LOCUS: User-Perceived Delay-Aware Service Placement and User Allocation in MEC Environment

Yu Chen, Sheng Zhang, *Member, IEEE*, Yibo Jin, *Member, IEEE*, Zhuzhong Qian, *Member, IEEE*, Mingjun Xiao, *Member, IEEE*, Jidong Ge, *Member, IEEE*, and Sanglu Lu, *Member, IEEE*

Abstract—In the multi-access edge computing environment, app vendors deploy their services and applications at the network edges, and edge users offload their computation tasks to edge servers. We study the user-perceived delay-aware service placement and user-allocation problem in edge environment. We model the MEC-enabled network, where the user-perceived delay consists of computing delay and transmission delay. The total cost in the offloading system is defined as the sum of service placement, edge server usage and energy consumption cost, and we need to minimize the total cost by determining the overall service-placing decision and user-allocation decision, while guaranteeing that the user-perceived delay requirement of each user is fulfilled. Our considered problem is formulated as a Mixed Integer Linear Programming problem, and we prove its NP-hardness. Due to the intractability of the considered problem, we propose a LOCal-search based algorithm for USer-perceived delay-aware service placement and user-allocation in edge environment, named LOCUS, which starts with a feasible solution and then repeatedly reduces the total cost by performing local-search steps. After that, we analyze the time complexity of LOCUS and prove that it achieves provable guaranteed performance. Finally, we compare LOCUS with other existing methods and show its good performance through experiments.

Index Terms—User-perceived delay, service placement, user allocation, edge computing, local search

1 Introduction

In Past years, we are witnessing the explosive growth of mobile and IoT devices, such as smartphones, wearable devices, self-driving vehicles, etc. Our daily life is exposed to a rich variety of services and applications, some of which are delay-sensitive and require low latency. Traditionally, the widely used cloud computing technology provides centralized service support for the applications, and computation tasks are offloaded to application vendors' servers in the cloud [1], [2]. However, the centralization of services leads to a long distance between users and clouds, which tends to increase the end-to-end latency. Thus, the existing cloud computing paradigm cannot satisfy the stringent timeliness requirements of the delay-sensitive applications.

- *Yu Chen, Sheng Zhang, Yibo Jin, Zhuzhong Qian, and Sanglu Lu are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: nju_chenyu@163.com, {sheng, qzz, sanglu}@nju.edu.cn, yibo.jin@smail.nju.edu.cn.*
- *Mingjun Xiao is with the School of Computer Science and Technology/ Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei 230052, China. E-mail: xiaomj@ustc.edu.cn.*
- *Jidong Ge is with the State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, Nanjing 210008, China. E-mail: gjd@nju.edu.cn.*

Manuscript received 14 May 2021; revised 7 Oct. 2021; accepted 9 Oct. 2021. Date of publication 14 Oct. 2021; date of current version 1 Nov. 2021.
This work was supported in part by National Key R&D Program of China under Grant 2017YFB1001801, in part by NSFC under Grants 61872175 and 61832008, and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization.
(Corresponding authors: Sheng Zhang and Yibo Jin.)
Recommended for acceptance by Y. Yang.
Digital Object Identifier no. 10.1109/TPDS.2021.3119948

Usually, network latency impacts application performance, service quality and user experience. In order to meet the requirements for low latency, a new paradigm of Multi-access Edge Computing (MEC) [3], [4], [5] is proposed as an extension of centralized clouds to tackle the challenge of network latency. The main characteristic of MEC is to bring the computation and storage resources to the edge networks. Edge users are directly connected to the nearest service-enabled edge networks, which can provide the capabilities of computing and caching. Application vendors can deploy their services and applications on the edge servers rather than the remote clouds in order to significantly reduce the latency from the cloud-hosted services to the end devices [6].

Service is an abstraction of applications hosted by the edge servers and requested by edge users, which includes Augmented Reality (AR), Virtual Reality (VR), facial identification, connected cars [7] and so on. Service placement refers to configuring the platform and storing the related libraries/ databases of a service on the edge server. Unlike the clouds which have huge and diverse resources, edge servers only have limited computing and storage resources to allow a small number of services to be placed [8]. Different kinds of services consume different amounts of resource and then result in different costs of service placement, which poses the challenge to tackling the service placement problem.

As shown in Fig. 1, MEC-enabled base stations each equipped with an edge server are densely distributed in the edge environment. The geographical coverage areas of them usually partially overlap in case of non-service areas where users fail to get service from any edge server [9], [10]. Each user in the overlapping will connect to one of MEC-

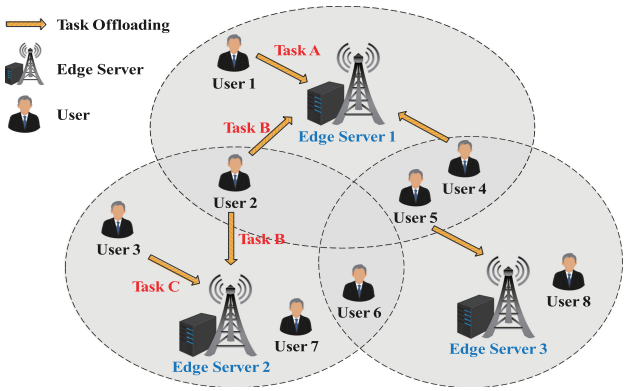


Fig. 1. User 1 offloads its task A to the only edge server 1, while user 2 divides its task B into two parts and offloads them to edge server 1 and 2.

enabled base stations covering them and be allocated to the associated edge server. Compared to cloud servers, there are limited computing resources on edge servers due to their size limit [11]. Thus, it is important to determine an effective user-to-edge-server allocation to prevent the waste of edge server resources.

When determining the user-to-edge-server allocation, the two popular paradigms of task offloading [3] include binary offloading and partial offloading. The paradigm of binary offloading applies for the highly integrated task, which cannot be partitioned and have to be offloaded as a whole to only one edge server. However, in practice, many applications are composed of multiple procedures or involve processing chunks of data (e.g., the application of vehicle counting in traffic surveillance videos), which makes it possible to partition the computation task into multiple parts and offload them to different edge servers for parallel execution.

Although service placement [9], [12], [13], [14], [15], [16], [17] and user allocation [18], [19], [20], [21] have received much attention in research field, a joint design of them has not been studied adequately. Some other works [22], [23], [24], [25], [26] have tackled the joint design of service placement and user allocation, but fail to consider the aspect of constrained user-perceived delay. The rest [27], [28], [29], [30] have already taken the user-perceived delay into consideration , but they fail to achieve the approximation guarantee by rigorous proof.

In this paper, we study the user-perceived delay-aware service placement and user allocation problem in MEC. The main contributions of this paper are as follows.

- We present the modeling of MEC-enabled network, user-perceived delay and total cost of task offloading system, based on which we consider the problem of user-perceived delay-aware service placement and user allocation.
- We formulate the user-perceived delay-aware service placement and user allocation problem as a mixed integer linear programming problem which aims at minimizing the total cost of task offloading system, and we prove that it is NP-hard by reducing another well-known NP-hard problem to it.
- Due to the NP-hardness of the considered problem, it is impossible to find the optimal solution in polynomial time. To effectively deal with its high complexity, we propose LOCUS, which is implemented based on three local-search operations *New*, *Swap* and *Delete*. We analyze the time complexity of LOCUS and prove that it achieves an approximation factor of $(8+\delta)$ for a sufficiently large constant δ.
- In the experiment, we use 20 Raspberry Pis and 5 PowerEdge R740s to simulate the edge users and edge servers, respectively; meanwhile, we design 5 kinds of services including word counting, word finding, vehicle counting in a video, pedestrian counting in a video and object detection in a video. Based on the real-world dataset derived from the AI City Datasets 2019 [31], we compare our proposed algorithm LOCUS with 4 existing designs. The experiment results show that LOCUS outperforms other methods and significantly reduces the total cost of task offloading system.

The remainder of this paper is organized as follows. We review the related work in Section 2. We introduce the system and notations in Section 3. In Section 4, we formulate the considered optimization problem and analyze its complexity. After that, we propose a polynomial-time algorithm LOCUS in Section 5 and then analyze its approximation guarantee in Section 6. Through some experiments, we compare our proposed algorithm with other existing designs and evaluate its performance in Section 7. Finally, we discuss some possible future works and conclude the paper in Section 8.

2 Related Work

We summarize some studies by the following categories and highlight their drawbacks compared with our work.

