# Joint Service Caching, Resource Allocation and Task Offloading for MEC-Based Networks: A Multi-Layer Optimization Approach

Weibo Chu, Xinming Jia, Zhiwen Yu, *Senior Member, IEEE*, John C.S. Lui, *Fellow, IEEE*, and Yi Lin

***Abstract*—To provide reliable and elastic Multi-access edge computing services, one feasible solution is to federate geographically proximate edge servers to form a logically centralized resource pool. Optimization of such systems, however, becomes challenging. In this paper, we study the problem of maximizing users' QoE in a MEC-based network, through jointly optimizing service caching, resource allocation and task offloading decisions. We formulate a mixed-integer nonlinear programming (MINLP) problem for the task and establish its NP-hardness. To tackle it efficiently, we propose a novel two-stage algorithmic solution based on approximation and decomposition theory. The proposed algorithm achieves high system performance while at the same time, ensures all constraints from different layers are satisfied. Meanwhile, the structure of the algorithm also fits the multi-layer optimizing feature, making it suitable to be implemented at different layers. In addition, we propose a distributed and online version of our mechanism with very limited information exchange between MEC servers, and further demonstrate how the cost of service switches from real MEC systems can be incorporated into our framework. We evaluate our mechanisms through simulations with both synthetic and real-world traces, and results indicate they are effective as compared to representative baseline algorithms.**

***Index Terms*—Distributed algorithm, MEC-based network, resource allocation, service caching, task offloading.**

## I. Introduction

### A. Background and Motivations

With the ever increasing popularity of mobile devices and the proliferation of IoT in recent years, we have witnessed an explosive growth of new applications such as augmented reality, interactive gaming and autonomous driving. These applications are typically resource-hungry and delay-sensitive. For example, VR systems often require latency less than 10 ms in order to make the VR world realistic [1], and the latency should be no more than 20 ms to ensure safety for autonomous cars [21]. Local execution of these applications is in general not feasible since resources of mobile devices (e.g., battery, storage) are limited. Relying on cloud to process and store data is also problematic due to the excessively long delay and unstable network connection.

To alleviate the tension between emerging applications and the resource-constrained devices, a new network computing paradigm called Multi-access Edge Computing [43] [34] (MEC), has been proposed. MEC allows users to execute their applications at the network edge by deploying computing and storage resources in close proximity to end users, and users offloading their tasks to the MEC servers. For example, in the VR application, one can offload the three most resource-intensive components, namely, the tracker, mapper and object recognizer [53] to the edge server, so as to enjoy fast response and energy reductions. This also brings additional benefits such as cost efficiency, context-awareness [45] [40], privacy/security enhancement [6], etc. As a result, MEC is widely recognized as a promising solution to enable emerging applications.

Whereas MEC is able to offer network computing services with low latency, providing reliable and elastic MEC services for the public remains a challenge. This is particularly true for the fact that MEC servers are often resource-constrained as compared with the cloud infrastructure (i.e., data centers), and that they are sporadically distributed and deployed. There is a trend that edge servers within the same geographic region group together to form a shared resource pool (i.e., the open edge computing environment [3] [4]), so as to improve the overall system performance. In fact, researchers have shown that by serving requests from nearby servers/BSs, users' experience can be significantly enhanced [20] [26].

As federating geographically close edge resources opens a door to build cloud-computing-like MEC services, optimization of such systems poses significant challenges to both academic and industry community, for the following reasons. First, the problem is intrinsically complicated as it involves addressing three different sub-problems, namely, *resource allocation*, *service caching (a.k.a service placement)* and *task scheduling/offloading*, each of which is nontrivial to tackle. Note that these sub-problems are tightly coupled in that addressing one without considering the others would probably lead to sub-optimum. Second, as the system is comprised of different type of resources at different layers (e.g., *computing*, *communication*,

Manuscript received 27 February 2023; accepted 12 April 2023. Date of publication 17 April 2023; date of current version 6 March 2024. The work of John C.S. Lui was supported in part by GRF under Grant 14215722 and in part by RGC under Grant SRFS2122-4S02. This work was supported in part by the National Natural Science Foundation of China under Grant 62172333, and in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2021JM-073. Recommended for acceptance by J. Ren. *(Corresponding author: Weibo Chu.)*

Weibo Chu, Xinming Jia, Zhiwen Yu, and Yi Lin are with the Northwestern Polytechnical University, Xi'an, Shaanxi 710071, China (e-mail: wbchu@nwpu.edu.cn; jxm12f@mail.nwpu.edu.cn; zhiwenyu@nwpu.edu.cn; ly_cs@nwpu.edu.cn).

