-	3.892
GA	3244	0.65	575	1.163	3804	0.74	698	1.329	4030	0.85	651	1.092
GA-capacity	3094	0.66	560	1.335	3705	0.75	626	1.279	3974	0.87	593	1.140
GA-task	3244	0.62	496	1.634	3772	0.71	586	1.079	3939	0.82	628	1.224
GA-distance	4105	0.65	315	1.110	4732	0.73	342	1.260	4892	0.83	324	1.402
GA-ST	4265	0.62	289	1.176	4980	0.70	425	1.198	5141	0.80	390	1.365

- **Allocation Rate:** It is measured by the percentage of users allocated to edge servers.
- **Migration Cost:** It is measured by the cost of service migration across all time slots.
- **CPU Time:** It is measured by the computation time taken to find a solution.

5.3 Experiment Results and Analyses

In the experiments, the parameters are tuned to achieve the optimal performance for the competing approaches. Specifically, the number of tasks of a user follows a Gaussian distribution with $u = 3$. We set the number of users to 512, and the number of edge servers to 125 in the EUA dataset, while the number of users is set to 1,000 and the number of edge servers to 345 in Shanghai-Telecom dataset. Edge users are generated based on the Gaussian distribution $\sigma = 55$ to simulate hot spots. Moreover, an edge server's available computing capacities follow a Gaussian distribution $N(20, 1)$ in both EUA datasets and Shanghai-Telecom datasets. The best and second-best values in each column are marked in dark and light grey, respectively. Furthermore, regarding the parameter settings of GA-ST, they are determined through iterative experiments to achieve the best experimental performance. Specifically, there are $N_{pop} = 50, N_{it} = 5, P_{cr} = 0.8$ and $P_{mu} = 0.3$. Parameter N_{mu} is related to the total number of task requests q from all users in time slot t, which is set by $q * 0.2$.

Table 3 summarizes the experimental results on four evaluation metrics across three EUA datasets with diverse distributions of users and servers, where we compare GA-ST with twelve competing methods partitioned into two categories, consisting of six baselines and six advanced competing approaches. First, we analyze the comparison between GA-ST and the six baselines, including Random, Greedy-distance, Greedy-capacity, Greedy-task, TD-EUA-H and TD-EUA-O. The results demonstrate that GA-ST achieves the highest overall QoE compared to the baselines. Specifically, GA-ST outperforms Random, Greedy-distance, Greedy-capacity, Greedy-task, TD-EUA-H and TD-EUA-O by 2,774, 1,806, 2,930, 2,831, 2,874 and 2,561 in QoE in EUA Dataset1, respectively. In terms of allocation rate, GA-ST is superior to Random, Greedy-distance, Greedy-capacity, Greedy-task and TD-EUA-H with an advantage of 16.98%, 16.98%, 12.72%, 21.56% and 19.23% on EUA Dataset1, respectively. The main reason for the advantage of GA-ST lies in its consideration of multiple key factors, including temporal features, geographical distance between users and servers, and task decomposition. Note that TD-EUA-O achieves the highest allocation rate, because it pursues global optimization. However, it generates a low QoE and takes the most time to find a solution. Furthermore, it can only solve a static global optimization problem of user allocation. In terms of migration cost, GA-ST achieves the second best performance, while Greedy-distance outperforms all the competing approaches because it always allocates a user to the nearest edge server. With regard to CPU time, TD-EUA-O takes the most time to find a solution, followed GA-ST. Note that TD-EUA-O's time consumption is several times higher than GA-ST's.

Second, we analyze the experimental results across three EUA datasets between GA-ST and the six advanced competing approaches, including TD-EUA-H-D, TD-EUA-O-D, GA, GA-capacity, GA-task and GA-distance. Compared to these approaches, GA-ST achieves the highest user overall QoE on EUA Dataset 1 and EUA Dataset 3, while TD-EUA-O-D obtains slightly higher QoE on EUA Dataset 2. More specifically, GA-ST is superior to the competing approaches with an advantage of 56.16% over TD-EUA-H-D, 11.09% over TD-EUA-O-D, 31.47% over GA, 37.84% over GA-capacity, 31.47% over GA-task and 3.89% over GA-distance in QoE on EUA Dataset 1. Due to the high computational complexity of TD-EUA-O-D, it is intractable in dealing with large-scale ST-EUA problems, where timely decision-making is desired in real-world application scenarios. As for allocation rate, GA-ST outperforms TD-EUA-H-D and TD-EUA-O-D in most cases, and has a similar performance as GA, GA-capacity, GA-task and GA-distance across three EUA Datasets. Note that the QoE achieved by GA-capacity is worse than GA, opposite to allocation rate. The underlying reason is that even though GA-capacity chooses edge servers with the most remaining resources to process tasks in the initial population, it does not consider the distance be-