2.1 Multi-Access Edge Computing

Xu *et al.* [9] investigated the dynamic service placement in MEC, and proposed an online algorithm which jointly optimizes task offloading and dynamic service caching to tackle the unknown system dynamics, service heterogeneity and decentralized coordination. Chen *et al.* [16] studied the problem of collaborative service placement in MEC and proposed an efficient decentralized algorithm where the service placement decisions of BSs are optimized. Xu *et al.* [17] designed a distributed game-theoretical mechanism for the problem of service placement, where resources are shared among the service providers and the social cost of them is minimized. Zhan *et al.* [19] designed a decentralized algorithm for user allocation and computation offloading, where game theory is applied in the algorithm design and users choose their offloading decisions independently. Chen *et al.* [21] formulated the task offloading and user allocation problem in MEC as a minority game, and proposed an minority game based scheme converging to a near-optimal point. Lai *et al.* [18] utilized the distributed nature of edge computing and proposed an efficient game-theoretic approach to tackle the problem of user allocation in the MEC environment.

These works have studied the issues related to service placement or user allocation in the multi-access edge computing environment, but fail to investigate the joint design of service placement and user allocation.

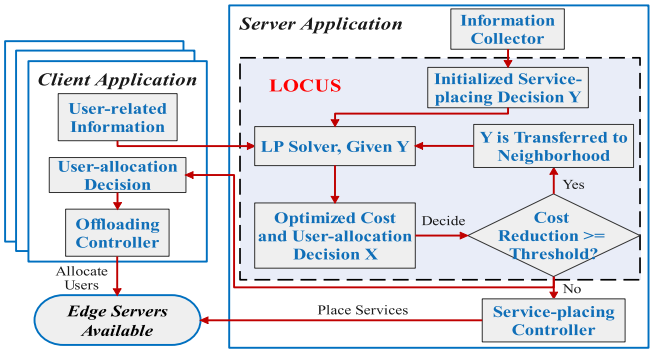


Fig. 2. The architecture of the edge-based service placement and user allocation system.

2.2 Service Placement and User Allocation

Behravesh *et al.* [22] proposed a heuristic algorithm to tackle the problem of joint user association, SFC placement, and resource allocation, employing mixed-integer linear programming techniques. Poularakis *et al.* [23] studied the joint optimization of service placement and request routing in MEC environment and proposed an algorithm that achieves close-to-optimal performance using randomized rounding. Yang *et al.* [24] designed a Benders decomposition-based algorithm to solve the problem of placing cloudlets and allocating requested tasks to cloudlets and public cloud with the minimum total energy consumption. Yao *et al.* [25] investigated how to deploy the servers in a cost-effective manner without violating the service quality, and a low-complexity heuristic algorithm was invented to address it. Tran *et al.* [26] considered the cost-aware joint service caching and task offloading assignment problem, and designed a polynomial-time iterative algorithm to tackle it.

However, these related works fail to consider the aspect of constrained user-perceived delay when tackling the joint design of service placement and user allocation.

2.3 User Perceived Delay

ShuffleDog [27] was implemented on commercial smartphones to significantly reduce the user-perceived latency of foreground apps in running with aggressive background workload. Zhang *et al.* [28] proposed queuing models for online service systems with proactive serving capability and characterize the user delay reduction by proactive serving. Jing *et al.* [29] investigated the content placement and delivery strategies in the cache-enabled wireless networks, which can characterize the end-to-end user-perceived delay and data rates simultaneously. Huang *et al.* [30] proposed a delay-tolerant wireless caching system that takes both the feedback delay and users' availability into consideration.

These works have already taken the user-perceived delay into consideration. However, they fail to achieves the approximation guarantee by rigorous proof.

3 System Model

Similar to [9], [26], [32], we divide time into many time slots, each of which has a duration matching the timescale where service placement and user allocation decisions can be updated. In the following, we consider the user-perceived delay-aware service placement and user allocation in a specific time slot where the index is omitted for the sake of simplicity.

TABLE 1
Major Notations Used for Model

Inputs	Descriptions
$\mathcal{N}, N$	Set of edge users, number of edge users
$\mathcal{M}, M$	Set of edge servers, number of edge servers
$\mathcal{S}, S$	Set of services, number of services
l_n, b_n	Workload and input data size of user n's task
F_m	Computing capacity of edge server m
L_m	Workload upper limit for edge server m
$\mathcal{N}_s$	Set of users requesting each service s
$\mathcal{N}_m$	Set of users within edge server m's coverage
$\mathcal{M}_n$	Set of user n's neighboring edge servers
$D^c_{n,m}$	Computing delay for user n on edge server m
$r_{n,m}$	Uplink data rate between user n and server m
$D^t_{n,m}$	Transmission delay between user n and server m
d_n	User n's perceived delay requirement
$c^p_{m,s}$	Cost to place service s at edge server m
c^u_m	Cost for one unit workload execution on server m
c^e	Cost for one unit of energy consumption
C^p, C^u, C^e	Service placement cost, edge server usage cost, energy consumption cost
Decisions	**Descriptions**
$x_{n,m}$	Fraction of user n's task offloaded to server m
$y_{m,s}$	Binary service-placing variable indicating whether service s is placed on edge server m
$\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$	Overall user-allocation and service-placing decision

The system architecture considered in this paper is depicted in Fig. 2. On the client side, users upload their user-related information (e.g., input data size, workload, user-perceived delay requirement) to the server application; afterwards, they receive the user-allocation decisions in turn. The offloading controllers on the client side allocate users to the edge servers available and offloading users' tasks to the corresponding edge servers.

In the server application, our proposed local-search based algorithm LOCUS (which will be introduced in detail in Section 5) is invoked, and it takes the information about edge servers and services (e.g., computing capacity of edge servers, workload upper limit for edge servers and set of services) as the input. Based on the results calculated from LOCUS, the service-placing controller places services at the corresponding edge servers and send the user-allocation decisions back to the clients.

In the rest of this section, we present the modeling of MEC-enabled network, user-perceived delay and total cost of task offloading system. Some important notations used for model are listed in Table 1.

3.1 MEC-Enabled Network

We consider a MEC-enabled wireless network consisting of M edge servers, denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$. As shown in Fig. 1, each edge server m is accessible via a base station covering a specific geographical area $Cover(m)$ and can provide computing services to edge users in its coverage area. We assume that there are S services, denoted as $\mathcal{S} = \{1, 2, \ldots, S\}$, and each user n in the set of edge users $\mathcal{N} = \{1, 2, \ldots, N\}$ has a task requiring one of these services to be executed.

Each user's task has its own computation demand and input data size. We let the workload and input data size of each user n's task be l_n [CPU cycles] and b_n [bits], respectively. In the MEC-enabled network, each edge user can be

in the coverage area of multiple edge servers and it can choose to offload its task to these edge servers. As considered in [9], [26], [32], we assume that each user can split its task into multiple portions and offload each of them to different edge server, and that the computation demand of each user's task is proportional to the input data size. We define the continuous user-allocation variable as $x_{n,m} \in [0,1]$ to denote the fraction of user n's task offloaded to server m. Thus, we have

$$\sum\nolimits_{m\in\mathcal{M}} x_{n,m} = 1, \forall n \in \mathcal{N}. \tag{1}$$

Then we can calculate that the workload and input data size of user n's task offloaded to the edge server m are $x_{n,m}l_n$ and $x_{n,m}b_n$, respectively.

Following the ideas in [9], [20], [33], though the multi-access edge computing server may perform parallel processing by allocating its computing capacity to the multiple computing tasks that have been offloaded to it, we reasonably assume that each user's task on the edge server m can be processed with computing capacity F_m [CPU cycles/s] since the workload on each edge server is limited (i.e., L_m [CPU cycles]) in a specific time-slot, which means that the total sum of the task workloads offloaded to edge server m in the specific time-slot cannot exceed L_m. Thus, the limited computing resources constraint of edge server m can be showed as

$$\sum\nolimits_{n\in\mathcal{N}} x_{n,m} l_n \le L_m, \forall m \in \mathcal{M}. \tag{2}$$

Besides, we define the binary service-placing variable as $y_{m,s} \in \{0,1\}$, where $y_{m,s} = 1$ if service s is placed at edge server m and $y_{m,s} = 0$ otherwise. Edge users requesting service s can offload their tasks to server m only when $y_{m,s} = 1$. Thus, it must hold that

$$x_{n,m} \le y_{m,s}, \forall n \in \mathcal{N}_s, \tag{3}$$

where $\mathcal{N}_s \subseteq \mathcal{N}$ represents the set of edge users requesting service s. For ease of exposition, we use $\mathcal{X} \triangleq \{x_{n,m} : n \in \mathcal{N}, m \in \mathcal{M}\}$ and $\mathcal{Y} \triangleq \{y_{m,s} : m \in \mathcal{M}, s \in \mathcal{S}\}$ to denote the overall user-allocation decision and overall service-placing decision, respectively.

3.2 User-Perceived Delay

Similar to the works in [15], [34], [35], the user-perceived delay in the MEC environment is mainly determined by computing delay and transmission delay.

Computing Delay. As mentioned previously, the computing capacity of each edge server m is F_m [CPU cycles/s] in the specific time slot. Thus, the computing delay for user n on the edge server $m \in M_n$ can be calculated as $D^c_{n,m} = x_{n,m}l_n/F_m$.