John C.S. Lui is with The Chinese University of Hong Kong, Central Ave, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TMC.2023.3268048

*storage* and *networking*), fully optimizing the system would unavoidably require an elegant usage of these heterogeneous resources, with a variety of constraints taken into account. Last, a real MEC system is time-evolving in that both the system (i.e., network condition) and user-generated workload fluctuate over time. To deal with the uncertainty and dynamics, a desired control mechanism should operate in an adaptive and distributed manner. In addition, it should also be of low cost so that the system can scale up.

### B. Contributions

In this paper, we study the problem of maximizing users' quality of experience (QoE) in a MEC-based network, through jointly optimizing service caching, resource allocation and task offloading decisions. We provide a *unified framework* to address the aforementioned challenges and propose a *multi-layer optimization* approach. More specifically, we formally define user's QoE as the weighted sum of task latency reduction and energy savings at UEs, and formulate an optimization problem for the task, with various constraints from different layers taken into account. The derived problem is proven NP-hard and we propose a novel solution by approximating it with another problem that is mathematically tractable (although still NP-hard). We then decompose the approximate problem into multiple sub-problems, where each of them can be efficiently solved by existing algorithms. Based on the centralized algorithmic solution and problem structure, we further devise a distributed and online mechanism that adapts to both workload and system variations, which at the same time requires very limited information exchange between nearby servers. Moreover, our framework allows us to explicitly incorporate the cost of service switches from an MEC system to trade-off the cost of reconfiguration and system performance. Simulation and numerical results illustrate that our proposed mechanism can improve users' QoE while at the same time make the system stable, as compared with representative baseline algorithms.

In summary, we make the following contributions:

1) We consider a heterogeneous *multi-user* and *multi-server* environment, and propose a fine-grained task offloading scheme where: 1) the tasks of each service are divided into multiple sub-types based on their characteristics such as data size, computation intensity, etc. This makes our model more realistic and also different from existing work that assumes the same type of tasks of each service/application. The fine-grained characterization of tasks enables us to further improve system performance through fully leveraging this heterogeneity; 2) a probabilistic task offloading strategy is designed for each user which allows it to offload arbitrarily part of its tasks to a nearby server for remote computing. This is unlike the widely adopted binary offloading policy in the current literature, i.e., each user may offload the whole application to one MEC server.

2) Instead of the delay requirements for each service or user, we introduce a new QoS constraint called *offloading rate* into the jointly optimizing task, which states that the proportion of tasks for each service processed by the MEC-based network should be no less than a preset threshold. This QoS requirement is extracted/formed from the service providers' point of view, i.e., based on the SLA (Service Level Agreement) between service providers and the MEC infrastructure provider, and it has not yet been considered in the previous work.

3) While there exists plenty of research on the optimization of MEC systems, limited work however has been performed so far in the multi-server environment [39] [8][58], regarding jointly optimizing resource allocation, service caching and task offloading, over the communication, networking, storage and computing layer. Moreover, although various optimization model and algorithmic solutions have been proposed, most of them are centralized and cannot be readily applied in a distributed and decentralized environment. In this paper, we provide a unified framework for the joint optimization task, taking into account constraints from all involved layers. We model the problem as a MINLP program, and propose an efficient two-stage algorithm based on approximation and decomposition theory. Our problem formulation allows us to develop a fully distributed and adaptive algorithm for each MEC server, through limited information exchange between neighboring nodes. Moreover, the derived mechanism is also secure as there is no need for each server to expose its local traffic information to neighboring nodes. We further consider influential factors from a real MEC system, in particular, service switches that can significantly degrade system performance, and develop a *cost-aware online* algorithm for optimal system control.

4) We evaluate our mechanism through both synthetic and trace-driven simulations. Results indicate that our mechanism outperforms the baselines, and the cost-aware online algorithm converges quickly and that it is capable of achieving high performance while at the same time maintain low system instability.