TABLE 3
Experimental results of user allocation among competing approaches on Shanghai-Telecom datasets

Methods	Shanghai-Telecom Dataset 1				Shanghai-Telecom Dataset 2				Shanghai-Telecom Dataset 3			
	QoE	AR	Cost	CPU Time	QoE	AR	Cost	CPU Time	QoE	AR	Cost	CPU Time
Random	3517	0.68	538	0.039	4937	0.85	552	0.049	5165	0.79	649	0.046
Greedy-distance	6530	0.65	135	0.044	8034	0.81	117	0.055	9234	0.78	185	0.055
Greedy-capacity	3229	0.71	460	0.182	4382	0.87	499	0.274	4361	0.80	456	0.281
Greedy-task	3285	0.66	405	0.071	4674	0.82	480	0.084	5143	0.78	491	0.075
TD-EUA-H	3101	0.71	205	0.238	4329	0.88	296	0.351	4354	0.79	341	0.358
TD-EUA-O	3709	0.83	-	29.717	4859	0.94	-	20.200	4683	0.81	-	8.629
TD-EUA-H-D	6883	0.65	373	0.156	8813	0.81	551	0.218	10120	0.78	426	0.279
TD-EUA-O-D	8578	0.71	-	9.960	11326	0.87	-	14.645	11733	0.80	-	9.750
GA	6683	0.75	675	2.341	7847	0.88	615	3.373	7839	0.85	669	3.025
GA-capacity	6482	0.80	601	3.686	7302	0.91	618	3.314	7231	0.86	648	3.313
GA-task	6631	0.74	603	2.574	7678	0.87	586	2.967	7904	0.83	667	3.197
GA-distance	9215	0.74	350	3.338	10563	0.86	369	3.711	11377	0.84	353	3.813
GA-ST	9472	0.73	332	3.431	11183	0.86	310	3.711	12067	0.83	373	3.819

tween edge users and servers which is of vital importance to QoE. However, GA randomly selects edge servers for users' task requests when initializing the population. Therefore, it may potentially choose nearby edge servers to obtain a higher QoE. With regard to migration cost, TD-EUA-H-D still achieves the best performance followed by GA-ST. The main reason is that it tends to allocate users to their nearest edge servers. In terms of computational cost, TD-EUA-O-D and TD-EUA-H-D take the most and least CPU time to find a solution, respectively, while GA, GA-capacity, GA-task, GA-distance and GA-ST have similar time consumption.

Table 4 summarizes the experimental results on three Shanghai-Telecom datasets, where we compare GA-ST with six baselines and six advanced competing approaches. Among all the Shanghai-Telecom datasets, GA-ST achieves the highest QoE compared to the six baselines. For instance, GA-ST achieve superior QoE with an advantage of 169.32%, 45.05%, 189.23%, 188.34%, 205.41% and 155.37% over Random, Greedy-distance, Greedy-capacity, Greedy-task, TD-EUA-H and TD-EUA-O on Shanghai-Telecom Dataset 1, respectively. As for allocation rate, TD-EUA-O still achieves the highest allocation rate, followed by GA-ST. Similarly, Greedy-distance obtains the best performance in migration cost, although GA-ST can efficiently allocate the tasks to nearby servers. The computational cost has the same fluctuations between Shanghai-Telecom and EUA datasets.

From the above results, we conclude that GA-ST achieves the best balance across multiple evaluation metrics on all the datasets, which is critical in real-world application scenarios.