Transmission Delay. To keep the reasonable complexity of the physical-layer wireless channels, we consider that the users and edge servers use single antenna for transmission, and that the uplink channel gain $h_{n,m}$ between user n and edge server m is constant based on the best channel gain conditions of the users during the specific task offloading time-slot [9], [26], [36].

Additionally, we consider that neighboring edge servers are assigned orthogonal frequency and employ enhanced inter-cell interference coordination techniques, which is proposed in LTE Rel. 10 [37]. Thus, each user can occupy an orthogonal subchannel with bandwidth w. Then we can calculate the uplink data rate [bits/s] between user n and edge server m, according to the Shannon-Hartley formula [33], [38], [39], as

$$r_{n,m} = w\log_2(1 + p_n h_{n,m}/\sigma^2), \tag{4}$$

where the transmission power p_n is the input parameter determined by each user n, σ^2 is the background noise variance, and $p_n h_{n,m}/\sigma^2$ is the signal-to-noise ratio of the uplink channel between user n and edge server m.

Thus, when user n is connected to edge server $m \in M_n$, the transmission delay for it can be calculated as

$$D^t_{n,m} = \frac{x_{n,m} b_n}{w\log_2(1 + p_n h_{n,m}/\sigma^2)}. \tag{5}$$

Total User-Perceived Delay. Based on the computing delay and transmission delay for each user n, we calculate the user-perceived delay as

$$D_n = \max_{m\in\mathcal{M}_n} \{D^c_{n,m} + D^t_{n,m}\}, \tag{6}$$

where we define $\mathcal{M}_n \triangleq \{m \in \mathcal{M} | n \in Cover(m)\}$, and user n can offload the task to its neighboring edge servers in $\mathcal{M}_n$. It is worth noting that we neglect the delay for the edge servers to send back the results of the computation as in [26], [36], [40], [41], due to the fact that in many MEC-enabled services (e.g., video analysis and massive text mining), the size of the computation result is much smaller than the input data size.

For each user and the service it requests, too high user-perceived delay is not acceptable. Thus, it must hold that

$$D_n \le d_n, \forall n \in \mathcal{N}, \tag{7}$$

where d_n means user n's perceived delay requirement.

3.3 Total Cost of Task Offloading System

Similar to previous work [9], [26], different user allocation and service placing decisions incur different service placement, edge server usage and energy consumption costs in the specific time-slot.

3.3.1 Service Placement Cost

Due to the rapid development of storage technology [40], we consider that the available space for service placing at the edge servers is unlimited, while introducing the cost associated with the service placement. This cost accounts for the monetary cost imposed by network infrastructure or service providers for storage space utilization at edge servers. Thus, the service placement cost is calculated as

$$C^p = \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{s\in\mathcal{S}} y_{m,s} c^p_{m,s}, \tag{8}$$

where $c^p_{m,s}$ represents the cost to place service s at edge server m. Generally, $c^p_{m,s}$ depends on the storage ability of server m and computing complexity of service s.

3.3.2 Edge Server Usage Cost

When the processing load is offloaded to the edge servers, the edge server usage cost accounts for the computation consumption, the fees of which are charged by network infrastructure or service providers. Thus, the edge server usage cost is defined as

$$C^u = \sum\nolimits_{m\in\mathcal{M}} \sum\nolimits_{n\in\mathcal{N}_m} x_{n,m} l_n c_m^u, \tag{9}$$

where we use $\mathcal{N}_m \triangleq \{n \in \mathcal{N} | n \in Cover(m)\}$ to denote the set of users within edge server m's coverage, and c_m^u represents the cost associated with execution of one unit workload on the edge server m.

3.3.3 Energy Consumption Cost

Energy consumption plays an important role for the users in the multi-access computing environment [42], and here we use the monetary cost of energy consumption to represent its importance. As mentioned previously, when user n is connected to server $m \in \mathcal{M}_n$, the transmission delay can be calculated as Eq. (5). Thus, the total energy consumption cost for all users is

$$C^e = \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}_m} x_{n,m} \frac{p_n b_n}{w \log_2(1 + p_n h_{n,m}/\sigma^2)} c^e, \tag{10}$$

where the transmission power p_n is the input parameter determined by each user n, and the monetary cost for one unit of energy consumption is set to c^e.

3.3.4 Total Cost

Considering that the monetary costs of service placement, edge server usage and energy consumption are the most important in our task offloading system, we get the total cost based on them as

$$C_{total} = C^p + C^u + C^e. \tag{11}$$

We will give the problem formulation in the next section, where the total cost in the task offloading system will be minimized, while guaranteeing that the user-perceived delay requirement of each user is satisfied.

4 Problem Formulation and Analysis

In this section, we formulate our considered problem and then show its intractability by complexity analysis.

4.1 Problem Formulation

Given the user-allocation decision $\mathcal{X}$ and service-placing decision $\mathcal{Y}$, we can get the total cost in the task offloading system as $W(\mathcal{X}, \mathcal{Y}) = C_{total}$ in Eq. (11). Thus, the problem of user-perceived delay-aware joint service placement and user allocation assignment which aims at minimizing the total cost can be formulated as a mixed integer linear programming problem

$$\mathbb{P}_1 : \min_{\mathcal{X},\mathcal{Y}} \quad C^p + C^u + C^e \tag{12a}$$

$$\text{s.t.} \quad x_{n,m} \in [0,1], \qquad \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \tag{12b}$$

$$y_{m,s} \in \{0,1\}, \qquad \forall m \in \mathcal{M}, \forall s \in \mathcal{S}, \tag{12c}$$

$$\textstyle\sum_{m\in\mathcal{M}_n} x_{n,m} = 1, \qquad \forall n \in \mathcal{N}, \tag{12d}$$

$$x_{n,m} \le y_{m,s}, \qquad \forall n \in \mathcal{N}_s, \tag{12e}$$

$$\textstyle\sum_{n\in\mathcal{N}_m} x_{n,m} l_n \le L_m, \qquad \forall m \in \mathcal{M}, \tag{12f}$$

$$D^c_{n,m} + D^t_{n,m} \le d_n, \quad \forall n \in \mathcal{N}, \qquad \forall m \in \mathcal{M}. \tag{12g}$$

In the formulated problem $\mathbb{P}_1$ above, constraints (12b) and (12c) specify the definitional domains of the user-allocation decision and service-placing decision variables, respectively. Constraint (12d) ensures that all of the workload from each user will be executed collectively by the edge servers. Constraint (12e) ensures that the users requesting service s can offload their tasks to server m only when service s has been placed on server m. Constraint (12f) guarantees that the total sum of the task workloads offloaded to each edge server cannot exceed its workload upper limit. Constraint (12g) guarantees the user-perceived delay requirement should be satisfied.

4.2 Complexity Analysis

Proposition 1. *$\mathbb{P}_1$, the problem of user-perceived delay-aware joint service placement and user allocation assignment which aims at minimizing the total cost is NP-hard.*

Proof. We first briefly describe the well-known NP-hard problem, capacitated facility location problem (CFLP) [43]. Then we reduce CFLP to our considered problem $\mathbb{P}_1$ and show that problem $\mathbb{P}_1$ is also NP-hard.

① *Desciprtion of CFLP.* In the capacitated facility location problem, suppose that there are $\mathbb{N}$ customers and $\mathbb{M}$ facilities. We let a_n denote the demand of customer n, and each customer can divide its demand into multiple parts, which are sent to different facilities for production; meanwhile, we suppose that each facility m has a production capacity u_m, which is the maximum amount of product that can be produced by facility m.

We use f_m to denote the cost of opening facility m and g_{mn} means the cost of shipping the product from facility m to customer n. Thus, the total cost is the sum of the facility-opening cost and product-shipping cost. In order to meet some fixed demands at minimum cost, we need to decide (i) which of the $\mathbb{M}$ facilities to open and (ii) which open facilities to use to supply the $\mathbb{N}$ customers.

② *Reducing CFLP to $\mathbb{P}_1$.* By writing each instance of CFLP as a special case of problem $\mathbb{P}_1$, we can reduce CFLP to problem $\mathbb{P}_1$:

- The number of services in problem $\mathbb{P}_1$ is set to 1, and each user's perceived delay requirement in problem $\mathbb{P}_1$ is set large enough.
- The consumer n with demand a_n in CFLP is mapped to the user n with computation demand l_n in problem $\mathbb{P}_1$.
- The (opened) facility m with production capacity u_m in CFLP is mapped to the edge server m

(where the service is placed) with workload upper limit L_m in problem $\mathbb{P}_1$.
- The cost of opening facilities in CFLP is mapped to the cost of placing services in problem $\mathbb{P}_1$.
- The cost of shipping products in CFLP is mapped to the total cost of edge server usage and energy consumption in problem $\mathbb{P}_1$.