The remainder of this paper is organized as follows. In Section II we introduce system model and problem formulation. Section III describes our novel centralized algorithmic solution to the problem. Section IV elaborates decentralized mechanism and the cost-aware online algorithm. Section V presents numerical studies and simulation results. We give related work in Section VI and conclude the paper in Section VII.

## II. System Model and Problem Formulation

### A. System Model

Consider a Mobile-Edge Computing-based network as shown in Fig. 1, which consists of multiple base stations (BSs), MEC servers and user equipments (UEs). Each BS covers a specific area for serving its associated UEs, and is connected via backhaul links to other BSs. Moreover, each BS is equipped with one or more servers (hereafter called MEC server) so that computing tasks from UEs with stringent requirements (e.g., short delay, high energy efficiency) can be offloaded. To make our model generic, these BSs could be macro cells (eNBs), small cells (SCeNBs), or femto cells (HeNBs), and the MEC servers could be micro data centers, edge clouds or computing servers with high capacities, accordingly. Note that unlike traditional cloud data centers with abundant resources, MEC servers are often resource-limited so that they can only accommodate a limited

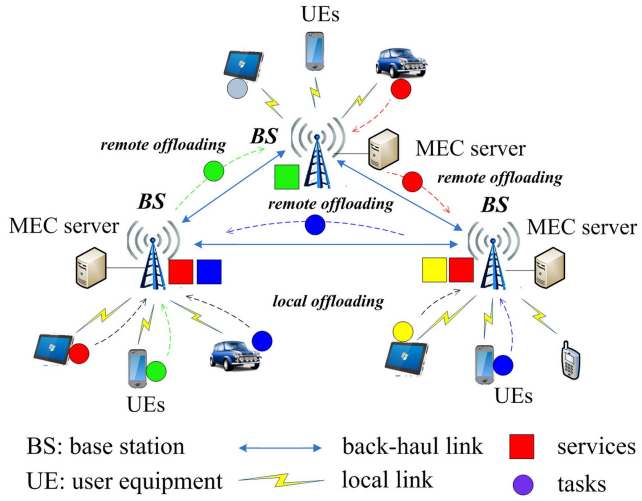



Fig. 1. A mobile-edge computing-based network.

number of services simultaneously, and process certain amount of tasks per time interval.

With multiple BSs/MEC servers,[1] a computing task from UE can be either performed at the UE where it originates, or be offloaded to the local MEC server if the required service is hosted and significant performance gains can be achieved. To fully exploit the power of the network and enhance users' QoE, we allow tasks from each BS to be potentially routed to its neighboring BSs through back-haul links. This occurs when, i.e., the required service is not locally present but it happens to be cached by a nearby BS. Throughout the paper, we assume that a computing task will not be directed to the remote cloud due to excessively long delay.

There are several fundamental problems with regard to how to utilize the resources of the network and how tasks from UEs are performed. Among them, *service placement* (a.k.a service selection or service caching) aims at selecting appropriate execution servers to instantiate service instances, and therefore it can significantly impact system performance as it determines whether an MEC server is able to process a specific type of tasks. Service placement is generally hard to plan due to its combinatorial nature, i.e., the problem is often modeled as a 0-1 programming problem. Moreover, for performance gains such as shorter latency and reduced energy consumption, only when the required service is allocated with adequate resources will a task be offloaded, and this raises the *service resource allocation* problem which manages the mapping of services to resources. There are two kinds of resources in a MEC-based network – *dedicated* and *amortized*. Resources such as CPU cycles and network bandwidth are often regarded as dedicated in that serving a task requires a dedicated share of the resource, and the total resource consumption is the sum of resource requirements of tasks/services scheduled on the server. On the other hand, storage resources for hosting services (code & data) is usually considered amortized in that one copy of data would support all tasks for the service.

In addition to service caching and service resource allocation, communication model is another important aspect that we should take into account. Communication-related parameters such as interference mode, channel gain, transmit power of UEs can affect the key performance metrics of task offloading. For example, data rate between UE and BS is a function of the bandwidth allocation and power management. This implies that optimization of communication (e.g., *bandwidth allocation* and *power management* at UEs) should also be considered as part of the overall task. Last but not least, the *request scheduling* problem – which deals with how to distribute tasks among servers within the capacity of the network, while at the same time meet certain performance requirements.