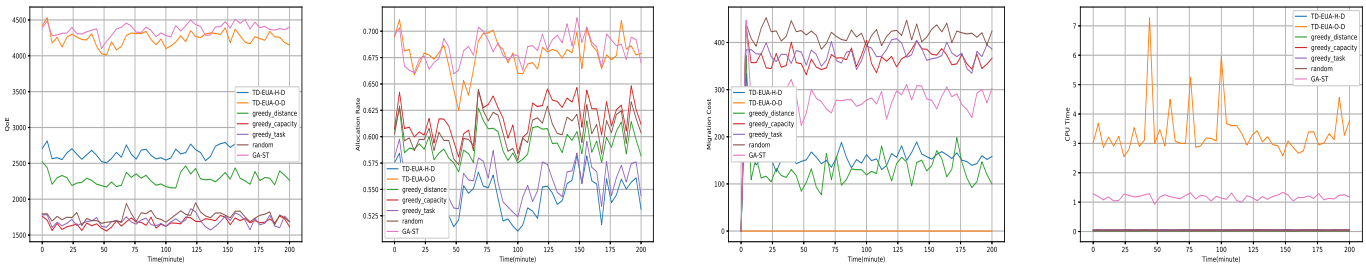
5.4 Performance Impact of Parameters

To evaluate the performance impact among competing approaches, we vary the average capacity of edge servers. In the experiments, we generate users' service requests and moving trajectories within 2 hours, including totally 512 users in EUA Dataset 2. By tuning three different capacity levels of edge servers, we compare the performance impact on four evaluation metrics between GA-ST and six competing approaches, including Random, Greedy-distance, Greedy-capacity, Greedy-task, TD-EUA-O-D and TD-EUA-H-D. The results on 50%, 100% and 150% of edge servers' capacity along with the changes of time are shown in Figs. 5, 6 and 7, respectively.

The subfigure (a) of Figs. 5 $\sim$ 7 compare the QoE achieved by the approaches where edge servers' capacities vary from 50% to 150% in steps of 50%. As the capacity of edge server increases, all competing approaches achieve a better performance on users' QoE in different time slots. More specially, GA-ST outperforms most of competing approaches in maximizing QoE. It is approximately 4,300 in Fig. 5(a), 5,100 in Fig. 6(a) and nearly 7,000 in Fig. 7(a), respectively. It is worth noting that Greedy-task, Random and Greedy-capacity achieve similar performance in maximizing QoE, when the capacity of edge server is 50% or 100%. However, as it increases to 150%, Greedy-task and Random outperform Greedy-capacity in QoE. The underlying reason is that as edge servers' capacity gradually increases, computing resources become increasingly abundant, making it cost-ineffective to allocate a task to an edge server with the most remaining resources.

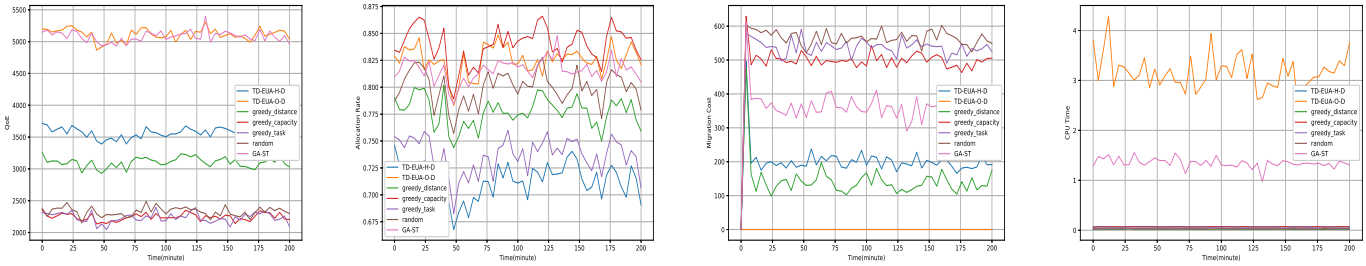
The subfigure (b) of Figs. 5 $\sim$ 7 compare the performance in allocation rate with the changes in edge servers' capacities. Similar to QoE, the allocation rates of all the competing approaches show the same fluctuation trend, as the capacity of edge servers increases. More specifically, when the capacity of edge servers is at a low level, the available computing resources is insufficient for serving users' service requests. As illustrated in Fig. 5(b), it shows that the allocation rates of all competing approaches range from 0.5 to 0.7. With the increase in edge servers' capacity, th allocation rate raises up to the range of 0.65 to 0.875 across the competing approaches in Fig. 6(b). As shown in Fig. 7(b), the allocation rates of all the competing approaches exceed 0.94. Furthermore, Greedy-capacity achieves the highest allocation rate when the capacity of edge servers is 100%. When edge servers' capacities vary, GA-ST always achieves high performance in terms of allocation rate.

