Online Optimal Service Selection, Resource Allocation and Task Offloading for Multi-Access Edge Computing: A Utility-Based Approach

Weibo Chu, Peijie Yu, Zhiwen Yu, John C.S. Lui, and Yi Lin

Abstract—Multi-access edge computing promises satisfactory user experience by offloading tasks to the MEC server deployed at the network edge. However, since the MEC server is often resource-limited as compared to the cloud infrastructure, how to efficiently utilize its resources for system performance optimization becomes a challenge. In this paper, we study this problem with the aim at maximizing user's QoE through jointly optimizing service selection, computation resource allocation and task offloading decision, which is less studied in existing literature. We formulate a mixed-integer nonlinear programming problem (MINLP) for the task and propose a utility-based approach together with a low-complexity resource-efficiency based heuristic to address the problem. We consider realistic settings, where centralized solutions may not apply and an optimal mechanism needs to adapt as system operates. A distributed algorithm based on the Lagrangian-dual based decomposition theory is proposed, and we prove all sub-problems derived can be efficiently solved. In line with the current VM technology, we develop a cost-aware online algorithm that explicitly incorporates the cost of service switches into service selection and resource allocation. We evaluate our mechanism through both synthetic and trace-driven simulations, and results indicate they are effective as compared to representative baseline algorithms.

Index Terms—Multi-access edge computing, service selection, computation resource allocation, task offloading, online algorithm

1 Introduction

Mobile applications have dominated the Internet for the past few years [13] and they are now becoming more and more resource-hungry and delay-sensitive. Applications such as augmented reality and interactive online gaming are common today and they impose a stringent requirement on both computing and networking capacity. Cloud computing, which is the *de-facto* platform for providing network computing services, nevertheless, is insufficient at supporting these applications due to the long distance (hence long delay) between cloud data center and user equipment (UE). The context-awareness of computation [32], [29] also requires workload being processed as locally as possible. These issues have led to a call for alternative network computing paradigms that bring computation and storage capacities of cloud in close proximity to end-users, giving rise to the concept known as Multi-access Edge Computing (MEC, formally known as Mobile Edge Computing [26]).

- *Weibo Chu, Peijie Yu, Zhiwen Yu, and Yi Lin are with Northwestern Polytechnical University, Xi'an, Shaanxi 710060, China. E-mail: {wbchu, zhiwenyu, ly_cs}@nwpu.edu.cn, pjyu@mail.nwpu.edu.cn.*
- *John C.S. Lui is with The Chinese University of Hong Kong, Shatin, Hong Kong. E-mail: cslui@cse.cuhk.edu.hk.*

Manuscript received 8 July 2021; revised 10 Feb. 2022; accepted 14 Feb. 2022. Date of publication 18 Feb. 2022; date of current version 5 June 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2019YFB2102200, the National Natural Science Foundation of China under Grant 62172333 and the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2021JM-073. The work of John C.S. Lui was supported in part by the GRF under Grant 14200321.
(Corresponding author: Weibo Chu.)
Recommended for acceptance by X. Costa-Perez.
Digital Object Identifier no. 10.1109/TMC.2022.3152493

MEC offers superior user experience by offloading resource-intensive computation from mobile devices to the MEC servers deployed at the network edge, i.e., base stations. However, as compared to the cloud infrastructure, the MEC server is often resource-limited due to constraints of physical space, power supply, etc. For example, it is common that network operators deploy a computing sever with small computation resources or a cluster with medium resources to implement cloudlet based mobile computing. This resource-limitation at the MEC server poses significant challenges for system operators to optimize performance since not all services can be simultaneously hosted at the network edge, and how to optimally select services to accommodate at the MEC server then becomes a critical research problem [48], [40].

Designing an optimal policy for service selection faces many challenges. First, services are heterogeneous in both resource requirement (i.e., CPU, memory) and workload/demand from users. A training task (i.e., updating DNN models) from IoT sensors typically consumes more computation resources but can tolerate long delays than a face recognition request from mobile phones. To achieve high system performance, characteristics of services and the corresponding tasks from UEs should be appropriately taken into account during the service selection process. Second, whereas services are often fixed in system, tasks from UEs and system conditions (i.e., channel quality) are changing over time. This requires that service selection needs to adapt to these changes as well. Third, as dynamic service selection is involved in a stochastic MEC system, service switches are unavoidable when some new services "replace" the existing ones. Service switches are generally not cost-free as they may significantly degrade

system performance, i.e., certain amount of time is needed to download a new service/VM from cloud and then start it at the MEC server, during which tasks cannot be served. The cost of service switches requires that optimal service selection policy be *cost-aware*, i.e., it achieves high system performance while at the same time incurs less service switches.

In this paper, we study the service selection[1] problem in a general MEC system with limited resources at the MEC server, i.e., an Edge Cloud Node (ECN) in a wireless local area network [16], and our goal is to optimize user experience. We formally define user's QoE (quality of experience) as the weighted sum of task latency reduction and energy savings at UEs, and formulate an optimization problem for the task through jointly optimizing service selection, computation resource allocation and task offloading decision, which are tightly coupled if one wants to optimize system performance. The problem is NP-hard in general and we propose a utility-based reformulation and then develop a low-complexity efficient heuristic to solve it. We also devise a distributed and online algorithm that adapts to both workload and system variations, which at the same time explicitly integrates the cost of service switches that cannot be simply neglected in a real MEC system. Simulation and numerical results demonstrate that our proposed mechanism is capable at improving user's QoE while at the same time keeping the system stable, as compared to representative baseline algorithms. For example, it is observed that our cost-aware online algorithm achieves more than 50% performance enhancement than the Top-Rate-Allocation algorithm (which is the most efficient among centralized baseline algorithms), while incurs less service switches under real-world task workloads.

More specifically, we make the following contributions:

(1) We propose a fine-grained task offloading policy by dividing tasks of each service into multiple sub-types based on their workload characteristics, i.e., task data size, computation intensity, transmission rate between UE and the MEC server, and making task offloading decisions based on these sub-types. This policy allows us to fully leverage the heterogeneity of tasks for system performance optimization compared to other policies such as service-level task offloading [48], while at the same time keeps the model computationally tractable, i.e., there are a small number of task offloading decision variables in the model.

(2) We study the problem of maximizing user's QoE by jointly optimizing service selection, VM resource allocation and task offloading decision. We formulate the joint optimization task as a mixed-integer nonlinear programming problem (MINLP) and prove its NP-hardness. To solve it efficiently, we reformulate the problem as a network utility maximization problem (NUM) with a cardinality constraint which is much simpler in form, and then propose a low-complexity *resource-efficiency* based heuristic for solving this NUM problem.

(3) We analyze structural properties of the joint optimization problem and develop a decentralized algorithm by decomposing it into sub-problems for the service providers and the MEC server, with the Lagrangian-dual based decomposition theory. We prove that all sub-problems derived can be efficiently solved and that the algorithm converges.

(4) The joint optimization task is further studied in an online setting, where service/VM switches at the MEC server can significantly impact system performance. We formally define the cost of service switches, and propose a *cost-aware* online algorithm for optimal service selection and resource allocation.

(5) We evaluate our mechanism through both synthetic and trace-driven simulations. Results indicate that as compared to baseline algorithms, our *resource-efficiency* based heuristic significantly improves system performance, and the cost-aware online algorithm converges fast and that it is capable of achieving high performance while at the same time effectively avoids system instability, i.e., a slight amount of VM switches is incurred as system operates.

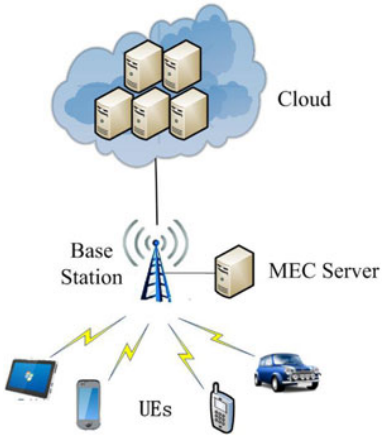


Fig. 1. A simple multi-access edge computing system.

The remainder of this paper is organized as follows. In Section 2 we introduce system model and the problem we concentrate. Section 3 gives problem formulation for the joint optimization task and its alternative NUM representation. We also present the resource-efficiency based heuristic in this section. Section 4 describes our decentralized mechanism and Section 5 elaborates the cost-aware online algorithm. Section 6 presents numerical results via simulations. We discuss future research directions and related work in Sections 7 and 8 concludes the paper.

2 Problem Description

We consider a generic Multi-access Edge Computing system as shown in Fig. 1, where a set of collocated User Equipments (UEs) (e.g., smart phones, IoT sensors, unmanned vehicles) are served by a base station. To generalize the model, this base station can be either macro cell (eNB), small cell (SCeNB), or femto cell (HeNB). Integrated with the base station is a set of servers (called MEC server) deployed by the mobile network operator so that computation-intensive and latency-sensitive tasks from UEs can be offloaded to the server for better network performance and enhanced users' quality of experience (QoE).

1. Sometimes also referred to as service placement or service caching in existing literature.

Without loss of generality, we assume there are N different type of tasks (e.g., image processing, navigation) generated by UEs in the system. Each type i can be further categorized into N_i sub-types based on the parameters of tasks (e.g., data size, computation intensity, transmission rate between UE and the base station). Unlike previous work that allocate computing resources of the MEC server to UEs [25], [2] or tasks [52], in this work, we consider allocating computing resources to individual services that are hosted by virtual machines (VMs), each of which is capable of performing computation for one specific type of task. For example, navigation for unmanned vehicles and face recognition for mobile users are hosted by two separate VMs for security consideration. It is assumed that not all services/VMs[2] are hosted simultaneously at the MEC server due to its resource (memory, storage, etc) constraints.

Computation tasks are generated by UEs. Once a computation task is generated, the request is routed to the MEC server for potential offloading. The computation can only be performed at the server if the corresponding service is hosted and significant performance gains can be achieved. Otherwise, the task is computed locally at the UE where it originates.[3]

With the above setting, three questions arise naturally as how to utilize the computing resource of the MEC server and how tasks are computed. More specifically, these questions are: *1) which services/VMs should be hosted at the MEC server? – also known as the service selection problem, 2) how much resources should be allocated to these services? – also known as the VM resource allocation problem, and 3) which computation tasks should be offloaded to the MEC server so that significant performance gains can be achieved? – also known as the task offloading decision problem*.

Obviously, different answers to the above questions will lead to different system performance and users' QoE. Moreover, it is not hard to see that the three design choices (service selection, VM resource allocation and task offloading) are tightly coupled if we want to optimize the system performance, i.e., computation resources should be allocated to those services such that significant performance gains can be achieved by offloading their tasks to the server. To fully realize the potential of MEC, a *joint optimization* of service selection, resource allocation and task offloading decision is thus required. It is also worth noting that as the characteristics of tasks and network conditions fluctuate over time, in practice we need an online and adaptive (possibly distributed) algorithmic solution.

In this paper, we seek to answer the aforementioned questions by studying the problem of maximizing users' QoE, with both latency and energy consumption of processing computation workloads taken into account. Starting from models to characterize latency and energy consumption in MEC, we formulate the joint optimization problem as a mixed-integer nonlinear programming problem (MINLP) and prove its NP-hardness. We show that the problem can be reformulated as a network utility maximization (NUM) problem with a tractable form. This NUM problem, although still challenging to solve, facilitates us to design efficient heuristic algorithm by exploiting its structural properties. The joint optimization task is further studied under decentralized and online settings, for which we also develop efficient algorithmic solutions.

2. We use services and VMs interchangeably.

3. We assume in this work that tasks will not be offloaded to cloud due to the excessive long propagation delay.

TABLE 1
Main Notations

Symbol	Definition
N	Number of services in system
M	Number of services that can be hosted at the MEC server
F	CPU frequency at the MEC server
F^{MAX}	Maximal CPU frequency that can be allocated to a service
i	Index of task types
j	Index of sub-types of tasks
A_{ij}	A task of type ij (sub-type j in task of type i)
λ_{ij}	Arrival rate of task A_{ij}
L_{ij}	Data size of task A_{ij}
C_{ij}	Computation intensity of task A_{ij}
f_{ij}	CPU frequency of the UE where task A_{ij} originates
p_{ij}	Transmission power of the UE where task A_{ij} originates
K_{ij}	Energy coefficient of the UE where A_{ij} originates
w_{ij}	Data rate between the UE where A_{ij} originates and the MEC server
T_{ij}^{Local}	Latency of task A_{ij} if it is computed locally at UE
T_{ij}^{MEC}	Latency of task A_{ij} if it is computed at the MEC server
E_{ij}^{Local}	Energy consumption of performing task A_{ij} at UE
E_{ij}^{MEC}	Energy consumption at UE if task A_{ij} is computed at the MEC server
x_{ij}	A binary variable denoting whether task A_{ij} is offloaded
y_i	A binary variable denoting whether service/VM i is hosted at the MEC server
F_i	Computation resource allocated to service i

3 Problem Formulation and Heuristic Algorithm

In this section, we present computation models of UEs and the MEC server, i.e., models to characterize latency and energy consumption when computation is performed at UE and the MEC server. We then give problem formulation for the joint service selection, VM resource allocation and task offloading decision based on these models. A network utility maximization reformulation of the problem is presented and a low-complexity heuristic algorithm to efficiently solve the problem follows then.