Obviously, all the problems described above are tightly coupled if we want to optimize the network performance, i.e., resource allocation should match the characteristics of user-generated tasks, and requests be scheduled according to server workload and network conditions. This naturally leads us to a *joint service placement, resource allocation, power management and task/request scheduling optimization* problem. The task is intrinsically challenging due to its cross-layer optimizing feature and the high heterogeneity and dynamics of both the system and workload.

### B. Problem Formulation

For simplicity of notations, we denote the set of BSs and services as $\mathcal{M}$ and $\mathcal{N}$, respectively. The number of BSs and services are $M$ and $N$. The set of UEs associated with each BS $k \in \mathcal{M}$ is denoted as $\mathcal{L}(k)$. Each task corresponds to a specific service and thus the number of different tasks/task types (e.g., image processing, navigation) equals to the number of services. We denote $\mathcal{R}$ as the set of different tasks, and sometimes use $\mathcal{R}$ and $\mathcal{N}$ interchangeably as $|\mathcal{R}| = |\mathcal{N}| = N$. To fully leverage the heterogeneity of workloads, we further divide each type of task $i \in \mathcal{R}$ into multiple sub-types $\mathcal{N}(i)$ based on the parameters of tasks (e.g., data size, computation intensity). For example, two users requesting the same face recognition service may introduce different computing demand due to different resolutions of their images submitted to the MEC server, and accordingly these two tasks are considered from different sub-types. This fine-grained classification of tasks also makes our model more practical. Table I gives main notations used in this work.

*1) Communication Model:* We assume that the available transmission bandwidth at each BS $k \in \mathcal{M}$, denoted by $B^k$ (Hz), is divided among different UEs $\mathcal{L}(k)$. Moreover, the type of fading on wireless channels between UEs and the BS is frequency-flat block fading. The channel power gain $H_u^k$ from UE $u \in \mathcal{L}(k)$ to BS $k$ can be represented by $H_u^k = h_u^k g_0^k \left(\frac{d_0^k}{d_u^k}\right)^{\theta^k}$, where $h_u^k$ is the small-scale fading channel power gain from UE $u$ to BS $k$, $g_0^k$ is the path-loss constant, $\theta^k$ is the path-loss exponent, $d_0^k$ is the reference distance and $d_u^k$ is the distance from UE $u$ to BS $k$. Let $\alpha_u^k \in [0, 1]$ denote the portion of bandwidth allocated to UE $u$, and $N_0^k$ be the noise power spectral density at the receiver of the BS. The data rate $w_u^k$ of UE $u$ at BS $k$ then

[1] We use BS and MEC server exchangably throughout this paper.

TABLE I
MAIN NOTATIONS

| Symbol | Definition |
|---|---|
| $\mathcal{M}$ | Set of BSs/servers in system |
| $\mathcal{N}$ | Set of services in system |
| $\mathcal{R}$ | Set of different tasks in system |
| $\mathcal{L}(k)$ | Set of UEs associated with BS $k$ |
| $F^k$ | CPU frequency at server $k$ |
| $F^{\text{MAX}}$ | Maximal CPU frequency that can be allocated to a service |
| $S^k$ | Storage capacity at server $k$ |
| $W^k$ | Communication capacity at server $k$ |
| $B^k$ | Bandwidth at BS $k$ |
| $i$ | Index of task types (services) |
| $j$ | Index of sub-types of tasks |
| $S_i$ | Size of service $i$ |
| $h_i^k$ | Offloading rate requirement for service $i$ at BS $k$ |
| $A_{ij}^{k,u}$ | A task of type $ij$ (sub-type $j$ in type $i$) from UE $u$ at BS $k$ |
| $\lambda_{ij}^{k,u}$ | Arrival rate of task $A_{ij}^{k,u}$ |
| $L_{ij}$ | Data size of a task of type $ij$ |
| $C_{ij}$ | Computation intensity of a task of type $ij$ |
| $f_u^k$ | CPU frequency of UE $u$ at BS $k$ |
| $p_u^k$ | Transmission power of UE $u$ at BS $k$ |
| $K_u^k$ | Energy coefficient of UE $u$ at BS $k$ |
| $w_u^k$ | Data rate between UE $u$ and BS $k$ |
| $\alpha_u^k$ | The portion of bandwidth allocated to UE $u$ at BS $k$ |
| $P_u^k$ | Transmission power of UE $u$ at BS $k$ |
| $T_{ij}^{k,u,\text{UE}}$ | Latency of task $A_{ij}^{k,u}$ if it is computed at UE |
| $T_{ij}^{k,u,s}$ | Latency of task $A_{ij}^{k,u}$ if it is offloaded to server $s$ |
| $E_{ij}^{k,u,\text{UE}}$ | Energy consumption of performing task $A_{ij}^{k,u}$ at UE |
| $E_{ij}^{k,u,s}$ | Energy consumption of offloading task $A_{ij}^{k,u}$ to server $s$ |
| $F_i^k$ | Computation resource allocated to service $i$ at server $k$ |
| $x_i^k$ | A binary var indicating if service $i$ is cached at server $k$ |
| $Pr_{ij}^{k,u,\text{UE}}$ | The probability that $A_{ij}^{k,u}$ is computed at UE |
| $Pr_{ij}^{k,u,s}$ | The probability that task $A_{ij}^{k,u}$ is computed at server $s$ |