Hence, NP-hard problem CFLP can be reduced to our considered problem $\mathbb{P}_1$, and problem $\mathbb{P}_1$ is NP-hard. □

Since problem $\mathbb{P}_1$ is NP-hard, we cannot obtain the exact solution to it in polynomial time. In the next section, we propose an approximate algorithm for it, which can give the suboptimal solution in the polynomial time.

5 Proposed Algorithm

In this section, we propose a polynomial-time local-search algorithm which achieves an approximation factor of $(8+\delta)$, for a sufficiently large constant δ. The local-search based algorithm first starts with an arbitrary feasible solution and then repeatedly improves the solution by performing local search steps. Within a polynomial number of local search steps, we can obtain a suboptimal solution achieving the desired approximation factor.

Algorithm 1. LP Solver for $\mathbb{P}_2(\mathcal{Y})$

Input: Y
Output: $\mathcal{X}$
1 Solve problem $\mathbb{P}_2(\mathcal{Y})$ using interior point method in polynomial time;
2 Obtain the minimized total cost $w(Y)$;
3 Output optimal user-allocation decision $\mathcal{X}$;

5.1 LP Solver for a Simplified Problem

To better describe the local search operations later, we first formulate a simplified problem from the original problem $\mathbb{P}_1$. For problem $\mathbb{P}_1$, we need to obtain a solution of overall user-allocation decision $\mathcal{X}$ and service-placing decision $\mathcal{Y}$. And if we fix the decision $\mathcal{Y}$, the original problem $\mathbb{P}_1$ can be simplified as a Linear Programming (LP) problem. Thus, we consider a user-allocation problem $\mathbb{P}_2(\mathcal{Y})$ where the service-placing decision $\mathcal{Y}$ is given, and we determine the user-allocation decision $\mathcal{X}$ to minimize the total cost $W_{\mathcal{Y}}(\mathcal{X}) = C_{total}$. Then the simplified problem $\mathbb{P}_2(\mathcal{Y})$ can be formulated as

$$\mathbb{P}_2(\mathcal{Y}): \min_{\mathcal{X}} \quad C^p + C^u + C^e \tag{13a}$$

$$\text{s.t.} \quad x_{n,m} \in [0,1], \qquad \forall n \in \mathcal{N}, \forall m \in \mathcal{M}, \tag{13b}$$

$$\textstyle\sum_{m \in \mathcal{M}_n} x_{n,m} = 1, \qquad \forall n \in \mathcal{N}, \tag{13c}$$

$$x_{n,m} \le y_{m,s}, \qquad \forall n \in \mathcal{N}_s, \tag{13d}$$

$$\textstyle\sum_{n \in \mathcal{N}_m} x_{n,m} l_n \le L_m, \qquad \forall m \in \mathcal{M}, \tag{13e}$$

$$D^c_{n,m} + D^t_{n,m} \le d_n, \qquad \forall n \in \mathcal{N}, \forall m \in \mathcal{M}. \tag{13f}$$

Different from problem $\mathbb{P}_1$, we only think about the user-allocation decision variables in problem $\mathbb{P}_2(\mathcal{Y})$. Furthermore, since problem $\mathbb{P}_2(\mathcal{Y})$ is an LP problem, it can be efficiently solved using interior point method (e.g., implemented in SciPy or Cplex) in polynomial time.

For convenience, we use $Y = \{(m,s) : y_{m,s} = 1, m \in \mathcal{M}, s \in \mathcal{S}\}$ to represent the set of services that have been placed at the corresponding server, and it can be determined by the overall service-placing decision $\mathcal{Y}$. As shown in Algorithm 1, we input the set Y of placed services and then get the optimal user-allocation decision $\mathcal{X}$. Besides, we use $w(Y)$ to denote the minimized total cost when the set of placed services is fixed at Y in Algorithm 1. Therefore, based on Algorithm 1, if we have obtained the optimal service-placing decision $\mathcal{Y}^*$, we can solve the considered problem $\mathbb{P}_1$ efficiently.

5.2 Three Local-Search Operations

To obtain the (near-)optimal service-placing decision, we design three local-search operations to let the service-placing decision $\mathcal{Y}$ get closer to the optimum step by step. Before describing the three local-search operations, we introduce the concept of *neighborhood* in problem $\mathbb{P}_1$, which will be used later in this section.

Definition 1 (Neighborhood). *In problem $\mathbb{P}_1$, the neighborhood of $\boldsymbol{Y}$ is defined as*

$$\boldsymbol{Nei}(\boldsymbol{Y}) = \{\boldsymbol{T} \subseteq \boldsymbol{F} : |\boldsymbol{T} - \boldsymbol{Y}| \le 1, |\boldsymbol{Y} - \boldsymbol{T}| \le 1\}, \tag{14}$$

where $\boldsymbol{F}$ is $\{(m,s) : m \in \mathcal{M}, s \in \mathcal{S}\}$.

When we apply a local-search based approach to the considered problem $\mathbb{P}_1$, if we are given a current feasible solution corresponding to a set Y of placed services, the local-search operations will traverse the neighborhood of Y and set the next new feasible solution as a set $\boldsymbol{T}$ of minimum cost in $\boldsymbol{Nei}(\boldsymbol{Y})$. Since the neighborhood of $\boldsymbol{Y}$ contains a polynomial number of solutions and the cost of each solution can be calculate via Algorithm 1 in polynomial time, the local-search operations mentioned above can be performed efficiently. Based on Definition 1, we divide the neighborhood of $\boldsymbol{Y}$ into 3 subsets:

- $\boldsymbol{Sub}_1 = \{\boldsymbol{T} \subseteq \boldsymbol{F} : |\boldsymbol{T} - \boldsymbol{Y}| = 1, |\boldsymbol{Y} - \boldsymbol{T}| = 0\}$;
- $\boldsymbol{Sub}_2 = \{\boldsymbol{T} \subseteq \boldsymbol{F} : |\boldsymbol{T} - \boldsymbol{Y}| = 1, |\boldsymbol{Y} - \boldsymbol{T}| = 1\}$;
- $\boldsymbol{Sub}_3 = \{\boldsymbol{T} \subseteq \boldsymbol{F} : |\boldsymbol{T} - \boldsymbol{Y}| = 0, |\boldsymbol{Y} - \boldsymbol{T}| = 1\}$.

Next we design three local-search operations including *New*, *Swap* and *Delete*. From the set $\boldsymbol{Y}$ of placed services, we can move to the set $\boldsymbol{Sub}_1$, $\boldsymbol{Sub}_2$ and $\boldsymbol{Sub}_3$ through operation *New*, *Swap* and *Delete*, respectively.

Algorithm 2. Local-Search Operation *New*

Input: Set Y of placed services
Output: $\boldsymbol{bstNeighbor}$, $cstReduce$
1 $\boldsymbol{bstNeighbor} \leftarrow \varnothing$;
2 $cstReduce \leftarrow 0$;
3 **for** *each* $(m,s) \in \{(m,s) : m \in \mathcal{M}, s \in \mathcal{S}\} - \boldsymbol{Y}$ **do**
4 **if** $w(\boldsymbol{Y}) - w(\boldsymbol{Y} \cup \{(m,s)\}) > cstReduce$ **then**
5 $\boldsymbol{bstNeighbor} \leftarrow \boldsymbol{Y} \cup \{(m,s)\}$;
6 $cstReduce \leftarrow w(\boldsymbol{Y}) - w(\boldsymbol{bstNeighbor})$;

New. As show in Algorithm 2, we design the local-search operation *New*. Through *New*, we add a new element to set $\boldsymbol{Y}$, which means that a new service will be placed at some edge server. From line 3 to 8, we traverse the neighborhood of $\boldsymbol{Y}$ to find the best neighbor that reduces the total cost most. Finally, we get the best neighbor $\boldsymbol{bstNeighbor}$ in $\boldsymbol{Sub}_1$, and the corresponding cost reduction is $cstReduce$.

Swap. In Algorithm 3, we describe the local-search operation *Swap*. Through *Swap*, we replace an element in $\boldsymbol{Y}$ with an element not in $\boldsymbol{Y}$, which means that a placed service will be removed and a new service will be placed at some edge server. Thus, we traverse $\boldsymbol{Nei}(\boldsymbol{Y})$ to find the appropriate to-be-added and to-be-removed services in lines 3-10. We finally obtain the best neighbor $\boldsymbol{bstNeighbor}$ in $\boldsymbol{Sub}_2$ and maximize the cost reduction.