The subfigure (c) of Figs. 5 $\sim$ 7 compare the performance in migration cost along with the changes in edge servers' capacities. As demonstrated, it follows a similar trend as QoE and allocation rate. As the capacity increases, it has



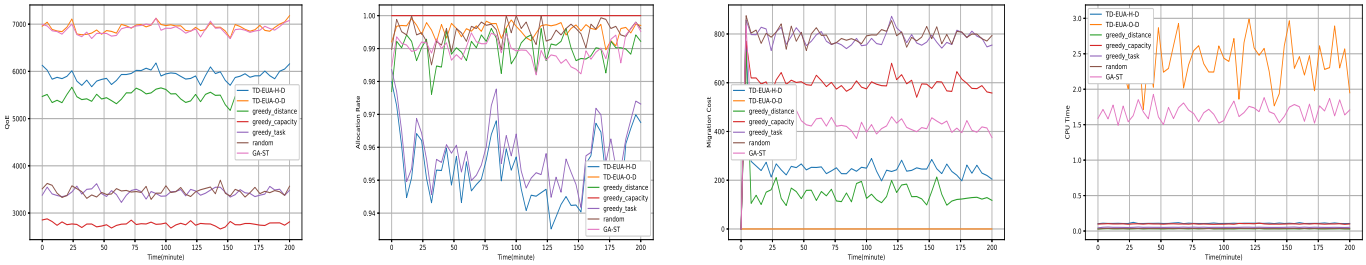
(a) QoE (b) Allocation Rate (c) Migration Cost (d) CPU Time

Fig. 6. Performance comparisons on 50% edge servers' capacity among GA-ST and competing approaches.



(a) QoE (b) Allocation Rate (c) Migration Cost (d) CPU Time

Fig. 7. Performance comparisons on 100% edge servers' capacity among GA-ST and competing approaches.



(a) QoE (b) Allocation Rate (c) Migration Cost (d) CPU Time

Fig. 8. Performance comparisons on 150% edge servers' capacity among GA-ST and competing approaches.

an upward trend in terms of migration cost. The reason is that the complexity of th ST-EUA problem increases with the increase in the capacity of edge servers, producing more possible solutions. It is worth noting that the migration cost of TD-EUA-O-D reaches the highest and is not evaluated, since it needs to reallocate all the users in each time slot t. As for GA-ST, it achieves the third best performance after Greedy-distance and TD-EUA-H-D with the change in edge servers' capacities.

The subfigure (d) of Figs. 5 $\sim$ 7 compare the performance in CPU time along with the changes in edge servers' capacities. On the contrary, as the capacity of edge servers increases, the CPU times of all the competing approaches decline. As shown in the results at different capacity levels, the computation time of GA-ST fluctuates slightly and it takes significantly less time consumption than TD-EUA-O-D. As for TD-EUA-H-D, Greedy-distance, Greedy-capacity, Greedy-task and Random, their CPU times always remain at a low level.

5.5 Threats to Validity

To our best knowledge, it is the first attempt to tackle the spatio-temporal EUA problem with task decomposition. When verifying the performance of GA-ST, we compare it with a random baseline, three greedy-based approaches, four state-of-the-art approaches and four GA-based approaches. Some of these approaches are not specifically designed to solve the ST-EUA problem as GA-ST. As a result, GA-ST tends to outperform these approaches, which threatens the construct validity. In addition, there is no real-world dataset that describes user distribution, user mobility path, and edge server distribution accurately at the same time. To minimize this external threat, we conducted the experiments on two widely-used datasets, i.e., the EUA dataset and the Shanghai-Telecom dataset with three different types of user distributions.