3.1 Problem Formulation

We start by analyzing the problem when all parameters are fixed and are given a prior. Let λ_i and λ_{ij} be the arrival rate of tasks of type i and that of tasks of type ij (sub-type j in type i), respectively, $\lambda_i = \sum_{j=1}^{N_i} \lambda_{ij}$. We make no assumptions on λ_i and λ_{ij}, i.e., the task arrivals can be Poisson or any other processes. Each task A_{ij} can be characterized by a tuple $A_{ij}(L_{ij}, C_{ij}, f_{ij}, p_{ij}, K_{ij}, w_{ij})$, where L_{ij} is the data size of the task, C_{ij} is the required computation intensity (number of CPU cycles per bit); f_{ij}, p_{ij} and K_{ij} are CPU frequency, transmission power and energy coefficient of the UE where A_{ij} is generated, respectively. And w_{ij} is the data transmission rate between the UE and the MEC server. Table 1 summarizes the main notations used in this paper.

When A_{ij} is computed locally, the latency T_{ij}^{Local} of the task is simply the time needed to perform computation at the UE, which is

$$T_{ij}^{\text{Local}} = \frac{L_{ij} \times C_{ij}}{f_{ij}}, \tag{1}$$

where $L_{ij} \times C_{ij}$ is the computation workload (in CPU cycles). The energy consumption E_{ij}^{Local} is [26]

$$E_{ij}^{\text{Local}} = K_{ij} \times (L_{ij}C_{ij}) \times f_{ij}^2. \tag{2}$$

Let F and F_i be the CPU resource of the MEC server and that allocated to service/VM i, respectively. When task A_{ij} is offloaded to the MEC server, the latency T_{ij}^{MEC} (which includes the time to upload the data and perform computation at the server[4]) can be given as follows:

$$T_{ij}^{\text{MEC}} = \frac{L_{ij}}{w_{ij}} + \frac{L_{ij} \times C_{ij}}{F_i}, \tag{3}$$

and the energy consumption E_{ij}^{MEC} at UE is[5]

$$E_{ij}^{\text{MEC}} = p_{ij} \times \frac{L_{ij}}{w_{ij}}. \tag{4}$$

Since in MEC, users' QoE is mainly related to latency and energy consumption, we define the gain G_{ij} of offloading task A_{ij} to the MEC server as the weighted sum of latency reduction and energy savings, which is as follows:

$$G_{ij}(F_i) = \alpha_{ij} \times \frac{E_{ij}^{\text{Local}} - E_{ij}^{\text{MEC}}}{E_{ij}^{\text{Local}}} + \beta_{ij} \times \frac{T_{ij}^{\text{Local}} - T_{ij}^{\text{MEC}}}{T_{ij}^{\text{Local}}}, \tag{5}$$

where $\alpha_{ij} \in [0,1]$ ($\beta_{ij} = 1 - \alpha_{ij}$) is a constant denoting the relative weight between energy savings and latency reduction. When α_{ij} = 1, the problem becomes that of minimizing energy consumption, while α_{ij} = 0 means the goal is to minimize task latency. Note that these two weights can vary from task to task.

Let $x_{ij} \in \{0,1\}$ be a variable denoting whether task A_{ij} is offloaded, with $x_{ij} = 1$ if A_{ij} is computed at the MEC server, and $x_{ij} = 0$ otherwise. Likewise, let $y_i \in \{0,1\}$ denote whether service/VM i is hosted at the MEC server, with $y_i = 1$ if VM i is present, and $y_i = 0$ otherwise. Also denote M as the number of VMs that the MEC server can accommodate due to its resource constraint, and F^{MAX} as the maximum CPU frequency that a single VM can be allocated to. With these notations, we can formulate the offline joint service selection, VM resource allocation and task offloading decision problem with the goal to maximize users' QoE as the following optimization problem

$$\text{Max:} \quad \sum_{i=1}^{N} \sum_{j=1}^{N_i} \lambda_{ij} \times G_{ij}(F_i) \times x_{ij} \tag{6a}$$

$$\text{s.t.:} \quad \sum_{i=1}^{N} F_i \le F, \tag{6b}$$

$$\sum_{i=1}^{N} y_i \le M, \tag{6c}$$

$$x_{ij} \le y_i, \forall i, j, \tag{6d}$$

$$0 \le F_i \le F^{\text{MAX}}, \forall i \tag{6e}$$

$$y_i \in \{0,1\}, x_{ij} \in \{0,1\}, \forall i, j, \tag{6f}$$

where y_i, F_i, and x_{ij} are decision variables that correspond to service selection, resource allocation and task offloading decision, respectively. Constraint (6b) and (6c) represent resource limitation at the MEC server, and (6d) states that a task can only be offloaded to the server if the corresponding service is hosted.

The above problem is in fact a mixed-integer nonlinear programming (MINLP) problem that is typically hard to solve. In general, it is NP-hard as shown in the following theorem.

Theorem 3.1. *The offline joint optimization problem as stated in (6) is NP-hard.*

Proof. We prove the theorem by showing that the following problem (7), which is a simplified version of problem (6) (by dropping the cardinality constraint), is NP-hard

$$\text{Max:} \quad \sum_{i=1}^{N} \sum_{j=1}^{N_i} \lambda_{ij} \times G_{ij}(F_i) \times x_{ij} \tag{7a}$$

$$\text{s.t.:} \quad \sum_{i=1}^{N} F_i \le F, \tag{7b}$$

$$0 \le F_i \le F^{\text{MAX}}, \forall i \tag{7c}$$

$$x_{ij} \in \{0,1\}, \forall i, j \tag{7d}$$

For the above problem, constraint (7d) can be relaxed as $0 \le x_{ij} \le 1$ since this does not change the optimal solution. Moreover, based on Eq. (5), we can see that the objective of the problem can be expressed as : $\min\ g(\boldsymbol{z}) = h(\boldsymbol{z}) + \sum_i \frac{f_i(\boldsymbol{z})}{h_i(\boldsymbol{z})}$ with the variable $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{F})^{\text{T}}$ and feasible solution set being a compact convex set, where $\boldsymbol{x} = (x_{11}, x_{12}, \ldots)^{\text{T}}$, $\boldsymbol{F} = (F_1, F_2, \ldots)^{\text{T}}$, $h(\boldsymbol{z}) = \sum_{i=1}^{N} \sum_{j=1}^{N_i} \lambda_{ij} (\frac{\alpha_{ij} p_{ij} L_{ij}}{w_{ij} E_{ij}^{\text{Local}}} + \frac{\beta_{ij} L_{ij}}{w_{ij} T_{ij}^{\text{Local}}} - 1) x_{ij}$ is a convex function (linear), $f_i(\boldsymbol{z}) = \sum_{j=1}^{N_i} \frac{\lambda_{ij} \beta_{ij} L_{ij} C_{ij}}{T_{ij}^{\text{Local}}} x_{ij} \ge 0$ is convex (linear) and $h_i(\boldsymbol{z}) = F_i > 0$ is concave. This kind of problem, namely, the Sum-of-Ratios Problem, has been shown NP-hard [14].

Next, we prove NP-hardness of problem (6) by contradiction. Suppose there exists a polynomial-time algorithm that solves problem (6). By setting $M \ge N$ and $y_i = 1\ \forall i$,

4. Following the common practice [8], we assume a negligible amount of post-processing data to be transmitted back to the UE.

5. Since our goal is to optimize users' QoE, we ignore energy consumption of the MEC server. Furthermore, although a simple energy consumption model for uploading data is adopted in this work, we note that more complex models of energy consumption at UEs, i.e., a smartphone in 4G LTE networks [17], can also be handled by our framework and algorithms proposed.

we are able to solve problem (7) in polynomial-time, and this contradicts the conclusion that (7) is NP-hard. □

3.2 Alternative Problem Representation

Given that problem (6) is NP-hard, we seek efficient heuristic algorithms for approximate solution. We make two observations from Eq. (5): 1) the gain $G_{ij}(F_i)$ of offloading task A_{ij} to the MEC server is an increasing and concave function of the allocated resource F_i, and 2) positive gains can only be achieved when adequate resource is allocated. Let F_{ij}^0 be the resource allocation such that $G_{ij}(F_{ij}^0) = 0$, i.e., $G_{ij}(F_i) > 0$ when $F_i > F_{ij}^0$, and $G_{ij}(F_i) < 0$ when $F_i < F_{ij}^0$. We define a utility function $U_{ij}(F_i)$ for task A_{ij} as follows:

$$U_{ij}(F_i) = \begin{cases} 0, & F_i < F_{ij}^0 \\ G_{ij}(F_i), & F_i \geq F_{ij}^0 \end{cases}, \tag{8}$$

and a utility function $U_i(F_i)$ for tasks of type i

$$U_i(F_i) = \sum_{j=1}^{N_i} \lambda_{ij} U_{ij}(F_i). \tag{9}$$

It is not hard to see that $U_i(F_i)$ is a nonnegative and increasing function of F_i.

Let $\boldsymbol{F} = (F_1, F_2, \ldots, F_N)$, and $\mathrm{Card}(\boldsymbol{F})$ be the cardinality function of $\boldsymbol{F}$, i.e., it equals to the number of non-zero elements in $\boldsymbol{F}$. We can now reformulate Problem (6) as the following network utility maximization problem (NUM) with a cardinality constraint

$$\text{Max:} \quad \sum_{i=1}^{N} U_i(F_i) \tag{10a}$$

$$\text{s.t.:} \quad \sum_{i=1}^{N} F_i \leq F, \tag{10b}$$

$$0 \leq F_i \leq F^{\mathrm{MAX}}, \forall i \tag{10c}$$

$$\mathrm{Card}(\boldsymbol{F}) \leq M, \tag{10d}$$

where the cardinality constraint (10d) represents the constraint that the MEC server can host M services at most.

Theorem 3.2. *Let (F_i^*, y_i^*, x_{ij}^*) and $(F_i^{\#})$ be the optimal solution to Problem (6) and (10), respectively. Then $F_i^* = F_i^{\#}$. Moreover, the two problems have the same maximum objective function value, i.e., $\sum_{i=1}^{N}\sum_{j=1}^{N_i} \lambda_{ij} \times G_{ij}(F_i^*) \times x_{ij}^* = \sum_{i=1}^{N} U_i(F_i^{\#})$.*

Proof. It is not hard to see that $y_i^* = 0$ iff $F_i^* = 0$. This is because $y_i^* = 0$ if $F_i^* = 0$, since hosting a service with no computing resource allocated brings no performance gains but it takes a place to provide service at the MEC server. Likewise $F_i^* = 0$ if $y_i^* = 0$, due to the fact that allocating computing resource to a service not hosted at the server also brings no performance gains but merely wastes the allocated resource. Moreover, according to (6d), we have $x_{ij}^* = 0$ if $y_i^* = 0$ (or equivalently $F_i^* = 0$), and therefore $G_{ij}(F_i^*)x_{ij}^* = 0$ if $F_i^* = 0$. To maximize the objective function, we also have $G_{ij}(F_i^*) > 0$ and $x_{ij}^* = 1$ if $F_i^* > 0$.

Now we prove that (F_i^*) is a feasible solution to Problem (10), i.e., it satisfies constraints (10b) (10c) (10d). This is because: 1) $\mathrm{Card}(\boldsymbol{F}^*) \leq M$ since $\sum_i y_i^* \leq M$ (from (6c)) and $y_i^* = 0$ iff $F_i^* = 0$ as we have proved (recall that $y_i^* \in \{0, 1\}$), and 2) constraint (10b) and (10c) also appear in Problem (6) (see (6b) and (6e)).

On the other hand, given that $(F_i^{\#})$ be the optimal solution to Problem (10), we can construct a solution $(F_i^{\#}, y_i^{\#}, x_{ij}^{\#})$ such that it is feasible to Problem (6) . The construction is as follows: 1) $y_i^{\#} = 0$ if $F_i^{\#} = 0$, $y_i^{\#} = 1$ if $F_i^{\#} > 0$, and 2) $x_{ij}^{\#} = 1$ if $F_i^{\#} > 0$ and $G_{ij}(F_i^{\#}) > 0$, and $x_{ij}^{\#} = 0$ otherwise. Then we can see that $\sum_i y_i^{\#} \leq M$ since $\mathrm{Card}(\boldsymbol{F^{\#}}) \leq M$, i.e., $(F_i^{\#}, y_i^{\#}, x_{ij}^{\#})$ satisfies constraint (6c). It is also easy to check that constraints (6b) (6d) (6f) are satisfied by $(F_i^{\#}, y_i^{\#}, x_{ij}^{\#})$. The theorem follows by observing from Eqs. (8) and (9) that $\sum_{i=1}^{N} U_i(F_i) = \sum_{i=1}^{N}\sum_{j=1}^{N_i} \lambda_{ij} \times G_{ij}(F_i)$ when $G_{ij}(F_i) > 0$, i.e., the two problems have the same objective function value. □

Remark: The proof of Theorem 3.2 also suggests a way to derive a solution to Problem (6) from a solution to problem (10), such that the two problems have the same objective function value. This allows us to obtain solution to Problem (6) by solving Problem (10), which is much simpler in form as it only contains resource allocation variables. However, as a nonlinear optimization problem with cardinality constraint, problem (10) is still very hard to solve, since even a linear optimization problem with cardinality constraint is shown NP-hard [11].