Algorithm 3. Local-Search Operation *Swap*

Input: Set $\boldsymbol{Y}$ of placed services
Output: $\boldsymbol{bstNeighbor}$, $cstReduce$
1 $\boldsymbol{bstNeighbor} \leftarrow \varnothing$;
2 $cstReduce \leftarrow 0$;
3 **for** *each* $(m, s) \in \boldsymbol{Y}$ **do**
4 **for** *each* $(m', s') \in \{(m, s) : m \in \boldsymbol{\mathcal{M}}, s \in \boldsymbol{\mathcal{S}}\} - \boldsymbol{Y}$ **do**
5 **if** $w(\boldsymbol{Y}) - w(\boldsymbol{Y} \cup \{(m', s')\} - \{(m, s)\}) > cstReduce$ **then**
6 $\boldsymbol{bstNeighbor} \leftarrow \boldsymbol{Y} \cup \{(m', s')\} - \{(m, s)\}$;
7 $cstReduce \leftarrow w(\boldsymbol{Y}) - w(\boldsymbol{bstNeighbor})$;

Delete. The design of operation *Delete* in Algorithm 4 is similar to *New* and *Swap*. The neighborhood of $\boldsymbol{Y}$ is traversed to find the optimal to-be-removed service which maximizes the cost reduction $cstReduce$ in lines 3-10, and the best neighbor $\boldsymbol{bstNeighbor}$ can be obtained.

Algorithm 4. Local-Search Operation *Delete*

Input: Set $\boldsymbol{Y}$ of placed services
Output: $\boldsymbol{bstNeighbor}$, $cstReduce$
1 $\boldsymbol{bstNeighbor} \leftarrow \varnothing$;
2 $cstReduce \leftarrow 0$;
3 **for** *each* $(m, s) \in \boldsymbol{Y}$ **do**
4 **if** $w(\boldsymbol{Y}) - w(\boldsymbol{Y} - \{(m, s)\}) > cstReduce$ **then**
5 $\boldsymbol{bstNeighbor} \leftarrow \boldsymbol{Y} - \{(m, s)\}$;
6 $cstReduce \leftarrow w(\boldsymbol{Y}) - w(\boldsymbol{bstNeighbor})$;

5.3 Local-Search Algorithm LOCUS

Now we are ready to describe the local-search algorithm LOCUS in Algorithm 5. In step 1, we initialize the service placement decisions and calculate the initial total cost. As shown in lines 1-5, we place the services that are requested by users in $\boldsymbol{\mathcal{N}}_m$ at each edge server m. Then in step 2, LOCUS repeatedly invokes one of the three local-search operations *New*, *Swap* and *Delete* designed above, as long as the cost reduction in each step is sufficiently large. In Algorithm 5, the function $p(M, S)$ is a polynomial in $M \cdot S$, where M is the edge server number and S is the service number (e.g., we can choose $p(M, S) = M^2S^2$). It is worth noting that when LOCUS is being implemented, the three basic operations *New*, *Swap* and *Delete* invoked in LOCUS can be performed in parallel, and within each iteration, the operation reducing the cost most will be selected. Besides, in each basic operation *New*, *Swap* or *Delete*, the operations of calculating $w(\cdot)$ can also be performed in parallel.

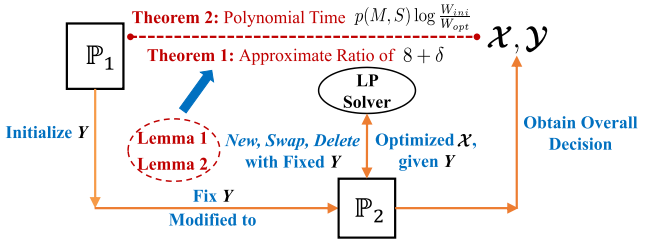


Fig. 3. Relationship between proposed problems, algorithms and theorems.

Algorithm 5. LOCUS for User-Perceived Delay-Aware Service Placement and User Allocation

1 %Step 1:Initialize service placement decisions;
2 $\boldsymbol{Y} = \varnothing$;
3 **for** *each* $m \in \boldsymbol{M}$ **do**
4 **for** *each* $n \in \boldsymbol{\mathcal{N}}_m$ **do**
5 $\boldsymbol{Y} \leftarrow \boldsymbol{Y} \cup \{(m, s_n^r)\}$;
6 %Step 2:Iterative local search for (near-)optimum;
7 **repeat**
8 $local\_success \leftarrow$ False;
9 $(\boldsymbol{bstNeighbor}, cstReduce) \leftarrow New(\boldsymbol{Y})$;
10 **if** $cstReduce \geq w(\boldsymbol{Y})/p(M, S)$ **then**
11 $\boldsymbol{Y}, local\_success \leftarrow \boldsymbol{bstNeighbor}, True$;
12 Goto line 7;
13 $(\boldsymbol{bstNeighbor}, cstReduce) \leftarrow Swap(\boldsymbol{Y})$;
14 **if** $cstReduce \geq w(\boldsymbol{Y})/p(M, S)$ **then**
15 $\boldsymbol{Y}, local\_success \leftarrow \boldsymbol{bstNeighbor}, True$;
16 Goto line 7;
17 $(\boldsymbol{bstNeighbor}, cstReduce) \leftarrow Delete(\boldsymbol{Y})$;
18 **if** $cstReduce \geq w(\boldsymbol{Y})/p(M, S)$ **then**
19 $\boldsymbol{Y}, local\_success \leftarrow \boldsymbol{bstNeighbor}, True$;
20 **until** $local\_success = False$

6 Performance Analysis

The relationships between all the lemmas and theorems are illustrated in Fig. 3, where the main result regarding the optimization objective with approximate guarantee achieved by LOCUS is shown in *Theorem 1* and the result about the time complexity is shown in *Theorem 2*.

For convenience, we use $w_e(\boldsymbol{Y})$ to represent the sum of the edge server usage cost and energy consumption cost, and use $w_s(\boldsymbol{Y})$ to represent the service placement cost when *Algorithm 1* is applied to the problem $\mathbb{P}_2(\boldsymbol{\mathcal{Y}})$, where the set of placed services is $\boldsymbol{Y}$. And we use $\boldsymbol{Y}^*$ to denote the set of placed services corresponding to the optimal service-placing decision $\boldsymbol{\mathcal{Y}}^*$.

Before we analyze the approximation guarantee of LOCUS, we refer to the work [44] and give two lemmas, as shown in Lemmas 1 and 2. The two lemmas can be obtained based on the *difference graph* that captures the differences in an arbitrary solution $(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})$ and the optimal solution $(\boldsymbol{\mathcal{X}}^*, \boldsymbol{\mathcal{Y}}^*)$. Specifically, we consider the flow $\boldsymbol{\mathcal{X}}$ as a flow in a bipartite graph with vertices corresponding to services and edge users. In particular, there are $x_{n,m}$ units flowing from service $y_{m,s}$ to edge user n along each edge $(y_{m,s}, n)$. In order to compare the arbitrary solution $(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})$ with the optimal solution $(\boldsymbol{\mathcal{X}}^*, \boldsymbol{\mathcal{Y}}^*)$, we consider the flow $\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}^*$ whereby each edge $(y_{m,s}, n)$ has

$x_{n,m} - x^*_{n,m}$ units of flow. The difference graph can bound the cost of reassigning the computation load from the currently cached services to the services cached in the optimal solution, and the bounded cost difference is described in Lemmas 1 and 2. For brevity, we here omit the detailed proof of Lemmas 1 and 2, which can be referred to in [44].

Lemma 1 ([44]). *If there is no (m, s) which is not in the current $\boldsymbol{Y}$, such that*

$$w(\boldsymbol{Y}) - w(\boldsymbol{Y} \cup \{(m,s)\}) \geq w(\boldsymbol{Y})/p(M,S), \tag{15}$$

then

$$w_e(\boldsymbol{Y}) \; < \; w(\boldsymbol{Y}^*) + MSw(\boldsymbol{Y})/p(M,S), \tag{16}$$

Lemma 2 ([44]). *If there is no (m, s) in the current $\boldsymbol{Y}$ to perform the* Swap *or* Delete *operation so that the total cost is reduced by at least $w(\boldsymbol{Y})/p(M,S)$, then*

$$w_s(\boldsymbol{Y})(1 - \frac{(MS)^2}{p(M,S)}) \; < \; 5w(\boldsymbol{Y}^*) + 2w_e(\boldsymbol{Y}) + \frac{w(\boldsymbol{Y})}{MS}. \tag{17}$$

Theorem 1. *The solution obtained from LOCUS has the total cost at most $8 + \delta$ times that of the optimal cost in problem $\mathbb{P}_1$, for a sufficiently large constant δ.*

Proof. According to Algorithm 5, when LOCUS terminates after some local-search operations, there is no local-search operation *New*, *Swap* or *Delete* so that the total cost can be reduced by at least $w(\boldsymbol{Y})/p(M,S)$ any more. Thus, based on the two lemmas above, inequations (16) and (17) are satisfied. Plugging (16) into (17), we get

$$w_s(\boldsymbol{Y})\left(1 - \frac{(MS)^2}{p(M,S)}\right) \; < \; 7w(\boldsymbol{Y}^*) + \frac{2MSw(\boldsymbol{Y})}{p(M,S)} + \frac{w(\boldsymbol{Y})}{MS}. \tag{18}$$