6 Discussion

6.1 Relation to Service Oriented Computing

Service Oriented Computing (SOC) is a computing paradigm that utilizes services as fundamental components for developing and integrating real-world applications. The core of SOC is how to use services effectively and efficiently to facilitate service-oriented system development. Researchers have carried out investigations on SOC and made a lot of prominent contributions to service discovery, selection, composition, recommendation, mashup creation, QoS management, and verification [19].

With the development of network technology, the Internet has become a huge resource repository for global information transmission and sharing. In recent years, more

and more web services have been published over the Internet, which significantly increases the burden on energy consumption, network congestion and security risks. Edge computing, as a new paradigm, has recently emerged to address these issues. However, it raises many new and critical challenges, e.g., edge user allocation, computing offloading and edge data caching.

Although service-oriented computing and edge computing are different in nature, they share some similarities. More specifically, SOC techniques can be applied to support the research on edge computing, such as the application of service discovery and selection in computing offloading. Simultaneously, edge computing can support diverse applications that promote the development of SOC and raise new research challenges. For example, in the edge environment, QoS prediction can use a federated learning framework to protect users' privacy and obtain more accurate prediction performance.

6.2 Optimization Techniques in Traditional Architectures

Optimization techniques have been extensively investigated for task allocation in the fields of traditional architectures, including distributed and grid computing, and cloud computing. With the popularity of edge computing, it is rapidly receiving many attentions from researchers to solve research challenges in edge computing, such as edge user allocation. Since these computing paradigms have their own characteristics, they focus on different application requirements and optimization objectives for task allocation.

Distributed and grid optimization. Grid computing exploits a group of networked computers that collaborate as a virtual supercomputer to handle data-intensive computing tasks. Generally, it uses specifically middleware to decompose a huge scientific computing task into a bunch of independent and logically related subtasks, which are then processed and fused by corresponding nodes with sufficient computing capacities. When some computing nodes crash, the computing task will not fail, because its subtasks can be reassigned to other computing nodes for processing. Therefore, job scheduling optimization and subtask result aggregation are the main optimization challenges in grid computing [20], [21].

Cloud optimization. Compared with grid computing, cloud computing offers computing and storage resources in a much more flexible manner. Service vendors can hire virtual machines (VMs) for hosting their services in the cloud without having to worry about the maintenance of the underlying physical machines. From their perspective, task allocation optimization focuses on scheduling various tasks across heterogeneous cloud nodes. There may be dependencies among these tasks, or deadlines. In this context, task allocation optimization often aims to maximize system throughput, minimize task execution time, etc [22], [23].

Edge optimization. Edge computing extends cloud computing by deploying edge servers at base stations. Aiming to ensure low service latency for users, task allocation in the edge computing environment often focuses on service latency minimization, cost minimization, etc. The edge computing environment exhibits two new characteristics compared with the cloud computing environment, i.e., proximity constraint and capacity constraint [4]. Edge servers often need to collaborative to overcome the capacity constraint [24], [25]. The optimization techniques designed for solving the problem in the MEC environment must consider these unique characteristics.

7 Related Work

With the advances in mobile devices and the Internet of Things, cloud centers may easily be overwhelmed by excessive workloads, causing network latency and congestion. The emergence of edge computing overcomes the major drawback of service latency in cloud computing. As a new computing paradigm, edge computing integrates network, computing, storage, and application core capabilities for deployment and invocation of low-latency edge services close to end users. Services deployed on edge servers can provide fast responses to nearby users' requests. However, an edge server has a limited resource capacity, making it difficult or sometimes impossible to serve all the users within its coverage area. Offering many new opportunities, edge computing has raised a variety of new research challenges, e.g., edge user allocation (EUA) [2], [4], [6], [26], [27], computing offloading [28], [29], edge service placement [30], [31], [32], edge data management [24], [33], [34], [35], edge resource management [36], [37] and edge server placement [38], [39]. Specially, there are somewhat similarities between computing offloading and edge user allocation, where they both aim at executing the computation-intensive and latency sensitive mobile tasks on edge servers. However, they differ significantly in (but are not limited to) the following four main ways. First, computation offloading is studied from the edge infrastructure provider's perspective, e.g., Amazon and Vodafone, while edge user allocation is studied from the service vendor's perspective, e.g., YouTube and Uber. These are different stakeholders in the edge computing environment and have different interests and concerns. Second, computation offloading allocates tasks coming from all the users as generic computation tasks, while edge user allocation allocates users to edge servers so that they can be served with low service latency. Third, computation offloading assumes that a task can be processed by any edge server. Edge user allocation assumes that a service vendor must hire resources like CPUs, RAMs and storage to serve their users. Last but not least, computation offloading usually do not consider the costs incurred. Edge user allocation considers the cost of hiring resources on edge servers for serving users.