can be characterized as follows [36]

$$w_u^k = \begin{cases} \alpha_u^k B^k \log_2\left(1+\frac{H_u^k P_u^k}{\alpha_u^k N_0^k B^k}\right), & \alpha_u^k > 0 \\ 0, & \alpha_u^k = 0 \end{cases}, \quad (1)$$

where $P_u^k$ is the transmit power of UE $u$ at BS $k$.

Let $P_u^{\text{MAX}} > 0$ be the maximum transmit power that $u$ can allocate. Obviously, a valid bandwidth allocation at BSs and power management of UEs in the network should satisfy

$$\sum_{u\in\mathcal{L}(k)} \alpha_u^k \le 1, \alpha_u^k \in [0,1], \quad \forall k \in \mathcal{M}, u \in \mathcal{L}(k) \quad (2)$$

$$0 \le P_u^k \le P_u^{\text{MAX}}, \quad \forall k \in \mathcal{M}, u \in \mathcal{L}(k) \quad (3)$$

*2) Computing & Networking Model:* Since tasks are categorized into multiple sub-types, we use a tuple $A_{ij}^{k,u}(L_{ij}, C_{ij}, f_u^k, p_u^k, K_u^k, w_u^k)$ to characterize a task $A_{ij}^{k,u}$ of type $ij$ (sub-type $j \in \mathcal{N}(i)$ of task $i$) from UE $u$ at BS $k$, where $L_{ij}$ is the data size of the task, $C_{ij}$ is the required computation intensity (number of CPU cycles per bit); $f_u^k$, $p_u^k$ and $K_u^k$ are CPU frequency, transmit power and energy coefficient of the UE $u$ where the task is generated, respectively. And $w_u^k$ is the data rate between $u$ and BS $k$.

Let $T_{ij}^{k,u,\text{UE}}$ and $E_{ij}^{k,u,\text{UE}}$ be the latency and energy consumption when task $A_{ij}^{k,u}$ is computed at UE $u$, respectively. We have

$$T_{ij}^{k,u,\text{UE}} = \frac{L_{ij}\times C_{ij}}{f_u^k}, \quad (4)$$

$$E_{ij}^{k,u,\text{UE}} = K_u^k \times (L_{ij}C_{ij}) \times {f_u^k}^2, \quad (5)$$

where $L_{ij}\times C_{ij}$ denotes the computation workload (in CPU cycles) of the task. (4) states that the latency is simply the time needed to perform computation at UE if the task is not offloaded, and (5) is due to [34].