Based on inequation (16), we can also get

$$w_e(\boldsymbol{Y})(1 - \frac{(MS)^2}{p(M,S)}) \; < \; w(\boldsymbol{Y}^*) + \frac{MSw(\boldsymbol{Y})}{p(M,S)}. \tag{19}$$

Then we let inequation (18) plus (19), and we have

$$w(\boldsymbol{Y})\left(1 - \frac{(MS)^2}{p(M,S)}\right) \leq 8w(\boldsymbol{Y}^*) + \frac{3MSw(\boldsymbol{Y})}{p(M,S)} + \frac{w(\boldsymbol{Y})}{MS}. \tag{20}$$

Rearranging, we obtain

$$\frac{w(\boldsymbol{Y})}{w(\boldsymbol{Y}^*)} \; < \; 8\frac{1}{(1 - \frac{(MS)^2}{p(M,S)} - \frac{3MS}{p(M,S)} - \frac{1}{MS})}. \tag{21}$$

Therefore, for a sufficiently large constant δ satisfying

$$\delta \geq 8\left(\frac{1}{(1 - \frac{(MS)^2}{p(M,S)} - \frac{3MS}{p(M,S)} - \frac{1}{MS})} - 1\right), \tag{22}$$

the approximate optimal solution obtained from LOCUS has the total cost at most $8 + \delta$ times that of the optimal cost in problem $\mathbb{P}_1$. □

Theorem 2. *The proposed algorithm LOCUS terminates after at most $O(p(M,S)\log \frac{W_{ini}}{W_{opt}})$ local-search operations, where W_{ini} is the initial total cost obtained in step 1 of LOCUS and W_{opt} is the optimal total cost of problem $\mathbb{P}_1$.*

Proof. We let R be the number of local-search operations which are performed in LOCUS and let W_k be the current total cost of problem $\mathbb{P}_1$ after the kth operation is performed. Specifically, we set $W_0 = W_{ini}$. According to Algorithm 5, it holds that

$$W_k - W_{k+1} \geq \frac{W_k}{p(M,S)}, \quad \forall k \in \{0, 1, 2, \ldots, R-1\}. \tag{23}$$

Thus, we can obtain

$$\frac{W_k}{W_{k+1}} \geq \frac{1}{1 - \frac{1}{p(M,S)}}, \quad \forall k \in \{0, 1, 2, \ldots, R-1\}. \tag{24}$$

LOCUS terminates after R local-search operations and we get the (near-)optimal total cost of $\mathbb{P}_1$. Thus, we have

$$\frac{W_{ini}}{W_{opt}} \geq \frac{W_0}{W_R} = \frac{W_0}{W_1} \cdot \frac{W_1}{W_2} \cdot \ldots \cdot \frac{W_{R-1}}{W_R} \geq \left(\frac{1}{1 - \frac{1}{p(M,S)}}\right)^R. \tag{25}$$

Besides, according to the definition of natural constant e, it holds that

$$\left(1 - \frac{1}{p(M,S)}\right)^{-p(M,S)} \geq \mathrm{e}. \tag{26}$$

Combining inequations (25) and (26), we obtain

$$R \leq p(M,S)\log \frac{W_{ini}}{W_{opt}}. \tag{27}$$

Therefore, the proposed algorithm LOCUS terminates after at most $O(p(M,S)\log \frac{W_{ini}}{W_{opt}})$ local-search operations, which is polynomial in MS, and it is shown that LOCUS has polynomial-time complexity. □

7 Experiments and Result Analysis

In this section, we evaluate the performance of our proposed algorithm LOCUS through experiments with various settings. Besides, we compare our design with some other existing approaches.

7.1 Experiment Settings

Similar to the previous works [20], [21], we first consider a multi-access edge computing system containing 20 users and 5 edge servers. As demonstrated in Fig. 4, we use 20 Raspberry Pis and 5 PowerEdge R740s (Silver 4210R 2.4G, 2*16GB RDIMM) to act as the users and edge servers, respectively. We design 5 kinds of services as follows. *Word counting*: counts the occurrences of a specific word in a large amount of text; *Word finding*: finds the positions of a specific word in a large amount of text; *Vehicle counting in a video*: counts the number of vehicles in the traffic surveillance video; *Pedestrian counting in a video*: counts the number of pedestrians in the traffic surveillance video; *Object detection in a video*: detects a lost child in the store surveillance video. The videos we used are mostly derived from the AI City Datasets 2019 [31] (hundreds of video clips, 15 minutes for each clip) for video

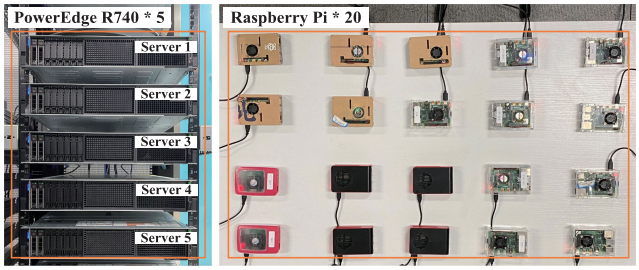
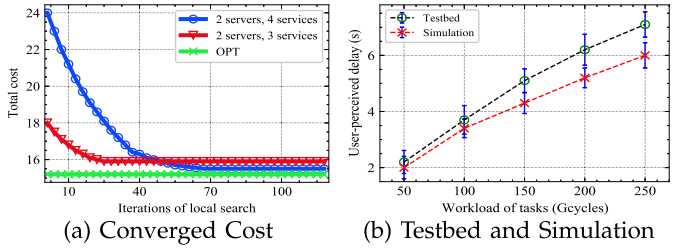


Fig. 4. Illustration of our testbed.



(a) Converged Cost (b) Testbed and Simulation

Fig. 5. Gradually converged total cost and user-perceived delay in testbed and simulation.

analytics upon YOLOv3[45]. The testbed mentioned above is used to test the effectiveness of our proposed algorithm.

Furthermore, to test the efficiency and scalability of LOCUS, we set and vary some parameters in the experiments. Based on the performance of hardware devices, the computing capacity F_m for each edge server m is assigned from the set $\{50, 100, 150\}$ Gcycles/s, and the workload upper limit L_m for edge server m is set to 900 Gcycles in the specific time slot. For the parameters in Shannon-Hartley formula, we refer to the works [20], [46]. The channel bandwidth is set to 20 MHz, and the background noise is set to 50 dBm. In a specific time-slot, each user holds the transmission power $p_n \sim N(1, 0.1)W$. We set the uplink channel gain $h_{n,m} = (dist_{n,m})^{\mu}$, where $dist_{n,m}$ is the distance between user n and edge server m, and the path loss factor μ is set to 4.

Beside, according to [15], each user n's perceived delay requirement d_n is uniformly distributed in [1,10] seconds. The optimization objective in problem $\mathbb{P}_1$ consists of three parts: service placement cost, edge server usage cost and energy consumption cost, and we assume that the three types of cost have almost the same influence on the overall optimization objective, as studied in [20], [26]. Based on the optimal solution derived from the MILP solver (e.g., Cplex, Mosek), we observe that the ratio of the number of placed service, units of executed workload and units of energy consumption is about $1 : 200 : 4$ on average. To balance the impacts of the three different costs, we set $c^p_{m,s} = 1$, $c^u_m = 1/200$ and $c^e = 1/4$. We compare our design LOCUS with 4 other schemes:

- Randomized Rounding Scheme (RRS): Similar to the design in [17], RRS first relaxes constraint (12c) into $0 \leq y_{m,s} \leq 1$, and then rounds the calculated fractional solution to an integer solution by using a randomized rounding technique.
- Greedy Scheme (GS): Traversing all users in a certain order, GS determines the way of service placement and user allocation for each user, which minimizes the total cost increase.
- All-service Scheme (AS): Assuming all the services are placed at each edge server, AS first solves the user allocation problem $\mathbb{P}_2(\mathcal{Y})$ in polynomial time, and then removes the services from the edge server wherever there is no corresponding offloaded task.
- OPTimal Scheme (OPTS): OPTS offers a baseline as it directly uses the MILP solver to solve $\mathbb{P}_1$.

7.2 Experiment Results

Cost Reduction and User-Perceived Delay. We first test the effectiveness of LOCUS and derive the cost reduction versus iterations in Fig. 5a. It shows that with the increase of iterations, the total cost in the task offloading system drops gradually. For the case of 2 edge servers and 4 services, we observe that the total cost needs more time to converge. However, when there are fewer edge servers and services, the cost easily becomes stable. Furthermore, we notion that the converged cost can be closer to the optimum when the numbers of edge servers and services are larger since there are more chances for local-search operations to reduce the total cost. For both of the two cases in Fig. 5a, the total cost can converge to a point which is near to the optimum, and it can be approximately guaranteed by Theorem 1.