3.3 Heuristic Algorithm

In this subsection, we propose an efficient heuristic algorithm to solve problem (10) based on its structural properties and the concept of *resource efficiency*. We make the following two observations about Problem (10): 1) each utility $U_i(F_i)$ is an increasing function of F_i – the resource allocated to service i, and 2) the objective function is separable, i.e., it is the sum of individual utilities $U_i(F_i)$ which is a function of its local variable F_i.

The main idea of our algorithm is that we first solve problem (10) with no cardinality constraint, i.e., the relaxed problem, which is a utility maximization problem with continuous variables. This relaxed problem, however, is not convex. Nevertheless, we can approach the optimal solution by exploiting the structural properties, as can be seen in Procedure 1. We then adjust the derived solution such that the cardinality constraint is satisfied while at the same time, the objective function value is further increased. This is achieved in Procedure 2.

Procedure 1. Allocating Resources to the N Services.

Input: Task arrival rates λ_{ij}'s; Utility functions $U_i(F_i)$'s; Number of services/VMs N; CPU resource of the MEC server F; Change of resource ΔF;
Output: Resource allocation $\boldsymbol{F} = (F_1, F_2, \ldots, F_N)$;

```
S ← {1, 2, ..., N}
for i ∈ S do
    F_i ← F^MAX.
    ΔU_i^-(F_i) ← U_i(F_i) − U_i(F_i − ΔF).
while Σ_{i=1}^N F_i > F do
    Select j ∈ S such that ΔU_j^-(F_j) = min{ΔU_i^-(F_i)}_{i∈S}.
    F_j ← F_j − ΔF.
    if U_j(F_j) = 0 then
        F_j ← 0, S ← S \ {j}.
    else
        ΔU_j^-(F_j) ← U_j(F_j) − U_j(F_j − ΔF).
```

Procedure 2. Adjusting Resources to M Services.

Input: Resource allocation $\boldsymbol{F} = (F_1, F_2, \ldots, F_N)$; Utility functions $U_i(F_i)$'s; Number of services/VMs M; Change of resource ΔF;

Output: Resource allocation $\boldsymbol{F} = (F_1, F_2, \ldots, F_M)$;

1: Re-arrange $\boldsymbol{F}$ as $\boldsymbol{F} = (F_{[1]}, F_{[2]}, \ldots, F_{[N]})$ such that $F_{[1]} \geq F_{[2]} \geq \ldots \geq F_{[N]}$;
2: $F_{\text{available}} \leftarrow F - \sum_{i=1}^{M} F_{[i]}$, $S \leftarrow \{[1], [2], \ldots, [M]\}$.
3: **for** $i \in S$ **do**
4: $\quad \Delta U_i^+(F_i) \leftarrow U_i(F_i + \Delta F) - U_i(F_i)$.
5: **while** $F_{\text{available}} > 0$ **do**
6: $\quad$ Select $j \in S$ such that $\Delta U_j^+(F_j) = \max\{\Delta U_i^+(F_i)\}_{i \in S}$.
7: $\quad F_j \leftarrow F_j + \Delta F$.
8: $\quad F_{\text{available}} \leftarrow F_{\text{available}} - \Delta F$.
9: $\quad$ **if** $F_j \geq F^{\text{MAX}}$ **then**
10: $\quad\quad F_{\text{available}} \leftarrow F_{\text{available}} + F_j - F^{\text{MAX}}$.
11: $\quad\quad F_j \leftarrow F^{\text{MAX}}$.
12: $\quad\quad S \leftarrow S \setminus \{j\}$
13: $\quad$ **else**
14: $\quad\quad \Delta U_j^+(F_j) \leftarrow U_j(F_j + \Delta F) - U_j(F_j)$.

More specifically, our algorithm consists of two procedures. In Procedure 1, we solve the relaxed problem by first allocating the maximum allowed CPU resource to each service, that is, F^{MAX}. We then take resource from each service until the sum of allocated resource equals to that of the MEC server. Note that each time a constant amount ΔF of CPU resource is taken from a service, the service with the least *resource efficiency* is selected (see line 6). Here *resource efficiency* is defined to be the difference of utility when ΔF resource is taken, i.e., $\Delta U_i^-(F_i) = U_i(F_i) - U_i(F_i - \Delta F)$. Taking resource from a service with the least resource efficiency guarantees that the total utility (objective function value) is decreased least.

After execution of Procedure 1, we obtain computation resource allocation to the N services. We then select the top M services with the largest utilities, and allocate the remaining resource (if any from other services) to these selected M services, as shown in Procedure 2. Note that each time we add a constant amount ΔF of CPU resource to a service, we select the service with the most *resource efficiency* (line 6). Similarly, here *resource efficiency* is defined to be the difference of utility when ΔF resource is added, i.e., $\Delta U_i^+(F_i) = U_i(F_i + \Delta F) - U_i(F_i)$. This guarantees that the objective function value is increased most. The algorithm terminates when all resource of the MEC server is allocated to the M services.

We note that our algorithm works in a greedy manner, as in both procedures it adds or takes resource from the service with the most or least resource efficiency. Assuming there exists an oracle for evaluating resource efficiency, the time complexity of Procedure 1 is $O(\frac{N \times F^{\text{MAX}} - F}{\Delta F})$, and that of Procedure 2 is at most $O(\frac{F}{\Delta F})$. Therefore, the time complexity of our algorithm is $O(\frac{N \times F^{\text{MAX}} - F}{\Delta F}) + O(\frac{F}{\Delta F}) = O(\frac{N \times F^{\text{MAX}}}{\Delta F})$.

4 Decentralized Mechanism

The heuristic algorithm proposed in the above section is centralized as it takes as input detailed information about tasks and network conditions, which include task arrival rates, task parameters, UE configuration, etc. However, in practice one may need decentralized mechanisms, i.e., when each application/service provider does not want to reveal its utility function to the MEC server, or the MEC server is not able to collect information about UEs. A decentralized mechanism also adapts to network changes and workload variations naturally. In this section, we propose such a decentralized mechanism based on structural properties of the problem under consideration and the Lagrangian-dual decomposition theory, where the MEC server and service providers work collectively to determine an appropriate resource allocation.

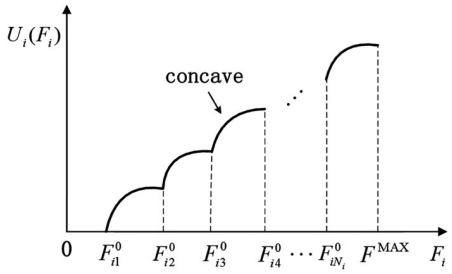


Fig. 2. Utility function $U_i(F_i)$.

4.1 Decentralized Mechanism

Similar to heuristic algorithm, the decentralized mechanism we propose also consists of two procedures, where the first procedure solves the relaxed problem (with no cardinality constraint), and the second procedure adjusts the solution so that resource of the MEC server is fully allocated to exactly M services.

In both procedures, we solve optimization problems using Lagrangian-based dual decomposition. The problem (called *primal problem*) we address in the first procedure is as follows:

$$\begin{aligned} \text{Max:} \quad & \sum_{i=1}^{N} U_i(F_i) \\ \text{s.t.:} \quad & \sum_{i=1}^{N} F_i \leq F, \\ & 0 \leq F_i \leq F^{\text{MAX}}, \forall i \end{aligned} \tag{11}$$

Let $\boldsymbol{F} = (F_1, F_2, \ldots, F_N)$ and $P \geq 0$. Here P is considered as the price of resource. Define Lagrangian

$$\begin{aligned} L(\boldsymbol{F}, P) &= \sum_{i=1}^{N} U_i(F_i) - P\left(\sum_{i=1}^{N} F_i - F\right) \\ &= \sum_{i=1}^{N} (U_i(F_i) - PF_i) + PF \end{aligned} \tag{12}$$

As the first term is separable, we have $\text{maximize}_{\boldsymbol{F}} \sum_{i=1}^{N} (U_i(F_i) - PF_i) = \sum_{i=1}^{N} \text{maximize}_{F_i} (U_i(F_i) - PF_i)$. According to [31], we can now readily formulate optimization problem for each service provider i as follows:

Service Provider i's Problem.

$$\begin{aligned} \text{Max:} \quad & U_i(F_i) - P \times F_i \\ s.t.: \quad & 0 \leq F_i \leq F^{\text{MAX}} \end{aligned} \tag{13}$$

Theorem 4.1. *Problem (13) can be efficiently solved, i.e, there exists a polynomial-time algorithm to it.*

Proof. Let F_{ij}^0 be the root of equation $G_{ij}(F_i) = 0$, $j = 1, 2, \ldots, N_i$. Without loss of generality, assume that $F_{i1}^0 < F_{i2}^0 < \ldots < F_{iN_i}^0 \leq F^{\text{MAX}}$. Since $G_{ij}(F_i)$ is concave, from Eqs. (8) and (9), we can see that $U_i(F_i)$ is a piece-wise increasing concave function, i.e., $U_i(F_i)$ is increasing and concave in each $[F_{ij}^0, F_{ij+1}^0]$ (see Fig. 2). It follows that the objective function $U_i(F_i) - PF_i$ is also concave in $[F_{ij}^0, F_{ij+1}^0]$. This implies that we can solve problem (13) by solving a series of problems, each one with the decision variable constrained in $[F_{ij}^0, F_{ij+1}^0]$, and each one is a convex optimization problem. The optimal solution is the one with the largest objective function value. □

Correspondingly, the dual problem for the MEC server is:

MEC Server's Problem.

$$\begin{aligned} &\text{Min: } g(P) = \sum_{i=1}^{N} g_i(P) + PF \\ &s.t. \quad P \geq 0 \end{aligned} \tag{14}$$

where $g_i(P)$ is the maximum value of the objective function solved in problem (13) for a given P. Note that problem (14) is convex even when problem (11) is not [3], [31].

The MEC server solves problem (14) by updating the price P. Note that as $g_i(P)$ is not differentiable in P, the projected sub-gradient method is adopted. More specifically, once the MEC server receives the required amount of resource from each service provider, its price is adjusted as follows:

$$P^{t+1} = \left[P^t + \gamma \left(\sum_{i=1}^{N} F_i^t - F \right) \right]^+. \tag{15}$$

where $\gamma > 0$ is a step size, t denotes time, and $[x]^+ = max\{x, 0\}$.

We want to emphasize that: 1) it is possible there exists multiple optimal solutions to problem (13). In this case, any optimal solution can be selected by the service provider i to form a valid sub-gradient of $g_i(P)$ with respect to P [4]; 2) although the sub-gradient based approach solves problem (14), the duality gap between the optimal primal solution and the optimal dual solution is generally non-zero due to nonconcavity of utility functions. Furthermore, computation resource allocation derived may also violate the capacity constraint from the MEC server. To circumvent this problem, we propose the following rule to obtain a valid resource allocation

$$F_i^a = \begin{cases} F_i^o, & \sum_{i=1}^{N} F_i^o \leq F \\ \frac{F_i^o \times F}{\sum_{i=1}^{N} F_i^o}, & \sum_{i=1}^{N} F_i^o > F \end{cases}. \tag{16}$$

where $\{F_i^o\}$ denotes the distributed solution from service provider i and F_i^a the regulated resource allocation to the service.

In short, in the mechanism each service provider locally calculates its required resources based on the price advertised by the MEC server, and the MEC server adjusts the price based on resources required by each service provider. The solution to the primal problem (11) is obtained when this process converges.

After execution of the first procedure, we have resource allocation to each service. The second procedure then starts by selecting the top M services with the largest utilities, with the help of each service provider reporting its utility function value when the first procedure terminates. The goal of this procedure is to allocate the remaining resources to the selected services. This can be achieved in a similar way as the first procedure. Formally, let F_i^a be the amount of resource that has been allocated to service i (obtained by executing the first procedure, see Eq. (16)), and S be the set of these selected M services. The primal problem we address in the second procedure can be formulated as

$$\begin{aligned} \text{Max:} \quad & \sum_{i \in S} U_i(F_i^a + \Delta F_i) \\ \text{s.t.:} \quad & \sum_{i \in S} \Delta F_i \leq F - \sum_{i=1}^{N} F_i^a, \\ & 0 \leq \Delta F_i \leq F^{\text{MAX}} - F_i^a, \forall i \in S \end{aligned} \tag{17}$$

where ΔF_i is the decision variable denoting the amount of resources (from services not in S) to be allocated to service i. Similarly, we can decompose problem (17) into sub-problems for each service provider and the MEC server by the Lagrangian-based dual decomposition. The properties of these sub-problems are identical to (13) and (14), and can be efficiently (and locally) solved in exactly the same way. We omit these details for space limitation and conciseness.