Recently, EUA problem has attracted a lot of attentions from academia and industry. Lai et al. [2] made the first attempt to tackle the EUA problem. They modeled the EUA problem as a variable sized vector bin packing problem, and solved it with the objective to allocate the most users to the fewest edge servers. Subsequently, user satisfaction was further applied as the criterion to measure whether user allocation is cost-effective, considering that users' requests on computing resources may be differentiated [6]. Peng et al. [11] targeted at mobile edge computing and modeled the EUA problem as a revolvable process. They proposed a mobility-aware greedy algorithm to find an approximate solution to effectively allocating users to their nearby servers with the consideration of user mobility. In our previous

work [9], we made the first attempt to study the distance-aware EUA problem. We considered the correlation between users' signal strength and their distances from edge servers.

Currently, existing studies have simply assumed that a user's service request on computing resources can either be fully fulfilled by a single edge server or cannot be satisfied at all. What is more, existing studies have neglected the complexity of wireless signal transmission and treated the EUA problem as a static global optimization. More importantly, users' service requests vary over time with temporal features that have not been fully considered by existing approaches and have yet to be properly explored and leveraged.

8 Conclusion and Future Work

In this paper, we investigated the problem of spatio-temporal edge user allocation (ST-EUA) with task decomposition. First, we formally formulated the ST-EUA problem by integrating spatio-temporal features and task decomposition, modelled it as a constrained optimization problem and proved its $\mathcal{NP}$-hardness. To solve the ST-EUA problem effectively and efficiently, we proposed a novel genetic algorithm-based heuristic approach GA-ST for finding an approximate solution that aims at maximizing users' overall QoE. Extensive experiments were conducted on two real-world datasets with different distributions of users and edge servers to evaluate the performance of GA-ST. The results demonstrate our proposed approach GA-ST outperforms 12 representative competing approaches and achieves a proper trade-off across multiple evaluation metrics.

In future work, we plan to explore the relationships among different edge servers by their communication paths to further boost the performance of edge user allocation.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61772128, 62172088), and Shanghai Natural Science Foundation (No. 21ZR1400400).

References

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

[2] 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 *16th International Conference on Service-Oriented Computing (ICSOC)*, 2018, pp. 230–245.

[3] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[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] P. Lai, Q. He, J. Grundy, and F. e. Chen, "Cost-effective app user allocation in an edge computing environment," *IEEE Transactions on Cloud Computing*, 2020.

[6] 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 *17th International Conference on Service-Oriented Computing (ICSOC)*, 2019, pp. 86–101.

[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 *40th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2020.

[8] 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 *2020 IEEE International Conference on Web Services (ICWS)*, 2020, pp. 294–301.

[9] Z. Xu, G. Zou, X. Xia, Y. Liu, Y. Gan, B. Zhang, and Q. He, "Distance-aware edge user allocation with QoE optimization," in *27th IEEE International Conference on Web Services (ICWS)*, 2020, pp. 66–74.

[10] G. Zou, Y. Liu, Z. Qin, J. Chen, Y. Gan, B. Zhang, and Q. He, "Td-eua: Task-decomposable edge user allocation with QoE optimization," in *18th International Conference on Service-Oriented Computing (ICSOC)*, 2020.

[11] 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 *26th IEEE International Conference on Web Services (ICWS)*, 2019, pp. 91–98.

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

[13] 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.

[14] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, 2016.

[15] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, and M. Satyanarayanan, "Adaptive VM handoff across cloudlets," *Technical Report CMU-CS-15-113*, 2015.