After that, we compare the user-perceived delay in the testbed and simulation with different workloads of tasks, as shown in Fig. 5b. We observe that the measured user-perceived delay in the testbed is slightly higher than that in the simulation because we ignore the MAC-level/queuing delays and the instability of the edge server computing capacity in our model, which can be captured in the testbed-based experiment.

Comparisons in Total Cost. We then compare the total cost induced in some video-related services (e.g., vehicle counting, pedestrian counting and object detection) and text-related services (e.g., word counting and word finding) with different estimated workloads of tasks. As shown in Fig. 6a, we notice that the variance of cost in some video-related services is significantly greater than that in the text-related services because the video content usually changes a lot, which will result in the edge server usage cost varying much more.

We study the changes in total cost with varying workloads of tasks, delay requirements and workload upper limits in the task offloading system. In Fig. 6b, when we increase the average workload of tasks, the total costs obtained with the 5 schemes also rise, due to the fact that more task workloads lead to higher server usage cost and energy consumption cost. Furthermore, as there are workload upper limits for each edge server, more servers are needed to complete the increased workloads, which results in higher service placement cost. As shown in Fig. 6c, when the average delay requirement is increased, the total costs obtained with the 5 schemes are reduced since edge users have more edge server choices to make the total cost smaller. Besides, when the delay requirements are relaxed to a certain extent, the total cost remains almost constant. Fig. 6d demonstrates the impacts of varying workload upper limit average on total cost. When the workload upper limit average is varying from 700 Gcycles to 1100 Gcycles, the total cost goes down, because increased workload upper limit contributes to completing more workload on each edge server, which saves the cost of placing extra services on other edge servers. In Section 6, it is proven that our proposed algorithm LOCUS

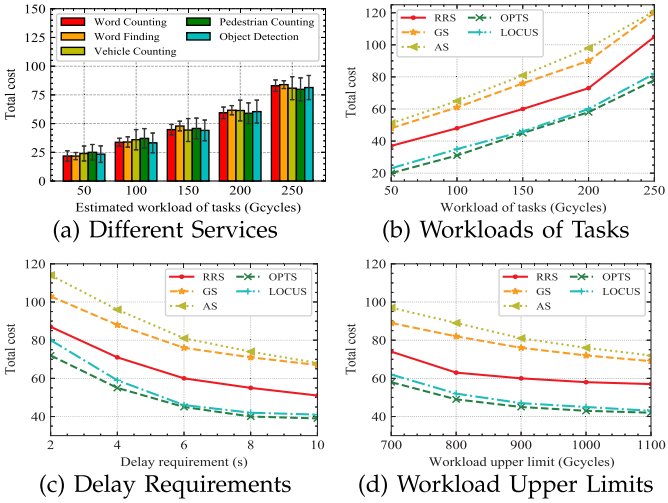


Fig. 6. Changes in total cost with varying workloads of tasks, delay requirements and workload upper limits.

achieves an approximation ratio. Actually, as shown in the experiment results, the total cost gap between LOCUS and the optimal scheme OPTS is very small. Additionally, in terms of total cost, our design LOCUS always performs much better than the other methods RRS, GS and AS.

Comparisons in Cost Components. We next focus on the three components (i.e., service placement cost, server usage cost and energy consumption cost) in total cost with varying numbers of users, servers and services. From Fig. 7, we can see that server placement cost is the main contributor to the total cost obtained with RRS and AS. For scheme RRS, each service placement decision variable $y_{m,s}$ is first relaxed to continuous variable in [0,1], thus it helps to select the better user allocation decisions which lead to lower server usage and energy consumption cost. Similarly, before AS solves the considered problem, it assumes all services are placed at each server. Thus, better user allocation decisions can be determined to minimize the server usage and energy consumption cost. Furthermore, the total cost of LOCUS and OPTS are lower since they both optimize the trade-off among service placement cost, server usage cost and energy consumption cost via the joint optimization of service placement and user allocation.

Comparisons in Calculation Time. We finally compare the time taken by different schemes to calculate the service placement and user allocation decisions with varying numbers of users, servers and services. As demonstrated in Fig. 8, when we increase the numbers of users, edge servers and services, more calculation time is needed to tackle the considered problem using RRS, OPTS and LOCUS. Moreover, it can be clearly seen that the calculation time OPTS takes to obtain the service placement and user allocation decisions grows the fastest, because it needs to find the optimal overall decision in the solution space of exponential size. For example, as shown in Fig. 8b, when the user number varies from 12 to 40, the added calculation time of OPTS is nearly 10^4 seconds, while the added calculation time of LOCUS is only hundreds of seconds. Therefore, our proposed algorithm LOCUS has good stability and scalability compared with others in different scenarios.

8 Conclusion and Future Work

In this paper, we investigate the user-perceived delay-aware service placement and user allocation problem in edge environment. We present the modeling of MEC-enabled network, user-perceived delay and total cost of task offloading system, based on which we formulate the user-perceived delay-aware service placement and user allocation problem as a mixed integer linear programming problem which aims at minimizing the total cost of task offloading system. Due to the intractability of our considered problem, we design a polynomial-time local-search based algorithm LOCUS which achieves provable guaranteed performance. We finally compare our proposed

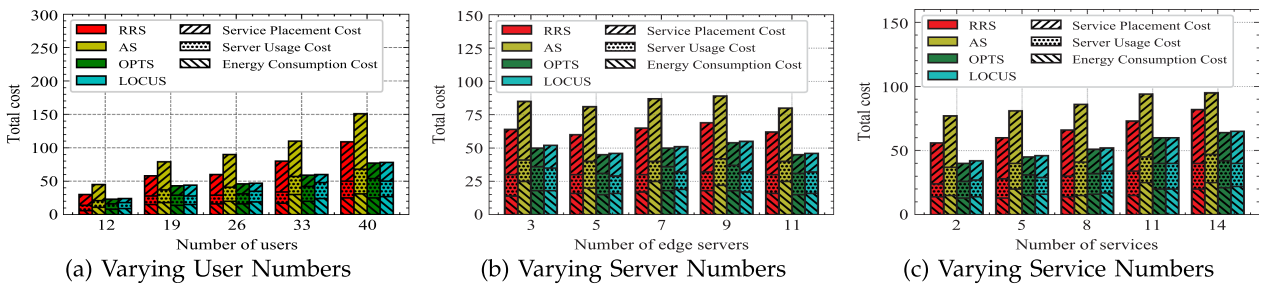


Fig. 7. Three components in total cost with varying numbers of users, servers and services.

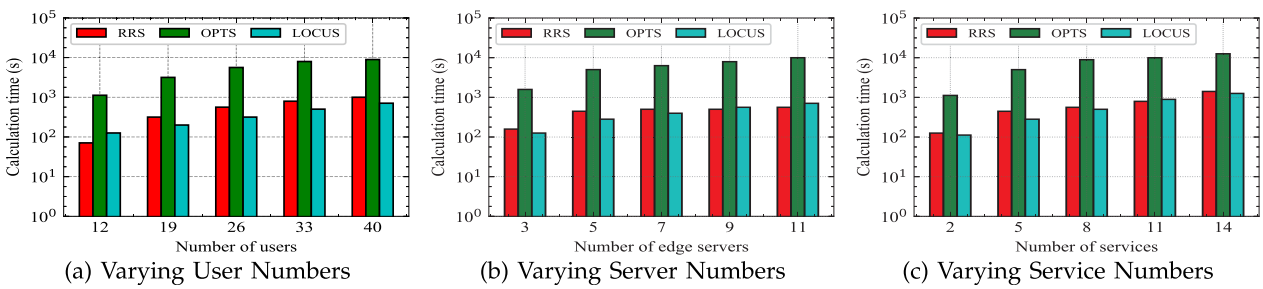


Fig. 8. Time taken by different schemes with varying numbers of users, servers and services.

algorithm with existing methods and show the good performance of LOCUS through experiments. However, there are still a few limitations in our work which needs future research effort. First, more accurate system model design (e.g., considering the delay of task queuing at the edge server) would help to determine the better service placement and user allocation decision. Second, applying our proposed algorithm to dynamic service placement requires extra handling of the service placement decisions in different time slots.

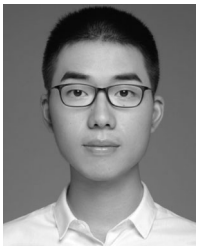
Acknowledgment

We thank the anonymous reviewers and our editor Y. Yang.