4.2 Complexity Analysis

Here we present an analysis on the computational complexity of our decentralized mechanism, by comparing it with the canonical price-based distributed algorithm based on dual decomposition for convex NUM problems [49], [24], which has proven computationally efficient. Note that as compared to the canonical price-based distributed algorithm to convex NUM problem, the added complexity by our mechanism comprises of two parts. The first one is on the regulating operation (Eq. (16)) performed by the MEC server, which obviously incurs very low cost and thus can be neglected. The second one is on solving the non-convex optimization problem (13) by each service provider. Instead of one convex optimization problem to be solved by each service provider as in the canonical distributed algorithm, in our mechanism multiple convex optimization problems are solved by each service provider. As a result, under the same accuracy of solutions, the complexity of our mechanism is at most $2 \times N^{\text{MAX}}$ times that of the canonical distributed algorithm assuming that utility functions are all concave, where $N^{\text{MAX}} = max\{N_i\}_{i=1}^{N}$ (recall that the factor 2 is for the fact that our mechanism consists of two procedures and that the second procedure incurs less cost than the first one).

Suppose interior-point algorithm using barrier method is applied to solve the convex optimization problem. Based on [3] (Chapter 11), we know that the complexity is composed of two parts, namely, iteration complexity (outer iterations) and per-iteration computation (inner iterations or

Newton iterations) cost. The former is on the order of $\lceil\frac{\log(2/(t^0\epsilon))}{\log u}\rceil$, where u and t^0 are the so-called barrier parameters, and ϵ is the accuracy of the solution. On the other hand, the per-iteration complexity is on the order of $\frac{2(u-1-\log u)(20-8\alpha)}{\alpha\beta(1-2\alpha)^2}+6$, where α and β are the backtracking parameters in Newton's method. As a result, the complexity of solving a single convex optimization problem is $\lceil\frac{\log(2/(t^0\epsilon))}{\log u}\rceil\times(\frac{2(u-1-\log u)(20-8\alpha)}{\alpha\beta(1-2\alpha)^2}+6)$, and that of solving each service provider i's sub-problem at each round can be given as $\lceil\frac{\log(2/(t^0\epsilon))}{\log u}\rceil\times(\frac{2(u-1-\log u)(20-8\alpha)}{\alpha\beta(1-2\alpha)^2}+6)\times N_i$. The complexity for solving sub-problems by all service providers at each round is therefore at most $O(\lceil\frac{\log(2/(t^0\epsilon))}{\log u}\rceil\times(\frac{2(u-1-\log u)(20-8\alpha)}{\alpha\beta(1-2\alpha)^2}+6)\times N\times N^{\mathrm{MAX}})$.

Now let us focus on how many rounds are required by the price-based dual decomposition method to solve the original problem (11), with the inexactness of the sub-problem solvers at each round. Note that this issue have been well studied for NUM problems with convex objective functions, i.e., the algorithm converges in $O(\frac{1}{\sqrt{k}})$ from [28], [36]. We expect the same rate of convergence by our mechanism, although it still requires rigorous mathematical proofs.

5 Cost-Aware Online Solution

We now focus on the joint service selection, VM resource allocation and task offloading decision problem in an *online setting*. Given our centralized and decentralized algorithms in Sections 3 and 4 , a natural approach for online solution is to apply them directly as the system operates, i.e., time is divided into successive slots and at the end of each slot we run either of the two algorithms to obtain a new solution for the next time slot. This approach, although straightforward, will not necessarily guarantee satisfactory system performance and could probably lead to poor users' QoE, since both algorithms fail to consider the cost of service switches, i.e., it takes time for the MEC server to fetch a VM from cloud and then start the service, during which tasks from UEs can not be serviced by the server. To optimize performance and control system stability, the cost of service switches should be incorporated into the design of the algorithm. In this section, we formally define the cost of switch for a service and then propose an online algorithm that optimally select services.

5.1 Cost-Aware Service Selection

We denote time as $T=0,1,2,\ldots$ by dividing it into successive slots with equal length ΔT. At the end of each slot t, a new solution to the joint optimization problem is obtained by running the algorithm we have proposed. Let S^t and $S^{'}$ be the set of services that are hosted at time slot t and that are newly computed at the end of slot t, respectively. Note that $S^{'}$ may differ in S^t. The task of optimal service selection then becomes how to select services in $S^t\cup S^{'}$ and how much resources should be allocated to these services, so that network utilities are maximized at time slot $t+1$.

Let F_i^t and $F_i^{'}$ be the amount of computation resource allocated to service i at time t, and that is newly computed at the end of time t (potentially to be adopted for time $t+1$), respectively. Since our goal is to maximize utilities, we define cost of switching a service i at time t as the utility changes incurred by re-allocating its resources from F_i^t to $F_i^{'}$. Formally, we have the following definitions:

Definition 5.1. *We define switch of a service i at time t as allocating computation resources to service i from F_i^t to $F_i^{'}$, and the cost of switch for i at time t as the utility changes incurred by this re-allocation.*

With the above definition, we now compute the cost of switches for services in S^t and $S^{'}$. Note that $S^t\cup S^{'}$ can be divided into three non-overlapping sub-sets:

- $S^t\setminus S^{'}$, these services are hosted at time slot t, but will not be present at time $t+1$. The cost of switching for a service i in $S^t\setminus S^{'}$ can be computed as
$$C_i^1=U_i^t(F_i^t), \tag{18}$$
where $U_i^t(F_i^t)$ is the utility of service i observed at time slot t.
- $S^t\cap S^{'}$, these services are hosted at time slot t and will also be present at time $t+1$, but with different resource allocation, i.e., from F_i^t to $F_i^{'}$. The cost of switch for a service i in $S^t\cap S^{'}$ can be calculated as
$$C_i^2=U_i^t(F_i^t)-U_i^{'}(F_i^{'}), \tag{19}$$
where $U_i^{'}(F_i^{'})$ is the utility of service i computed at the end of time slot t.
- $S^{'}\setminus S^t$, these services are not hosted at time t but will be hosted at $t+1$, i.e., it will be downloaded from cloud. Denote by T_i^{Start} as the time needed to fetch and start service i. Since a service $i\in S^{'}\setminus S^t$ is unavailable during this time, the cost of switch can be expressed as follows:[6]

$$C_i^3=-U_i^{'}(F_i^{'})+\frac{T_i^{\mathrm{Start}}}{\Delta T}U_i^{'}(F_i^{'})=(\frac{T_i^{\mathrm{Start}}}{\Delta T}-1)\times U_i^{'}(F_i^{'}). \tag{20}$$

Let $z_i\in\{0,1\}$ be a binary variable denoting whether or not to perform switch for service i, with $z_i=0$ if we keep the service configuration unchanged, and otherwise we adopt the new resource allocation for service i. The task of optimal service selection at the end of each time slot t can be formulated as a cost minimization problem, as follows:

$$\text{Min: } \sum_{i\in S^t\setminus S^{'}}C_i^1\times z_i+\sum_{i\in S^t\cap S^{'}}C_i^2\times z_i+\sum_{i\in S^{'}\setminus S^t}C_i^3\times z_i \tag{21a}$$

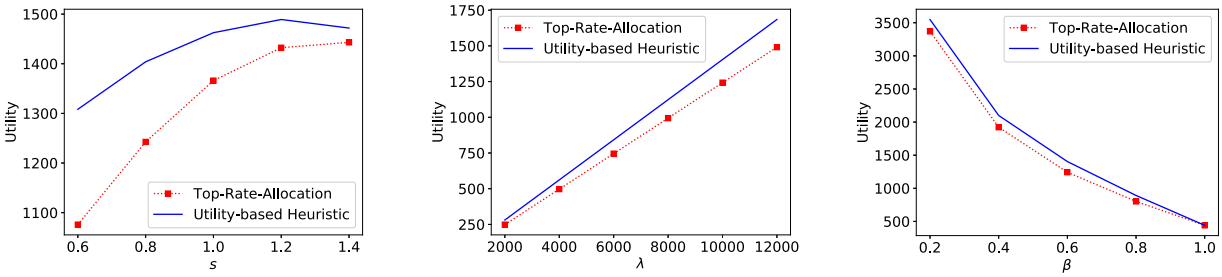
$$\text{s.t.: } \sum_{i\in S^t\cup S^{'}}(F_i^{'}-F_i^t)\times z_i\le 0, \tag{21b}$$

$$\sum_{i\in S^{'}\setminus S^t}z_i-\sum_{i\in S^t\setminus S^{'}}z_i\le 0 \tag{21c}$$

$$z_i\in\{0,1\},\forall i\in S^t\cup S^{'} \tag{21d}$$

where constraint (21b) is for the computation resource limitation at the MEC server, i.e., the amount of allocated

6. We assume the time needed to start a new service is less than the length of a time slot, i.e., $T_i^{\mathrm{Start}}\le\Delta T,\forall i$.



(a) $\beta = 0.5, \lambda = 10000$ (b) $\beta = 0.5, s = 0.8$ (c) $s = 0.8, \lambda = 10000$

Fig. 3. Performance comparison of our *centralized* heuristic algorithm and the TOP-Rate-Allocation algorithm: (a) under varying skewness parameter s; (b) under varying task arrival rate λ; (c) under varying weight parameter β.

resource for time $t+1$ should not exceed F (recall that $\sum_{i \in S^t} F_i^t = \sum_{i \in S'} F_i' = F$), and constraint (21c) is for the cardinality constraint.

Procedure 3. Online Algorithm for Selecting Services

1: Start with an initial service selection S^0 and resource allocation F^0.
2: Collect information about tasks and UEs at time slot 0.
3: Solve for S' and F' by running the centralized algorithm, i.e., Procedure 1 and 2.
4: $S^1 \leftarrow S'$, $F^1 \leftarrow F'$.
5: $t \leftarrow 1$.
6: **for** $t = 1, 2, \ldots$ **do**
7: Collect information about tasks and UEs.
8: Solve for S' and F'.
9: Obtain S^{t+1} and F^{t+1} by solving problem (21).
10: Allocate the remaining resource to services in S^{t+1} using Procedure 2, and update F^{t+1} accordingly.

Problem (21) is an integer linear programming (ILP) problem that has been well studied, and efficient algorithms, i.e., branch-and-bound, exist. Note that the solution to problem (21) does not guarantee that the server's resource is fully allocated. In that case, we allocate the remaining resource to the selected services by running the algorithm we proposed, i.e., Procedure 2, which further increases network utilities. A detailed description of the algorithm using centralized solution can be found in Procedure 3.

6 Evaluation

In this section, we perform numerical studies, first to show the efficacy of our resource-efficiency based heuristic algorithm on optimizing utilities, i.e., reducing task delays and energy consumption at UEs, and second to evaluate the performance of our cost-aware online algorithm. We use both synthetic and trace-driven simulations.

6.1 Evaluation Setup

For synthetic simulations, we assume there are 50 services in system and the MEC server can only accommodate 15 of them, i.e., $N = 50$ and $M = 15$. The number of sub-types of each task is randomly picked from $\{1, 2, 3, 4, 5\}$. CPU resource of the MEC server is set as $F = 50$ GHz, and the maximum resource allocated to each service should not exceed 10 GHz, i.e., $F^{\text{MAX}} = 10$ GHz. The change of resource ΔF in Procedures 1 and 2 is set as 1 MHz.

Task parameters are configured as follows. We assume that for each parameter there exists a set that contains its possible values, and the value of each parameter in simulation is randomly selected from the set. More specifically, in our simulation the data size of each task L_{ij} is randomly selected from [500 KB, 2000 KB, 3000 KB, 5000 KB, 10000 KB], and the computation intensity C_{ij} is selected from [100 bit/cycle, 200 bit/cycle, 300 bit/cycle, 400 bit/cycle, 500 bit/cycle]. Note that these settings represent workload of a typical face recognition application [8] leveraging MEC. Meanwhile, CPU frequency of UE f_{ij} is chosen from [0.5 GHz, 0.8 GHz, 1.0 GHz, 1.2 GHz], and the bandwidth between UE and the MEC server w_{ij} is selected from [1 Mb/s, 1.5 Mb/s, 2 Mb/s, 2.5 Mb/s, 3 Mb/s]. The transmission power of UE p_{ij} is chosen from [0.6 W, 0.8 W, 1 W, 1.2 W, 2 W], which is typical for a smartphone uploading data in commercial 4G LTE networks [17]. The energy coefficient of all UEs are set equally as $K_{ij} = K = 0.18 * 10^{-12}$.