Likewise, let $F_i^k$ be the amount of CPU resource at server $k$ that allocated to service $i$. We denote by $T_{ij}^{k,u,k}$ and $E_{ij}^{k,u,k}$ the latency and energy consumption when task $A_{ij}^{k,u}$ is offloaded locally at BS $k$, respectively. We have

$$T_{ij}^{k,u,k} = \frac{L_{ij}}{w_u^k} + \frac{L_{ij}\times C_{ij}}{F_i^k}, \quad (6)$$

$$E_{ij}^{k,u,k} = p_u^k \times \frac{L_{ij}}{w_u^k}, \quad (7)$$

where (6) indicates that when offloading, the latency of a task includes the time to upload data and perform computation at the MEC server[2], and (7) reflects that the energy consumption at UE is simply the energy taken for data uploading.[3]

In a Mobile-Edge Computing-based network, a task can be potentially offloaded to a nearby BS. Let $\mathcal{I}(k)$ be the set of neighbors of server $k$ (not necessarily one-hop away). When $A_{ij}^{k,u}$ is routed to and computed at a server $s \in \mathcal{I}(k)$, the energy consumption $E_{ij}^{k,u,s}$ and delay $T_{ij}^{k,u,s}$ of performing the task can be calculated as

$$T_{ij}^{k,u,s} = \frac{L_{ij}}{w_u^k} + \frac{L_{ij}\times C_{ij}}{F_i^s}, \quad (8)$$

$$E_{ij}^{k,u,s} = p_u^k \times \frac{L_{ij}}{w_u^k}. \quad (9)$$

where $F_i^s$ is the CPU resource allocated to service $i$ at server $s$. Notice that $E_{ij}^{k,u,k} = E_{ij}^{k,u,s}$ from (7) and (9). This is for the fact that back-haul links usually have much higher bandwidth than access links [20]. In other words, we can approximately conceive the benefit of scheduling a task from a local server to a remote server is due to the superior computing resources at the remote server.

In this work, we consider a probabilistic task/request scheduling policy. Let $Pr_{ij}^{k,u,s} \in [0,1]$ be the probability that task $A_{ij}^{k,u}$ from server $k$ is routed to server $s \in \mathcal{I}(k)$, and $Pr_{ij}^{k,u,k}$ be the probability that it is offloaded to the local server. We also denote by $Pr_{ij}^{k,u,\text{UE}}$ be the probability that $A_{ij}^{k,u}$ is performed at UE. To ensure that every task is processed, these task scheduling/routing variables should satisfy

$$\sum_{s\in\mathcal{I}(k)} Pr_{ij}^{k,u,s} + Pr_{ij}^{k,u,k} + Pr_{ij}^{k,u,\text{UE}} = 1, \quad \forall k \in \mathcal{M}, u \in \mathcal{L}(k). \quad (10)$$

[2]Following the common practice [16], we assume a negligible amount of post-processing data to be transmitted back to the UE.

[3]Since our goal is to optimize users' QoE, we ignore energy consumption at MEC servers.

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

$$G_{ij}{}^{k,u,s}(F_i^s, \alpha_u^k, P_u^k) = \alpha_{ij} \times \frac{E_{ij}^{k,u,\mathrm{UE}} - E_{ij}^{k,u,s}}{E_{ij}^{k,u,\mathrm{UE}}} + \beta_{ij} \times \frac{T_{ij}^{k,u,\mathrm{UE}} - T_{ij}^{k,u,s}}{T_{ij}^{k,u,\mathrm{UE}}} \tag{11}$$

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 $\alpha_{ij} = 1$, the problem becomes that of minimizing energy consumption, while $\alpha_{ij} = 0$ means the goal is to minimize task latency. Note that these two weights can vary from task to task. Moreover, by setting $F_i^{\mathrm{UE}} = f_u^k$ we have $G_{ij}^{k,u,\mathrm{UE}}(F_i^{\mathrm{UE}}, \alpha_u^k, P_u^k) = 0$, i.e., local computation at UE brings no gain.

*3) Storage Model:* We assume that each MEC server has limited storage resources for hosting services.[4] Let $S^k$ be the storage capacity at server $k$, and $S_i$ be the size of service $i$. Denote by $x_i^k \in \{0,1\}$ be a binary variable indicating whether service $i$ is cached at server $k$. We have the following storage constraint at each server $k$

$$\sum_{i \in \mathcal{N}} S_i \times x_i^k \le S^k, \quad \forall k \in \mathcal{M} \tag{12}$$