[16] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2016.

[17] 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.

[18] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2019, pp. 10–18.

[19] Y. Wei and M. B. Blake, "Service-oriented computing and cloud computing: Challenges and opportunities," *IEEE Internet Computing*, vol. 14, no. 6, pp. 72–75, 2010.

[20] E. N. Alkhanak, S. P. Lee, R. Rezaei, and R. M. Parizi, "Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues," *Journal of Systems and Software*, vol. 113, pp. 1–26, 2016.

[21] K. Eng, A. Muhammed, M. A. Mohamed, and S. Hasan, "A hybrid heuristic of variable neighbourhood descent and great deluge algorithm for efficient task scheduling in grid computing," *European Journal of Operational Research*, vol. 284, no. 1, pp. 75–86, 2020.

[22] M. Abd Elaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling iot tasks in cloud-fog computing environments," *Future Generation Computer Systems*, 2021. [Online]. Available: https://doi.org/10.1016/j.future.2021.05.026

[23] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Generation Computer Systems*, vol. 108, pp. 361–371, 2020.

[24] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Cost-effective app data distribution in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 31–44, 2021.

[25] Q. He, C. Wang, G. Cui, B. Li, R. Zhou, Q. Zhou, Y. Xiang, H. Jin, and Y. Yang, "A game-theoretical approach for mitigating edge ddos attack," *IEEE Transactions on Dependable and Secure Computing*, 2021. [Online]. Available: http://dx.doi.org/10.1109/TDSC.2021.3055559

[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*, 2020.

[27] 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.

[28] X. Xu, Q. Wu, L. Qi, W. Dou, S.-B. Tsai, and M. Z. A. Bhuiyan, "Trust-aware service offloading for video surveillance in edge

computing enabled internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1787–1796, 2020.
[29] M. Dai, Z. Su, Q. Xu, and N. Zhang, "Vehicle assisted computing offloading for unmanned aerial vehicles in smart city," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1932–1944, 2021.
[30] Y. Chen, S. Deng, H. Ma, and J. Yin, "Deploying data-intensive applications with multiple services components on edge," *Mobile Networks and Applications*, vol. 25, no. 2, pp. 426–441, 2020.
[31] Z. Xiang, S. Deng, J. Taheri, and A. Zomaya, "Dynamical service deployment and replacement in resource-constrained edges," *Mobile Networks and Applications*, vol. 25, no. 2, pp. 674–689, 2020.
[32] B. Li, Q. He, G. Cui, X. Xia, F. Chen, H. Jin, and Y. Yang, "Read: Robustness-oriented edge application deployment in edge computing environment," *IEEE Transactions on Services Computing*, 2020. [Online]. Available: http://dx.doi.org/10.1109/TSC.2020.3015316
[33] 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 *17th International Conference on Service-Oriented Computing (ICSOC)*, 2019, pp. 477–493.
[34] X. Xia, F. Chen, Q. He, and J. e. a. Grundy, "Online collaborative data caching in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 281–294, 2020.
[35] X. Xia, F. Chen, J. Grundy, M. Abdelrazek, H. Jin, and Q. He, "Constrained app data caching over edge server graphs in edge computing environment," *IEEE Transactions on Services Computing*, 2021. [Online]. Available: http://dx.doi.org/10.1109/TSC.2021.3062017
[36] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A framework for edge node resource management," *IEEE Transactions on Services Computing*, 2017.
[37] M. Zakarya, L. Gillam, H. Ali, I. Rahman, K. Salah, R. Khan, O. Rana, and R. Buyya, "epcAware: a game-based, energy, performance and cost efficient resource management technique for multi-access edge computing," *IEEE Transactions on Services Computing*, 2020.
[38] G. Cui, Q. He, F. Chen, H. Jin, and Y. Yang, "Trading off between user coverage and network robustness for edge server placement," *IEEE Transactions on Cloud Computing*, 2020.
[39] G. Cui, Q. He, X. Xia, F. Chen, H. Jin, and Y. Yang, "Robustness-oriented k edge server placement," in *20th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2020, pp. 81–90.