For task arrival process, we assume tasks from UEs are generated independently according to a Poisson process and that their popularity follows a Zipf distribution. Unless otherwise specified, the skewness parameter of the Zipf distribution is set as $s = 0.8$, the overall task arrival rate is set as λ = 10000/ sec, and the weight for each task is set as $\beta_{ij} = \beta = 0.5$.

We compare our mechanism with the following four benchmarks: Random-Allocation, Top-Rate-Allocation, Fixed-Allocation and Canonical Price-based allocation. Among them, the first three are centralized while the last one is decentralized.

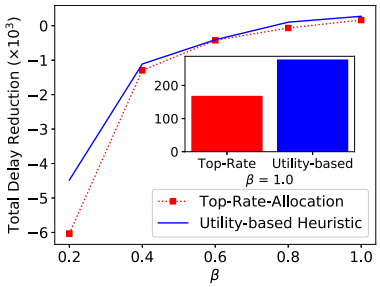
(1) *Random-Allocation*: This is the algorithm that allocates CPU resource of the MEC server to services randomly selected.

(2) *Top-Rate-Allocation*: This algorithm allocates CPU resource of the MEC server to services with the top largest arrival rate, i.e., services with the top M largest arrival rates are selected.

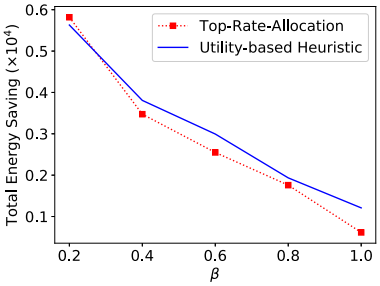
(3) *Fixed-Allocation*: This algorithm allocates CPU resource of the MEC server to services that are pre-determined.

(4) *Canonical price-based resource allocation*: This algorithm works with the assumption that all tasks from UEs will be offloaded to the MEC server, which leads to a convex optimization problem with decision variables $\boldsymbol{F} = (F_1, F_2, \ldots, F_N)$ and objective function $\sum_{i=1}^{N} \sum_{j=1}^{N_i} \lambda_{ij} \times G_{ij}(F_i)$. The standard price-based dual decomposition mechanism initially proposed for bandwidth allocation [31] is then applied.

Note that for benchmarks (1)–(3), CPU resource allocation to the selected M services is determined optimally by running our proposed resource-efficiency based heuristic,

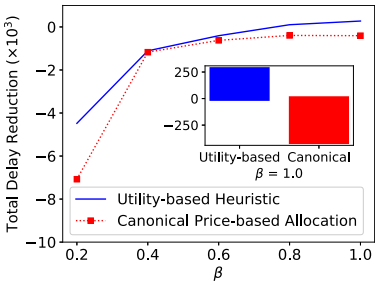


(a) Delay

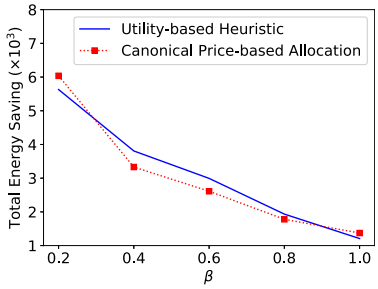


(b) Energy

Fig. 4. Delay and energy consumption achieved by *centralized* algorithms. $s = 0.8, \lambda = 10000/\text{sec}$.



(a) Delay



(b) Energy

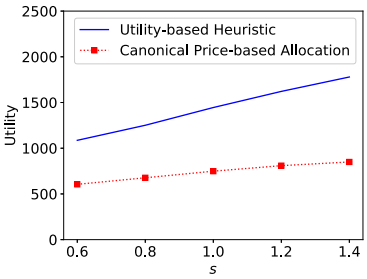
Fig. 6. Delay and energy consumption achieved by *decentralized* algorithms. $s = 0.8, \beta = 0.6$.

i.e., Procedure 1. For benchmark (4), we adopt the same two procedures to derive the distributed solution as in our decentralized mechanism. The only difference is that now the *primal* problem and the *sub-problem* for each service provider become convex.

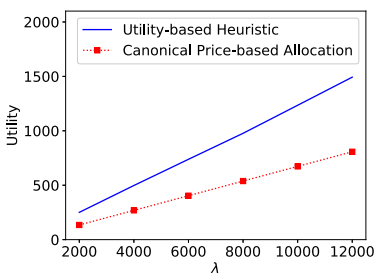
6.2 Simulation Results

6.2.1 Performance Over Synthetic Workload

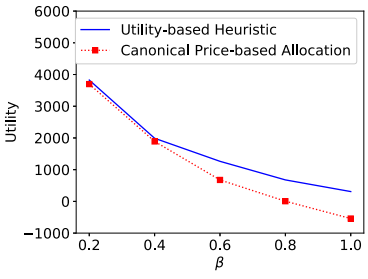
We first evaluate the performance of our resource-efficiency based heuristic (both *centralized* and *decentralized*), i.e., when task parameters are given a priori and only a single time slot is considered (therefore no cost of switch is incurred). Fig. 3 shows utilities achieved by the Top-Rate-Allocation and our centralized heuristic algorithm, under varying task arrival rate λ, skewness parameter s and weight β. From the figure, we can see that our centralized heuristic algorithm achieves more utility than the Top-Rate-Allocation algorithm, and as much as 22% improvement can be observed when $s = 0.6, \beta = 0.5$ and $\lambda = 10000/\text{sec}$. Meanwhile, the improvement decreases as the workload becomes more skewed (see Fig 3a). We believe this is due to the fact that the Top-Rate-Allocation algorithm tends to select services randomly when the skewness parameter is small, i.e., less than 1.0, since the task arrival rate tends to be uniformly distributed in that case, which then leads to a large utility improvement given that task parameters (task data size, computation intensity, etc) in simulation are randomly selected. Fig. 3b shows that as expected, the utility improvement grows linearly with the increase of the task arrival rate, since utility is a linear function in it under a fixed popularity distribution. Fig. 3c tells that the utility achieved by each algorithm can be significantly impacted by the weight parameter β, which implies that it can be used to effectively tune system performance, i.e., balancing delays and energy consumption. This property can be further observed in Fig. 4 where the actually delay reductions and energy savings of the two algorithms are depicted. More specifically, we find that more energy savings can be achieved when β is small whereas more delay reductions can be obtained when



(a) $\beta = 0.5, \lambda = 10000$

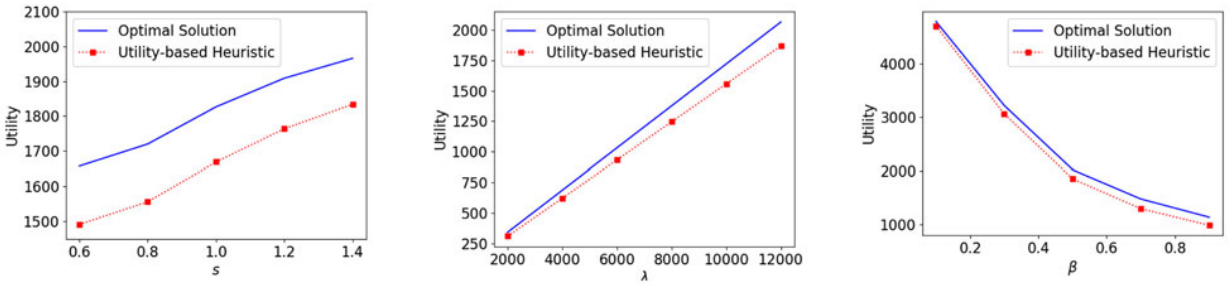


(b) $\beta = 0.5, s = 0.8$



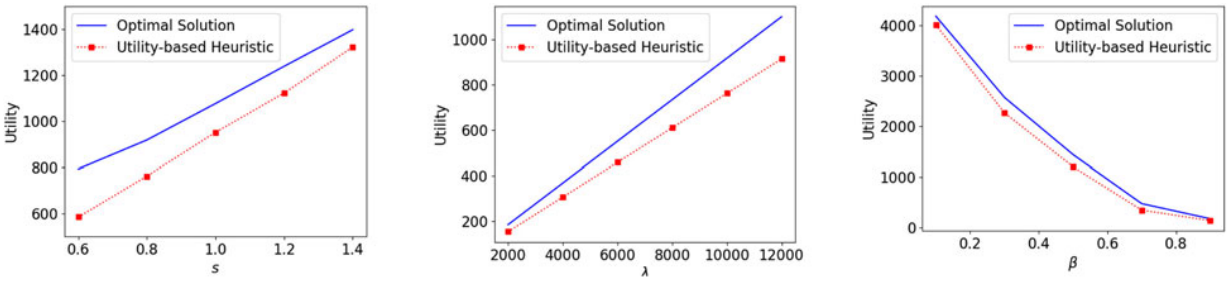
(c) $s = 0.8, \lambda = 10000$

Fig. 5. Performance comparison of our *decentralized* algorithm and the Canonical Price-based Allocation algorithm: (a) under varying skewness parameter s; (b) under varying task arrival rate λ; (c) under varying weight parameter β.



(a) $\beta = 0.6, \lambda = 10000$ (b) $\beta = 0.6, s = 0.8$ (c) $s = 0.8, \lambda = 10000$

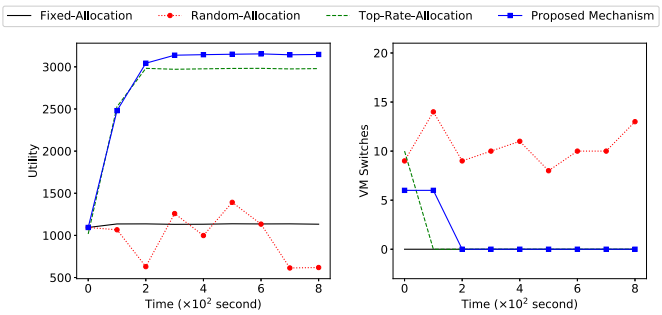
Fig. 7. Performance comparison to optimal solution ($N = 10$, $M = 3$).



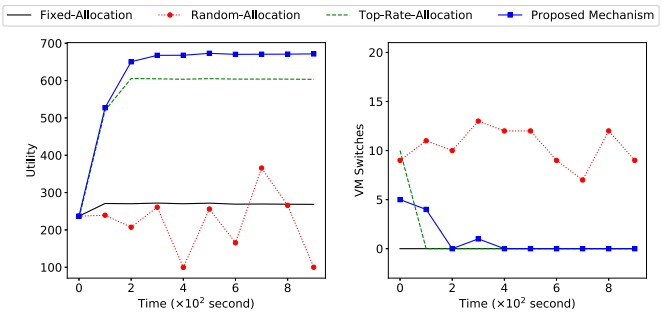
(a) $\beta = 0.6, \lambda = 10000$ (b) $\beta = 0.6, s = 0.8$ (c) $s = 0.8, \lambda = 10000$

Fig. 8. Performance comparison to optimal solution ($N = 20$, $M = 5$).

it gets larger. This is in accordance with the fact that β is the weight denoting the relative importance of optimizing delays (see Eq. (5)). It is also interesting to see from Fig. 4a that the delay reductions are negative when $\beta \leq 0.6$, even when the total utilities are positive (see Fig. 3c). After a careful examination we find that there are many tasks that will greatly benefit from offloading for energy savings at the cost of large delay penalties. For example, given 3GHz computation resource at the MEC server, a task with 2000KB data size, 100bit/cycle computation intensity, 1.0GHz UE's CPU frequency, 0.5Mb/s bandwidth and 0.8W trans power would achieve nearly 100% energy savings, however, the task delay will be 600% when it is offloaded.



(a) $s = 0.8, \beta = 0.3$



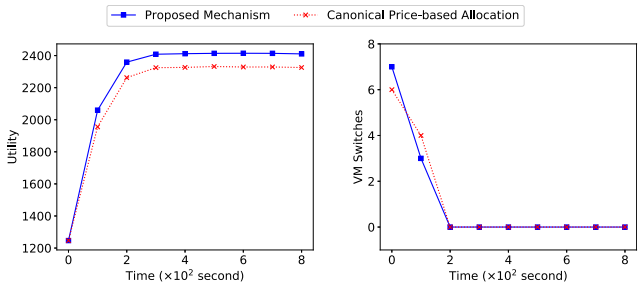
(b) $s = 1.2, \beta = 0.6$

Fig. 9. Performance comparison of online *centralized* algorithms under stationary simulated workload.

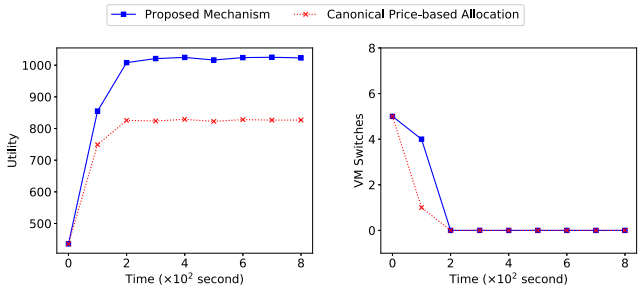
Figs. 5 and 6 give the performance of our *decentralized* algorithm and the Canonical Price-based Allocation algorithm. Again we can see that our algorithm outperforms and that as much as 45% improvement can be achieved when $s = 1.0$, $\beta = 0.5$ and λ = 10000/sec. Similar trends can be identified with regard to how the utility changes when each parameter varies, except that the utility gain achieved by our algorithm now increases as the workload becomes more skewed (Fig. 5a versus Fig. 3a).