References

[1] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Springer J. Internet Services Appl.*, vol. 1, no. 1, pp. 7–18, 2010.
[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.
[4] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Gener. Comput. Syst.*, vol. 70, pp. 59–63, 2017.
[5] A. Ndikumana *et al.*, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.
[6] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.
[7] 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.
[8] Q. Althebyan, Q. Yaseen, Y. Jararweh, and M. Al-Ayyoub, "Cloud support for large scale e-healthcare systems," *Springer Ann. Telecommun.*, vol. 71, no. 9, pp. 503–515, 2016.
[9] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.
[10] P. Lai *et al.*, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Int. Conf. Service-Oriented Comput.*, 2018, pp. 230–245.
[11] X. Cao, J. Zhang, and H. V. Poor, "An optimal auction mechanism for mobile edge caching," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 388–399.
[12] S. Pasteris, S. Wang, M. Herbster, and T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 514–522.
[13] 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 *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1459–1467.
[14] V. Farhadi *et al.*, "Service placement and request scheduling for data-intensive applications in edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1279–1287.
[15] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1468–1476.
[16] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377–390, Feb. 2021.
[17] 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 *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2066–2075.
[18] P. Lai *et al.*, "Quality of experience-aware user allocation in edge computing systems: A potential game," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 223–233.
[19] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jun. 2020.
[20] J. Zou, T. Hao, C. Yu, and H. Jin, "A3C-DO: A regional resource scheduling framework based on deep reinforcement learning in edge scenario," *IEEE Trans. Comput.*, vol. 70, no. 2, pp. 228–239, Feb. 2021.
[21] M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Heterogeneous edge offloading with incomplete information: A minority game approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2139–2154, Sep. 2020.
[22] R. Behravesh, D. Harutyunyan, E. Coronado, and R. Riggio, "Time-sensitive mobile user association and SFC placement in MEC-enabled 5G networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 3006–3020, Sep. 2021.
[23] 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 *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 10–18.
[24] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet of Things J.*, vol. 6, no. 3, pp. 5853–5863, Jun. 2019.
[25] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," *Wiley Online Library Concurrency Comput.: Practice Experience*, vol. 29, no. 16, 2017, Art. no. e3975.
[26] T. X. Tran, K. Chan, and D. Pompili, "COSTA: Cost-aware service caching and task offloading assignment in mobile-edge computing," in *Proc. 16th Annu. IEEE Int. Conf. Sens. Commun. Netw.*, 2019, pp. 1–9.
[27] G. Huang *et al.*, "ShuffleDog: Characterizing and adapting user-perceived latency of android apps," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2913–2926, Oct. 2017.
[28] S. Zhang, L. Huang, M. Chen, and X. Liu, "Proactive serving decreases user delay exponentially: The light-tailed service time case," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 708–723, Apr. 2017.
[29] W. Jing, X. Wen, Z. Lu, and H. Zhang, "User-centric delay-aware joint caching and user association optimization in cache-enabled wireless networks," *IEEE Access*, vol. 7, pp. 74 961–74 972, 2019.
[30] Z. Huang, B. Hu, and J. Pan, "Caching by user preference with delayed feedback for heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1655–1667, Mar. 2021.
[31] M. Naphade *et al.* "The 2019 AI city challenge," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 452–460.
[32] L. Chen and J. Xu, "Collaborative service caching for edge computing in dense small cell networks," *CoRR*, vol. abs/1709.08662, 2017.
[33] J. Yan, N. Li, Z. Zhang, A. X. Liu, J. F. Martinez, and X. Yuan, "Game theory based joint task offloading and resources allocation algorithm for mobile edge computing," *CoRR*, vol. abs/1912.07599, 2019.
[34] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[35] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 64–71, Aug. 2017.
[36] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[37] D. Astély, E. Dahlman, A. Furuskär, Y. Jading, M. Lindström, and S. Parkvall, "LTE: The evolution of mobile broadband," *IEEE Commun. Mag.*, vol. 47, no. 4, pp. 44–51, Apr. 2009.
[38] N. Li, J. Martinez-Ortega, and G. Rubio, "Distributed joint offloading decision and resource allocation for multi-user mobile edge computing: A game theory approach," *CoRR*, vol. abs/1805.02182, 2018.
[39] N. Li, J.-F. Martinez-Ortega, and V. H. Diaz, "Distributed power control for interference-aware multi-user mobile edge computing: A game theory approach," *IEEE Access*, vol. 6, pp. 36 105–36 114, 2018.
[40] I.-H. Hou, T. Zhao, S. Wang, and K. Chan, "Asymptotically optimal algorithm for online reconfiguration of edge-clouds," in *Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2016, pp. 291–300.
[41] D. Popescu, N. Zilberman, and A. Moore, "Characterizing the impact of network latency on cloud-based applications' performance," Univ. Cambridge, Comput. Lab., Tech. Rep. UCAM-CL-TR-914, Nov. 2017, doi: 10.17863/CAM.17588.
[42] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1405–1418, Jun. 2020.

[43] G. Cornuéjols, R. Sridharan, and J.-M. Thizy, "A comparison of heuristics and relaxations for the capacitated plant location problem," *Elsevier Eur. J. Oper. Res.*, vol. 50, no. 3, pp. 280–297, 1991.
[44] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Analysis of a local search heuristic for facility location problems," *Elsevier J. Algorithms*, vol. 37, no. 1, pp. 146–188, 2000.
[45] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.
[46] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.



Yu Chen received the BS degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2019, where he is currently working toward the PhD degree under the supervision of associate professor Sheng Zhang. He is a member of the State Key Laboratory for Novel Software Technology. He has published two papers including IEEE INFOCOM 2020 and IEEE ICPADS 2020. Currently, his research interests include mobile edge computing and game theory.

Sheng Zhang (Member, IEEE) received the BS and PhD degrees from Nanjing University, Nanjing, China, in 2008 and 2014, respectively. He is currently an associate professor with the Department of Computer Science and Technology, Nanjing University, China. He is also a member of the State Key Laboratory for Novel Software Technology. His research interests include cloud computing and edge computing. To date, he has published more than 80 papers, including those appeared in the *IEEE Transactions on Mobile Computing*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, MobiHoc, ICDCS, INFOCOM, SECON, IWQoS, and ICPP. He received the Best Paper Award of IEEE ICCCN 2020 and the Best Paper Runner-Up Award of IEEE MASS 2012. He is the recipient of 2015 ACM China Doctoral Dissertation Nomination Award. He is a senior member of the CCF.

Yibo Jin (Member, IEEE) received the BS degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2017, where he is currently working toward the PhD degree under the supervision of Professor Sanglu Lu. He was a visiting student with the Hong Kong Polytechnic University, Hong Kong, in 2017. His research interests include big data analytics, edge computing, and distributed machine learning.

Zhuzhong Qian (Member, IEEE) received the PhD degree in computer science, in 2007. He is currently an associate professor with the Department of Computer Science and Technology, Nanjing University, China. His research interests include cloud computing, distributed systems, and pervasive computing. He is the chief member of several national research projects on cloud computing and pervasive computing. He has published more than 30 research papers in related fields.

Mingjun Xiao (Member, IEEE) received the PhD degree from the University of Science and Technology of China (USTC), Hefei, China, in 2004. He is currently a professor with the School of Computer Science and Technology, University of Science and Technology of China. His research interests include mobile crowdsensing, blockchain, edge computing, mobile social networks, vehicular ad hoc networks, auction theory, data security, and privacy. He has published more than 90 papers in referred journals and conferences, including the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Mobile Computing*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Services Computing*, INFOCOM, ICNP, ICDCS, etc. He served as the TPC member of many top conferences, including INFOCOM20, INFOCOM19, ICDCS19, DASFAA19, INFOCOM18, etc. He is on the reviewer board of several top journals such as *IEEE Transactions on Mobile Computing*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Vehicular Technology*, IEEE Transactions on Cloud Computing, etc.

Jidong Ge (Member, IEEE) received the PhD degree in computer science from Nanjing University, Nanjing, China, in 2007. He is currently an associate professor with the Software Institute, Nanjing University. His research interests include cloud computing and edge computing, machine learning and reinforcement learning, process mining. His research results have been published in more than 90 papers in international journals and conference proceedings including the *IEEE Transactions on Mobile Computing*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Parallel and Distributed Systems*, *ACM Transactions on Knowledge Discovery from Data*, *IEEE Transactions on Services Computing*, *Journal of Parallel and Distributed Computing*, *Journal of the American Society of Echocardiography*, *Journal of Network and Computer Applications*, *Future Generation Computer Systems*, *Journal of Systems and Software*, *Information Sciences*, ESA, ExpSys, ICSE, ASE, AAAI, EMNLP, IWQoS etc.

Sanglu Lu (Member, IEEE) received the BS, MS, and PhD degrees from Nanjing University, Nanjing, China, in 1992, 1995, and 1997, respectively, all in computer science. She is currently a professor with the Department of Computer Science and Technology and the State Key Laboratory for Novel Software Technology. Her research interests include distributed computing, wireless networks, and pervasive computing. She has published more than 80 papers in referred journals and conferences in the above areas.