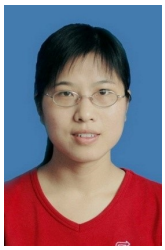
Guobing Zou is an Associate Professor and Dean of the Department of Computer Science and Technology, Shanghai University, China. He received his PhD in Computer Science from Tongji University, Shanghai, China, 2012. He has worked as a Visiting Scholar in the Department of Computer Science and Engineering at Washington University in St. Louis from 2009 to 2011, USA. His current research interests mainly focus on services computing, edge computing, data mining and intelligent algorithms, recommender systems. He has published more than 80 papers on premier international journals and conferences, including IEEE Transactions on Services Computing, IEEE Transactions on Network and Service Management, IEEE International Conference on Web Services and International Conference on Service-Oriented Computing.

Ya Liu is currently a master student in the School of Computer Engineering and Science, Shanghai University, China. Before that, she received a Bachelor degree in Computer Science and Technology at Jiangxi University of Finance and Economics, 2019. Her research interests include edge computing, QoS management, and heuristic algorithms. As the key member, she led a research and development group to successfully design and implement a service-oriented big data analysis and visualization platform that has widely applied in environmental protection agency. She has published two papers on the 18th International Conference on Service-Oriented Computing (ICSOC) and the 27th IEEE International Conference on Web Services (ICWS), respectively.

Zhen Qin is currently a master student in the School of Computer Engineering and Science, Shanghai University, China. Before that, he received a Bachelor degree in Computer Science and Technology at Shanghai University, 2018. He will pursue his Ph.D. degree at Zhejiang University, China. His research interests include Web service clustering, deep learning, and intelligent algorithms. He has published four papers on IEEE Transactions on Services Computing, Knowledge-Based Systems, the 18th International Conference on Service-Oriented Computing (ICSOC), and the 26th IEEE International Conference on Web Services (ICWS).

Jin Chen is currently a master student in the School of Computer Engineering and Science, Shanghai University, China. Before that, he received a Bachelor degree in Computer Science and Technology at Fujian Normal University, 2018. His research interests include Quality of Service, and deep learning. He has participated in a research and development group to successfully design and implement a service-oriented big data analysis and visualization platform. He has published two papers on IEEE Transactions on Network and Service Management, the 18th International Conference on Service-Oriented Computing (ICSOC).

Zhiwei Xu is currently an undergraduate student in the School of Computer Engineering and Science, Shanghai University, China. He will pursue his master degree in the School of Software at Tsinghua University, 2021. His research interests include edge services computing, online social networks, and data mining. He has published two papers on the 27th IEEE International Conference on Web Services (ICWS) and the 18th International Conference on Service-Oriented Computing (ICSOC), respectively.



Yanglan Gan is an Associate Professor in the school of Computer Science and Technology, Donghua University, Shanghai, China. She received her PhD in Computer Science from Tongji University, Shanghai, China, 2012. Her research interests include bioinformatics, service computing, and data mining. She has published more than 50 papers on premier international journals, including Bioinformatics, BMC Bioinformatics, IEEE/ACM Transactions on Computational Biology and Bioinformatics, IEEE Transactions on Services Computing, and IEEE Transactions on Network and Service Management.

Bofeng Zhang is a full professor in the School of Computer Engineering and Science at Shanghai University. He received his Ph.D. degree from the Northwestern Polytechnic University ($\mathcal{NPU}$) in 1997, China. He experienced a Postdoctoral Research at Zhejiang University from 1997 to 1999, China. He worked as a visiting professor at the University of Aizu from 2006 to 2007, Japan. He worked as a visiting scholar at Purdue University from 2013 to 2014, US. His research interests include service recommendation, intelligent human-computer interaction, and data mining. He has published more than 150 papers on international journals and conferences.

Qiang He received the first PhD degree from Swinburne University of Technology (SUT), Australia, in 2009 and the second PhD degree in computer science and engineering from Huazhong University of Science and Technology (HUST), China, in 2010. He is currently working as an Associate Professor in the Department of Computer Science and Software Engineering, Swinburne University of Technology, Australia. His research interests include mobile edge computing, software engineering, cloud computing, services computing, big data analytics, and green computing. More details about his research can be found at https://sites.google.com/site/heqiang/.