Figs. 7 and 8 show the performance of our heuristic algorithm as compared with the optimal solution, where the optimal solution is derived by an exhaustive method that evaluates all feasible solutions to problem (6), that is, all combinations of the values of the binary variables $\{x_{ij}\}$ and $\{y_i\}$ ($x_{ij} \leq y_i$). The optimal solution is the one with the highest objective function value among all these possible combinations. Due to NP-hardness of the problem, we only evaluate the optimal solution for system with small sizes, i.e., $N \leq 20$ and $M \leq 5$. From the figures we can see that our resource-efficiency based heuristic is able to achieve 86%∼98% of the optimal performance with the average 90%, when $N = 10$ and $M = 3$; and it achieves 73%∼96% with the average 84%, when $N = 20, M = 5$.

We then evaluate the performance of our cost-aware online algorithms. To this end, we set the time needed to load and start a service at the MEC server T_i^{Start} = 20 sec.



(a) $s = 0.8, \beta = 0.3$



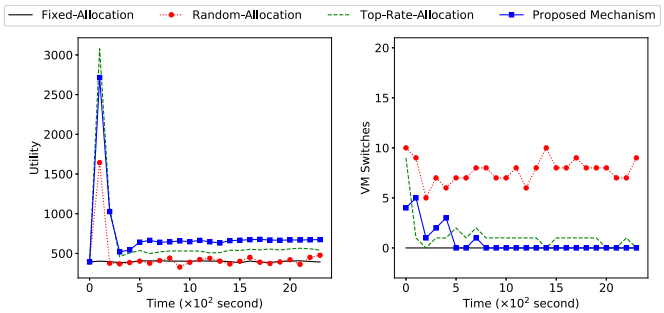
(b) $s = 1.2, \beta = 0.6$

Fig. 10. Performance comparison of online *decentralized* algorithms under stationary simulated workload.

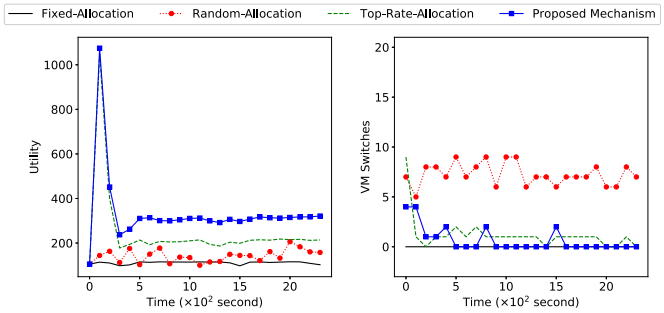
The length of each time slot ΔT is set as ΔT =100 sec, and the simulation time is 1000 sec. We investigate both utilities by different algorithms as well as the number of VM switches incurred in each time slot.

Fig. 9 shows performance of the four *centralized* algorithms as time elapses, where we randomly select 15 services at the very beginning of simulation, i.e., the first time slot, and allocate CPU resource of the MEC server equally among them. In each time slot that follows, the Fixed-Allocation algorithm keeps hosting these services, whereas the Random-Allocation picks services randomly. The Top-Rate-Allocation selects services based on the observed task arrival rates in each slot. Note that again resource allocation to the selected services in each time slot is determined optimally by our resource-efficiency based heuristic. We make the following observations from Fig. 9: 1) both our online centralized algorithm and the Top-Rate-Allocation outperform the other two in utilities, and our algorithm has the highest utility among them; 2) the Random-Allocation algorithm fluctuates and has the worst performance in both achieved utilities and VM switches, whereas the Fixed-Allocation behaves stably, i.e., no VM switch is incurred; and 3) both our cost-aware online algorithm and the Top-Rate-Allocation converge fast that it takes $2 \sim 4$ time slots for them to stabilize. These results indicate that our online algorithm can effectively optimize system performance while at the same time avoids system instability.

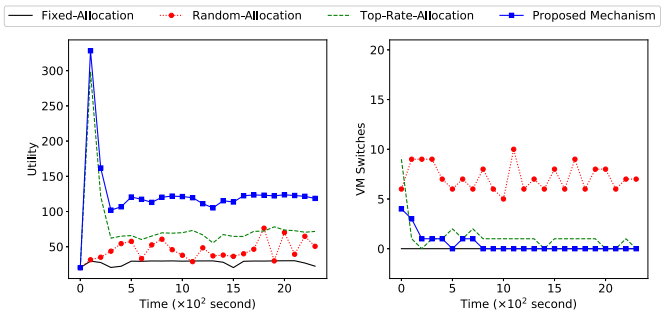
Fig. 10 shows the performance of our online *decentralized* algorithm and the Canonical Price-based Allocation algorithm. From the figure, we can see that although our decentralized algorithm achieves less utility than its centralized counterpart, it still outperforms the Canonical Price-based Allocation algorithm. For example, we observe more than 20% improvement when $s = 1.2$, $\beta = 0.6$. Meanwhile, both algorithms are cost-efficient that they keep hosting the same services when the process stabilizes.



(a) $\beta = 0.3$



(b) $\beta = 0.6$



(c) $\beta = 1.0$

Fig. 11. Performance comparison of online *centralized* algorithms in trace-driven simulation.

6.2.2 Performance Over Real World Traces

We also conduct a trace-based simulation to investigate the performance of our online algorithms (both *centralized* and *decentralized*) over real world traffic. The trace dataset we used is from [42], which contains packet inter-arrival times generated by 5 applications from 36 wireless devices. To have enough services in simulation, we regard each device as a single service and each application from a device as a sub-type of tasks. In this way, we have 36 services and 264 sub-types of tasks. Meanwhile, to simulate the task arrival process, we stretch the time axis in each trace by 60 (so that a second in the time axis now becomes a minute). The number of services that can be accommodated at the MEC server is set as 10, and all other parameters remain unchanged.

Fig. 11 shows the performance of the four *centralized* algorithms in trace-driven simulations. From the figure, we can see that: 1) our online algorithm outperforms the other three in the achieved utilities, under different weight parameters; 2) both our algorithm and the Top-Rate-Allocation converge fast under real world workloads, i.e., the convergence time is less than 5 time slots; and 3) it is interesting that our online algorithm even incurs less VM switches than the

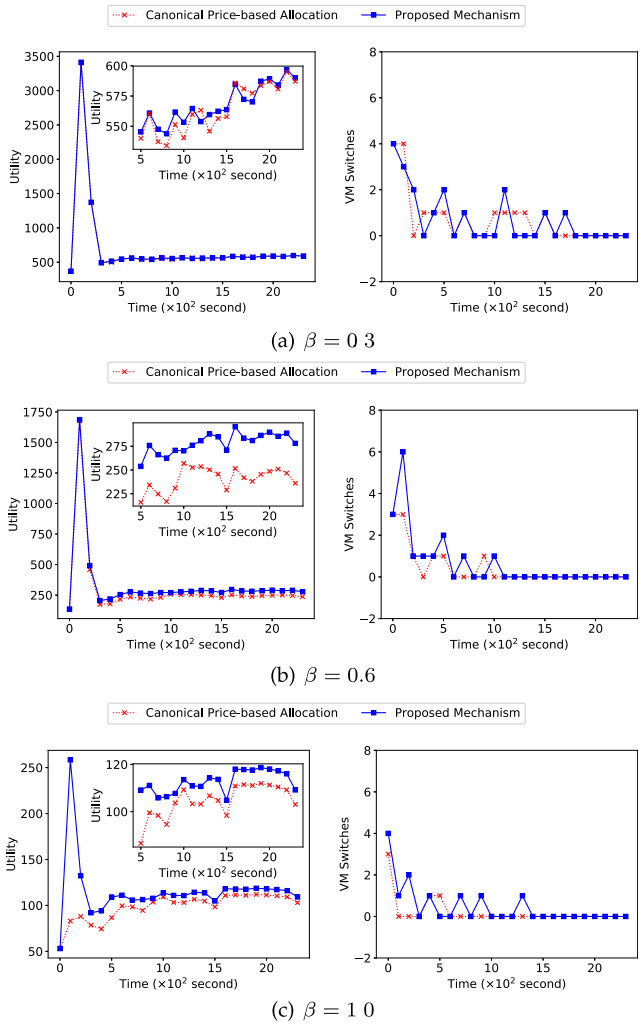


Fig. 12. Performance comparison of online *decentralized* algorithms in trace-driven simulation.

Top-Rate-Allocation algorithm, i.e., in most time slots we find that no VM switches is incurred by our algorithm whereas that is $2 \backsim 3$ for the Top-Rate-Allocation. This result indicates that under real-world workloads our online algorithm is more stable than the Top-Rate-Allocation.

The performance of our online *decentralized* algorithm and the Canonical Price-based Allocation algorithm are shown in Fig. 12. It is clear that our algorithm has a comparable performance with the Canonical Price-based Allocation algorithm when β is small, i.e., $\beta \leq 0.3$, and it outperforms when β gets larger. In particular, we find that in average 20% utility improvement can be achieved by our algorithm when $\beta = 0.6$ and $\beta = 1.0$. Moreover, both algorithms are stable as they almost incur no VM switches after the process converges.

It is also interesting to observe that there is always a peak in the utility before reaching convergence for both our centralized and decentralized online algorithms. A deep investigation reveals that the peak is due to the construction of dataset in simulation. More specifically, since in each trace only the packet inter-arrival times are recorded, the first time slot in the combined dataset thus contains all the flows (i.e., their first packet), which then results in a peak in task arrival rate and also the utilities (recall that utility is a linear function in task arrival rate).

6.3 Summary of Observations

We end this section by summarizing what we have found in numerical studies and simulations. First, as compared to baseline algorithms, both our centralized and decentralized resource-efficiency based algorithms improve system utility under a wide range of parameter settings. Second, through synthetic workload simulations, we observe that our cost-aware online algorithms converge fast and that they can achieve high utility while at the same time incur low cost of VM switches. Trace-driven simulations demonstrate that the online centralized algorithm is even more stable than the Top-Rate-Allocation algorithm, and the online decentralized algorithm is able to get a higher performance than the Canonical Price-based Allocation algorithm. Finally, task latencies and energy consumptions can be well balanced by our mechanism through adjusting the weight parameter.

7 Discussions and Related Work

In this section, we explore the implications of our mechanism and present some future research directions. We end with a brief discussion of the related work.

(1) Determining the Necessary Number of Edge Servers. We have assumed that the number of edge servers for each service is fixed and is a priori known, i.e., at most one VM can be hosted by the MEC server for each service, whereas in practice the number of edge servers are not known and should be part of the optimization problem. Here we show how our framework can be extended to tackle this problem, by allowing multiple VMs to be potentially hosted at the MEC server for each service.

Let n_i denote the number of VMs for service i. Assuming that tasks of type ij will be distributed evenly among these VMs due to loading balancing, i.e., the rate of tasks directed to each VM is $\frac{\lambda_{ij}}{n_i}$, and that CPU resources allocated to these VMs are identical, we can then formulate the following MINLP problem

$$\text{Max:} \quad \sum_{i=1}^{N} \sum_{j=1}^{N_i} \lambda_{ij} \times G_{ij}(F_i) \times x_{ij} \tag{22a}$$

$$\text{s.t.:} \quad \sum_{i=1}^{N} n_i F_i \leq F, \tag{22b}$$

$$\sum_{i=1}^{N} n_i y_i \leq M, \tag{22c}$$

$$\sum_{j=1}^{N_i} \frac{\lambda_{ij}}{n_i} \frac{L_{ij} C_{ij}}{F_i} < 1, \forall i, \tag{22d}$$

$$x_{ij} \leq y_i, \forall i, j \tag{22e}$$

$$0 \leq F_i \leq F^{\text{MAX}}, \forall i \tag{22f}$$

$$y_i \in \{0,1\}, x_{ij} \in \{0,1\}, n_i \in \mathbb{N}, \forall i, j \tag{22g}$$

where n_i becomes an extra variable to be optimized. To characterize users' Quality of Service, i.e., delays of performing tasks at the MEC server, each VM i can be modeled as a single-server queueing system shared by N_i customer classes

under the First-In-First-Out discipline. The latency of performing task A_{ij} at the MEC server comprises of three parts: 1) upload/transmission delay, which is $\frac{L_{ij}}{w_{ij}}$; 2) execution delay by the VM, which is $\frac{L_{ij}C_{ij}}{F_i}$; and 3) queueing delay, which is the time spent by the request sitting in a queue waiting to be processed. Note that in order for the system to be stable, workload at each VM should not exceed its computing capacity, as constraint (22d) states. The queue length distributions and queueing delay for a single-server system with multi-class customers under the FIFO discipline has been well studied, and simple formula has been devised to characterize the joint queue length distributions under the Poisson customer arrivals [9]. For example, let W_i be the steady-state waiting time for customers (given that the arrival processes are Poissonian, all customer classes have the same waiting time), and $L = \sum_{j=1}^{N_i} \lambda_{ij}(1 - z_j), |z_j| \leq 1, \forall j$, then we have [38]

$$E[e^{-LW_i}] = (1-\rho)\frac{L}{\sum_{j=1}^{N_i}[\lambda_{ij}(\beta_j\{L\} - z_j)]}. \tag{23}$$

where $\rho < 1$ is the load at the VM, $\beta_j\{\cdot\}$ is the Laplace-Stieljes transformation. The energy consumption of offloading tasks to the MEC server, and the delay and energy consumption of local computation at UEs remain unchanged.

(2) Incorporating Communication Models. Communication models (and power management) determine data rate of UEs and thus can impact delays and energy consumption of performing tasks. We show here through an example how our framework allows us to flexibly incorporate various communication models. Let L be the number of user equipments (UEs) in system. We use a tuple $A_{ij}^s(L_{ij}, C_{ij}, f_s, p_s, K_s, w_s)$ to characterize a task of type ij from UE s, where f_s, K_s, p_s and w_s represent the local CPU frequency, energy coefficient, transmit power and data rate of UE s, respectively. Delays and energy consumption of performing task A_{ij}^s when it is executed locally and when it is offloaded to the MEC server can be calculated as follows:

$$T_{ij,s}^{\text{Local}} = \frac{L_{ij} \times C_{ij}}{f_s}, \tag{24}$$

$$E_{ij,s}^{\text{Local}} = K_s \times (L_{ij}C_{ij}) \times f_s^2, \tag{25}$$

$$T_{ij,s}^{\text{MEC}} = \frac{L_{ij}}{w_s} + \frac{L_{ij} \times C_{ij}}{F_i}, \tag{26}$$

$$E_{ij,s}^{\text{MEC}} = p_s \times \frac{L_{ij}}{w_s}. \tag{27}$$

Denote W (Hz) as the available bandwidth of the system, which is shared by all UEs. Moreover, we assume wireless channels between UEs and the BS (MEC server) are i.i.d. frequency-flat block fading. The channel power gain H_s from UE s to the MEC server thus can be given as $H_s = h_s g_0 (\frac{d_0}{d_s})^\theta$, where h_s is the small-scale fading channel power gain from UE s to the MEC server, g_0 is the path-loss constant, θ is the path-loss exponent, d_0 is the reference distance and d_s is the distance from UE s to the MEC server. Let $\alpha_s \in [0, 1]$ denote the portion of bandwidth allocated to UE s, and N_0 be the noise power spectral density at the receiver of the MEC server. Data rate w_s of UE s then can be characterized as

$$w_s = \begin{cases} \alpha_s W \log_2(1 + \frac{H_s p_s}{\alpha_s N_0 W}), & \alpha_s > 0 \\ 0, & \alpha_s = 0 \end{cases} \tag{28}$$

Let $\lambda_{ij,s}$ be the arrival rate of task A_{ij}^s, and G_{ij}^s be the gain of offloading A_{ij}^s to the MEC server. Also denote by $x_{ij,s}$ a binary variable indicating whether or not A_{ij}^s is offloaded to the MEC server. Then we can formulate the following joint communication, computation and networking optimization problem

$$\text{Max:} \quad \sum_{s=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N_i} \lambda_{ij,s} \times G_{ij,s}(F_i, \alpha_s, p_s) \times x_{ij,s} \tag{29a}$$

$$\text{s.t.:} \quad \sum_{i=1}^{N} F_i \leq F, \tag{29b}$$

$$\sum_{i=1}^{N} y_i \leq M, \tag{29c}$$

$$\sum_{s=1}^{L} \alpha_s \leq 1, \tag{29d}$$

$$x_{ij,s} \leq y_i, \forall i, j, s \tag{29e}$$

$$0 \leq F_i \leq F^{\text{MAX}}, \forall i \tag{29f}$$

$$0 \leq p_s \leq p_s^{\text{MAX}}, \forall s \tag{29g}$$

$$y_i \in \{0,1\}, x_{ij,s} \in \{0,1\}, \alpha_s \in [0,1], \forall i, j, s \tag{29h}$$

where α_s and p_s become two new variables to be optimized. The above problem is, of course, NP-hard. To solve it effectively, our idea is again to leverage the concept of *resource efficiency*. More specifically, since $\max_{\{F_i,\alpha_s,p_s\}} \{\sum_s \sum_i \sum_j \lambda_{ij,s} \times G_{ij,s}(F_i, \alpha_s, p_s) \times x_{ij,s}\} = \max_{\{F_i\}}\{\max_{\{\alpha_s,p_s\}} \sum_s \sum_i \sum_j \lambda_{ij,s} G_{ij,s}(F_i, \alpha_s, p_s) x_{ij,s}\}$, problem (29) can be decomposed into two sub-problems: the first (upper-level) sub-problem is to adjust CPU resource allocations to services so that system performance can be maximized, which can be formulated as follows:

$$\text{Max}_{\{F_i\}}: \quad \sum_{s=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N_i} \lambda_{ij,s} \times G_{ij,s}(F_i) \times x_{ij,s} \tag{30a}$$

$$\text{s.t.:} \quad \sum_{i=1}^{N} F_i \leq F, \tag{30b}$$

$$\sum_{i=1}^{N} y_i \leq M, \tag{30c}$$

$$x_{ij,s} \leq y_i, \forall i, j, s \tag{30d}$$

$$0 \leq F_i \leq F^{\text{MAX}}, \forall i \tag{30e}$$

$$y_i \in \{0,1\}, x_{ij,s} \in \{0,1\}, \forall i, j, s \tag{30f}$$

Note that problem (30) is essentially the same as the one that we addressed in the current work, and therefore the same resource efficiency and utility based heuristic algorithm can be applied. The second (lower-level) sub-problem is to determine the optimal wireless bandwidth allocations to UEs and the optimally power allocation of UEs, given a fixed CPU resource allocations $\boldsymbol{F} = (F_1, F_2, \ldots, F_N)$ by the upper level. Mathematically, this sub-problem can be formulated as

$$\text{Max}_{\{\alpha_s, p_s\}}: \quad \sum_{s=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N_i} \lambda_{ij,s} \times G_{ij,s}(\alpha_s, p_s) \times x_{ij,s} \tag{31a}$$

$$\text{s.t.:} \quad \sum_{s=1}^{L} \alpha_s \leq 1, \tag{31b}$$

$$x_{ij,s} \leq y_i, y_i = 1 \text{ if } F_i > 0, \forall i, j, s \tag{31c}$$

$$0 \leq p_s \leq p_s^{\text{MAX}}, \forall s \tag{31d}$$

$$\alpha_s \in [0, 1], \forall s \tag{31e}$$

To solve problem (31), let $\mathcal{M} = \{i | F_i > 0\}$, we consider the following variant

$$\text{Max}_{\{\alpha_s, p_s\}}: \quad \sum_{s=1}^{L}\sum_{i\in\mathcal{M}}\sum_{j=1}^{N_i} \lambda_{ij,s} \times G_{ij,s}(\alpha_s, p_s) \tag{32a}$$

$$\text{s.t.:} \quad \sum_{s=1}^{L} \alpha_s \leq 1, \tag{32b}$$

$$\epsilon \leq p_s \leq p_s^{\text{MAX}}, \forall s \tag{32c}$$

$$\alpha_s \in [\epsilon, 1], \forall s \tag{32d}$$

which is obtained by relaxing α_s and p_s to be strictly larger than or equal to a small positive constant $0 < \epsilon < \frac{1}{L}$ (so that the objective function is differentiable), and by assuming all tasks related to the hosted services will be offloaded to the MEC server. The optimal value of problem (32) is smaller than problem (31) but can be expected close to that of (31) by setting ϵ sufficiently small. Problem (32) can be tackled with the folowing Gauss-Seidel method [27]:

1) Obtaining the optimal transmit power. For a fixed bandwidth allocation $\{\alpha_s\}$, the optimal transmit power p_s at UE s is achieved at either the stationary point of the objective function or one of the boundary points, which can be given as follows:

$$p_s = \underset{p_s \in \{v \in \mathcal{H} | \epsilon \leq v \leq p_s^{\text{MAX}}\}}{\text{argmax:}} \sum_{i\in\mathcal{M}}\sum_{j=1}^{N_i} \lambda_{ij,s} \times G_{ij,s}(\alpha_s, p_s) \tag{33}$$

$$\mathcal{H} = \{\epsilon, p_s^{\text{MAX}}\} \cup \{p_s | \sum_{i\in\mathcal{M}}\sum_{j=1}^{N_i} \lambda_{ij,s} \frac{dG_{ij,s}(\alpha_s, p_s)}{dp_s} = 0\} \tag{34}$$

2) Obtaining the optimal wireless bandwidth allocation. For a fixed transmit power allocation $\{p_s\}$, the optimal wireless bandwidth allocation $\{\alpha_s\}$ can be obtained by solving the following problem

$$\text{Max}_{\{\alpha_s\}}: \quad \sum_{s=1}^{L}\sum_{i\in\mathcal{M}}\sum_{j=1}^{N_i} \lambda_{ij,s} \times G_{ij,s}(\alpha_s, p_s) \tag{35a}$$

$$\text{s.t.:} \quad \sum_{s=1}^{L} \alpha_s \leq 1, \tag{35b}$$

$$\alpha_s \in [\epsilon, 1], \forall s \tag{35c}$$

The structure of the above problem naturally suggests a Lagrangian-based method to solve it efficiently. More specifically, let $\gamma > 0$ be the Lagrangian multiplier associated with the constraint $\sum_{s=1}^{L} \alpha_s \leq 1$, the partial Lagrangian function can be written as follows:

$$\mathcal{L}(\{\alpha_s\}, \gamma) = -\sum_{s=1}^{L}\sum_{i\in\mathcal{M}}\sum_{j=1}^{N_i} \lambda_{ij,s} G_{ij,s}(\alpha_s, p_s) + \gamma(\sum_{s=1}^{L} \alpha_s - 1) \tag{36}$$

Based on KKT (Karush-Kuhn-Tucker) conditions, the optimal wireless bandwidth allocation α_s^* and the optimal Lagrangian multiplier γ^* should satisfy

$$\alpha_s^* = \begin{cases} \max\{\epsilon, \psi_s(\gamma^*)\}, \forall s, \gamma^* > 0 \\ \sum_{s=1}^{L} \alpha_s^* \leq 1 \end{cases} \tag{37}$$

where $\psi_s(\gamma^*)$ denotes the root of $-\sum_{i\in\mathcal{M}}\sum_{j=1}^{N_i} \lambda_{ij,s} \frac{dG_{ij,s}(\alpha_s, p_s)}{d\alpha_s} = \gamma$, which is positive and unique as $-\sum_{i\in\mathcal{M}}\sum_{j=1}^{N_i} \lambda_{ij,s} \frac{dG_{ij,s}(\alpha_s, p_s)}{d\alpha_s} > 0$ decreases with α_s. This implies a bisection search over $[\gamma_{\text{L}}, \gamma_{\text{U}}]$ for the optimal γ^*, where $\gamma_{\text{L}} = \max_{1\leq s\leq L}\{-\sum_{i\in\mathcal{M}}\sum_{j=1}^{N_i} \lambda_{ij,s} \frac{dG_{ij,s}(\alpha_s, p_s)}{d\alpha_s}|_{\alpha_s=1}\}$ and γ_{U} satisfies $\sum_s \max\{\epsilon, \psi_s(\gamma_{\text{U}})\} < 1$. The search process terminates whenever $|\sum_s \max\{\epsilon, \psi_s(\gamma^*)\} - 1| < \xi$, where $\xi > 0$ is the accuracy of the algorithm.

In short, the Gauss-Seidel method iteratively updates the transmit power and wireless bandwidth allocations until convergence. The objective function value of problem (32), given a fixed CPU resource allocation, is then used to evaluate the resource efficiency.

Related Work The issue of resource allocation, task offloading and service selection/placement in MEC has been investigated separately or jointly. Existing work on resource allocation and task offloading can be generally classified into four categories: i) Single-User Single-Server MEC systems [44], [51], ii) Multi-User Single-Server MEC systems [10], [20], [41], [34], iii) Multi-User Multi-Server MEC systems [50], [7], [47], [18], and iv) fog-cloud network [23], [1]. The authors in [44] proposed convex optimization techniques to minimize the transmission energy consumption for mobile devices. Authors in [51] introduced a deep reinforcement learning framework to minimize the weighted sum of execution delay and energy consumption. [10] proposed a new strategy based on a Q-Learning algorithm, and [20] proposed pricing game-based offloading decision and interference-avoid communication resource allocation for minimizing the energy-time cost on the

mobile terminal side. Authors in [41] formulated the problem as a new Integer Program, and [34] leveraged the Lyapunov optimization to maximize the overall task offloading rate.

To minimize the energy-time cost of all mobile users, authors in [50] proposed an efficient bisection search method, and [7] introduced a hybrid relaying (HR) approach. The work on [47] applied stack-based cache mechanism (SCM) to ensure the fairness of server resources allocation, and [18] proposed a bilevel optimization approach to minimize the total energy consumption on the mobile terminal side under the delay constraint. [23] applied ADMM to jointly optimize computation offloading and wireless resource allocation, with the aim to reduce offloading transmission latency and release the constraint of limited radio resource. A deep recurrent Q-network (DRQN) approach in [1] is proposed to guarantee users' QoS requirements.

For service placement/selection problem, researchers have proposed Tabu-Search (TS) meta-heuristic [5], Low-MEP [19] and the contextual Multi-armed Bandit (MAB) [30] approaches to minimize users' perceived-latency. Collaborative Service Placement (CSP) [6] and Matching Theory based approach [53] also have been used to maximize the system performance and minimize the traffic load. To maximize the number of requests served, [15] proposed an iteration-based algorithm, and [33] proposed to use a randomized rounding technique.

Distributed mechanisms for MEC offloading and resource management have been extensively studied in literature, where game theory has been widely adopted as a mathematical tool for the design and analysis of the algorithms. For example, Liu *et al.* in [22] formulated a Stackelberg game for the interaction between mobile users and the MEC server, and proposed a price-based distributed offloading algorithm. Zhao *et al.* in [54] studied the collaborative task offloading in a vehicular network, where both the MEC server and cloud can be utilized. They proposed a distributed resource allocation and task offloading algorithm through formulating a potential game for the offloading decision making sub-problem and achieving optimal resource allocation using Lagrange multiplier method. Authors in [35] proposed a game-theoretical joint optimal pricing and resource allocation algorithm for mobile edge computing in NOMA-based networks, taking into account the payment by MUs for communication and computation resources. Wang *et al.* in [43] extended mobile edge computing in wireless cellular networks with content caching, and investigated the joint resource allocation, task offloading and content caching problem. They proposed a distributed solution based on the ADMM method. Authors in [37] studied computation offloading in a distributed MEC network, and proposed a stochastic gradient descent algorithm to jointly optimize the offloading probability and transmission power of the SDs.

Most recent efforts are dedicated to optimization of MEC systems in more realistic settings. For example, Tang *et al.* [39] considered edge load dynamics and proposed a model-free deep reinforcement learning-based distributed task offloading algorithm for non-divisible delay-sensitive tasks. Li *et al.* [21] focused on task offloading with statistical QoS guarantees, and proposed an algorithm to provide statistical QoS guarantee for tasks using convex optimization theory and Gibbs sampling method. Wang *et al.* [45] studied the problem of minimizing the maximal computation and transmission delay among all users, and developed a multi-stack reinforcement learning algorithm to solve the joint task, spectrum, and transmit power allocation problem. Authors in [12] studied the joint service placement and request scheduling problem in edge clouds with the aim at maximizing the total amount of requests served by multiple edge clouds. Wen *et al.* [46] considered the issue of software caching and multicasting in a MEC system, and proposed two sub-optimal algorithms to solve the problem of joint software caching, computation offloading and communications resource allocation.

Our work differs from existing work in that: 1) we exploit heterogeneity of tasks from the same service based on their characteristics, which allows us to develop individual and fine-grained task offloading policies for them; 2) both the computation and storage resource constraint at the MEC server are considered in our system model; and 3) we explicitly incorporate the cost of service switches from practical MEC systems in the design of online service caching and task offloading algorithms.

8 Conclusion

We study the problem of maximizing user's QoE in a resource-limited MEC system through jointly optimizing service selection, resource allocation and task offloading decision, which are tightly coupled in the control of MEC systems. We formulate the task as a mixed-integer nonlinear programming problem and prove its NP-hardness. To solve it efficiently, we first reformulate it as a network utility maximization problem (NUM) and then propose a resource-efficiency based heuristic. We further develop distributed and online algorithms that adapt to system changes, with the cost of service switches being explicitly incorporated into the algorithm design. Last, we evaluate our mechanisms through extensive simulations and results illustrated their efficiency.

References

[1] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1041–1056, Jan. 2021.

[2] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Joint allocation of computation and communication resources in multiuser mobile cloud computing," in *Proc. IEEE 14th Workshop Signal Process. Adv. Wireless Commun.*, 2013, pp. 26–30.

[3] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*, Cambridge, U.K. : Cambridge Univ. Press, 2004.

[4] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley, "Notes on decomposition methods," *Notes EE364B, Stanford Univ.*, vol. 635, pp. 1–36, 2007.

[5] B. Brik, P. A. Frangoudis, and A. Ksentini, "Service-oriented MEC applications placement in a federated edge cloud architecture," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.

[6] 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.

[7] X. Chen, Y. Cai, Q. Shi, M. Zhao, B. Champagne, and L. Hanzo, "Efficient resource allocation for relay-assisted computation offloading in mobile-edge computing," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2452–2468, Mar. 2020.

[8] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[9] J. W. Cohen, *The Single Server Queue*, Amsterdam, The Netherlands: Elsevier, 2012.

[10] B. Dab, N. Aitsaadi, and R. Langar, "Q-learning algorithm for joint computation offloading and resource allocation in edge cloud," in *Proc. IFIP/IEEE Symp. Integr. Netw. Serv. Manage.*, 2019, pp. 45–52.

[11] I. R. De Farias and G. L. Nemhauser, "A polyhedral study of the cardinality constrained knapsack problem," in *Proc. Int. Conf. Integer Program. Combinatorial Optim.*, 2002, pp. 291–303.

[12] V. Farhadi *et al.*, "Service placement and request scheduling for data-intensive applications in edge clouds," *Proc. IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 779–792, Apr. 2021.

[13] G. M. D. T. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, 2019, Art. no. 2022.

[14] R. W. Freund and F. Jarre, "Solving the sum-of-ratios problem by an interior-point method," *J. Glob. Optim.*, vol. 19, no. 1, pp. 83–102, 2001.

[15] 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.

[16] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction-based VM allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 1702–1716, Nov./Dec. 2021.

[17] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Serv.*, 2012, pp. 225–238.

[18] P.-Q. Huang, Y. Wang, K. Wang, and Z.-Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4228–4241, Oct. 2020.

[19] S. Lee, S. Lee, and M.-K. Shin, "Low cost MEC server placement and association in 5G networks," in *Proc. Int. Conf. Informat. Commun. Technol. Convergence*, 2019, pp. 879–882.

[20] L. Li, L. Gu, J. Hong, and S. Jiang, "Joint computation offloading and wireless resource allocation in mobile edge computing," in *Proc. IEEE 4th Int. Conf. Comput. Commun.*, 2018, pp. 705–711.

[21] Q. Li, S. Wang, A. Zhou, X. Ma, F. yang, and A. X. Liu, "QoS driven task offloading with statistical guarantee in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 278–290, Jan. 2022.

[22] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 420–423, Jun. 2018.

[23] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Trans. Veh. Technol*, vol. 67, no. 12, pp. 12137–12151, Dec. 2018.

[24] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[25] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[26] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.

[27] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Glob. Commun. Conf.*, 2016, pp. 1–6.

[28] I. Necoara and V. Nedelcu, "Rate analysis of inexact dual first-order methods application to dual decomposition," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1232–1243, May 2014.

[29] S. Nunna *et al.*, "Enabling real-time context-aware collaboration through 5G and mobile edge computing," in *Proc. 12th Int. Conf. Informat. Technol.-New Gener.*, 2015, pp. 601–605.

[30] 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.

[31] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[32] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Commun. Surv. Tut.*, vol. 16, no. 1, pp. 414–454, First Quarter 2014.

[33] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in MEC networks with storage, computation, and communication constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1047–1060, Jun. 2020.

[34] J. Ren, K. M. Mahfujul, F. Lyu, S. Yue, and Y. Zhang, "Joint channel allocation and resource management for stochastic computation offloading in MEC," *IEEE Trans. Veh. Technol*, vol. 69, no. 8, pp. 8900–8913, Aug. 2020.

[35] R. Roostaei, Z. Dabiri, and Z. Movahedi, "A game-theoretic joint optimal pricing and resource allocation for mobile edge computing in noma-based 5G networks and beyond," *Comput. Netw.*, vol. 198, 2021, Art. no. 108352.

[36] Y. Su, Z. Wang, M. Cao, M. Jia, and F. Liu, "Convergence analysis of dual decomposition algorithm in distributed optimization: Asynchrony and inexactness," 2021, *arXiv:2103.02784*.

[37] F. Sufyan and A. Banerjee, "Computation offloading for distributed mobile edge computing network: A multiobjective approach," *IEEE Access*, vol. 8, pp. 149 915–149 930, 2020.

[38] T. Takine, "Queue length distribution in a fifo single-server queue with multiple arrival streams having different service time distributions," *Queueing Syst.*, vol. 39, no. 4, pp. 349–375, 2001.

[39] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, early access, Nov. 10, 2020, doi: 10.1109/TMC.2020.3036871.

[40] 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.

[41] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[42] A. S. Uluagac, Crawdad data set gatech/fingerprinting, Jun. 09, 2014. [Online]. Available: https://crawdad.org/gatech/fingerprinting/20140609/realtestbed/index.html

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

[44] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2443–2459, Apr. 2020.

[45] S. Wang, M. Chen, X. Liu, C. Yin, S. Cui, and H. Vincent Poor, "A machine learning approach for task and resource allocation in mobile-edge computing-based networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1358–1372, Feb. 2021.

[46] W. Wen, Y. Cui, T. Q. S. Quek, F.-C. Zheng, and S. Jin, "Joint optimal software caching, computation offloading and communications resource allocation for mobile edge computing," *IEEE Trans. Veh. Technol*, vol. 69, no. 7, pp. 7879–7894, Jul. 2020.

[47] K. Xiao, Z. Gao, C. Yao, Q. Wang, Z. Mo, and Y. Yang, "Task offloading and resources allocation based on fairness in edge computing," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2019, pp. 1–6.

[48] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Computer Commun.*, 2018, pp. 207–215.

[49] H. Yaïche, R. R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 667–678, Oct. 2000.

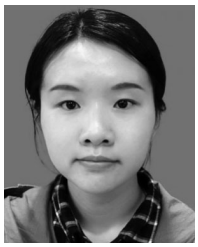
[50] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, Jan. 2020.

[51] J. Yan, S. Bi, and Y. J. A. Zhang, "Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5404–5419, Aug. 2020.

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

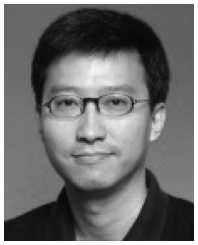
[53] N. Yu, Q. Xie, Q. Wang, H. Du, H. Huang, and X. Jia, "Collaborative service placement for mobile edge computing applications," in *Proc. IEEE Glob. Commun. Conf.*, 2018, pp. 1–6.
[54] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

Weibo Chu received the BS degree in software engineering and the PhD degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2005 and 2013, respectively. From 2011 to 2012, he was a visiting researcher with Microsoft Research Asia, Beijing. Since 2013, he has been with the School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an, China, where he is currently an associate professor. His research interests include internet measurement and modeling, traffic analysis, and performance evaluation. He has participated in various research and development projects on network testing, performance evaluation and troubleshooting, and gained extensive experiences in the development of networked systems for research and engineering purposes.



Peijie Yu received the BE degree from Hebei University, Baoding, China, in 2019. She is currently working toward the MS degree with the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. Her research interests include resource management and task scheduling in mobile edge computing.

Zhiwen Yu received the PhD degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2005. He is currently a professor and the dean of the School of Computer Science, Northwestern Polytechnical University. He was an alexander von humboldt fellow with Mannheim University, Germany, and a research fellow with Kyoto University, Kyoto, Japan. His research interests include ubiquitous computing and mobile computing.



John C.S. Lui (Fellow, IEEE) received the PhD degree in computer science from The University of California, Los Angeles. He is currently a professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His current research interests include communication networks, network/system security (such as cloud security and mobile security), network economics, network sciences (such as online social networks and information spreading), cloud computing, large-scale distributed systems, and performance evaluation theory. He serves in the editorial board of the *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *Journal of Performance Evaluation,* and *International Journal of Network Security*. From 2005 to 2011, he was the chairman of the CSE Department. He was the recipient of various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award. He was also the co-recipient of IFIP WG 7.3 Performance 2005 and IEEE/IFIP NOMS 2006 best student paper awards. He is an elected member of the IFIP WG 7.3, a fellow of the ACM and croucher senior research fellow.

Yi Lin received the PhD degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2005. He is currently an associate professor with the School of Computer Science, Northwestern Polytechnical University. His research interests include data storage and software engineering.