*4) Design Objectives:* Let $\lambda_{ij}^{k,u}$ be the arrival rate of task $A_{ij}^{k,u}$, and $F^k$ be the total CPU resources at server $k$. Also denote by $W^k$ be the communication capacity at server $k$, and $\mathcal{J}(k) = \mathcal{I}(k) \cup \{k\}$. With these notations, we can now formulate the *joint service placement, resource allocation, power management and task scheduling problem* for the MEC-based network, with the goal to maximize users' QoE, as the following optimization problem:

$$\underset{\{F_i^s, x_i^k, \alpha_u^k, P_u^k, Pr_{ij}^{k,u,s}\}}{\text{Maximize:}} \quad \sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{N}(i)} \sum_{k \in \mathcal{M}} \sum_{u \in \mathcal{L}(k)} \sum_{s \in \mathcal{J}(k)} \lambda_{ij}^{k,u} G_{ij}^{k,u,s}(F_i^s, \alpha_u^k, P_u^k) Pr_{ij}^{k,u,s} \tag{13a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{R}} F_i^k \le F^k, \quad \forall k, \tag{13b}$$

$$\sum_{i \in \mathcal{N}} S_i \times x_i^k \le S^k, \quad \forall k, \tag{13c}$$

$$\sum_{s \in \mathcal{J}(k)} Pr_{ij}^{k,u,s} \le 1, \quad \forall i, j, k, u \tag{13d}$$

$$\sum_{s \in \mathcal{I}(k)} \sum_{u \in \mathcal{L}(s)} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}(i)} \lambda_{ij}^{s,u} \times L_{ij} \times Pr_{ij}^{s,u,k} \le W^k, \forall k, \tag{13e}$$

$$\frac{\sum_{s \in \mathcal{J}(k)} \sum_{u \in \mathcal{L}(k)} \sum_{j \in \mathcal{N}(i)} \lambda_{ij}^{k,u} Pr_{ij}^{k,u,s}}{\sum_{u \in \mathcal{L}(k)} \sum_{j \in \mathcal{N}(i)} \lambda_{ij}^{k,u}} \ge h_i^k, \forall i, k, \tag{13f}$$

$$\sum_{u \in \mathcal{L}(k)} \alpha_u^k \le 1, \quad \forall k, \tag{13g}$$

$$0 \le P_u^k \le P_u^{\mathrm{MAX}}, \quad \forall k, u, \tag{13h}$$

$$0 \le F_i^k \le F^{\mathrm{MAX}}, \quad \forall i, k, \tag{13i}$$

$$0 \le \alpha_u^k \le 1, \quad \forall k, u, \tag{13j}$$

$$0 \le Pr_{ij}^{k,u,s} \le x_i^s, \quad \forall i, j, k, u, s, \tag{13k}$$

$$x_i^k \in \{0,1\}, \quad \forall i, k, \tag{13l}$$

where $F^{\mathrm{MAX}}$ is the maximum amount of CPU resource that a service can be allocated to by an MEC server. Note that constraint (13d) is different from (10) since executing a task at UE brings no performance gain and we therefore neglect it in the above problem formulation. Constraint (13b) and (13c) represent the CPU and storage resource limitations at an MEC server, respectively, and (13e) states that the volume of tasks scheduled to a server should not exceed its communication capacity. Constraint (13k) tells that a request will be directed to an MEC server only if the corresponding service is hosted at that server.

Meanwhile, rather than the widely concerned delay constraints in most existing work, here we introduce a new requirement (13f), which states that the proportion of tasks for each service processed by the MEC-based network should be no less than a preset threshold, i.e., $h_i^k \in [0,1]$. This constraint can be actually regarded as a QoS requirement from service providers. Indeed, from service providers' point of view, the most significant benefit of MEC is that delay-sensitive and computation-intensive tasks can be executed at the network edge, so that resource consumption at UEs can be dramatically reduced. The MEC infrastructure provider may offer this option to a service provider who has stringent offloading requirement, i.e., as a new metric in their SLA (Service Level Agreement), and this threshold can vary from service to service, and possibly from BS to BS.

*Theorem II.1: The offline joint optimization problem as stated in (13) is NP-hard.*

*Proof:* We prove the theorem by contradiction, by examining the following problem (14), which is obtained by considering the network with only one BS (BS $k$) and when all services are of the same sizes (i.e., $S_1 = S_2 = \cdots = S$), and under a fixed bandwidth allocation, power assignment and request routing $\{\alpha_u^{k'}, P_u^{k'}, Pr_{ij}^{k,u,k'}\}$

$$\underset{\{F_i^k\}}{\text{Maximize:}} \quad \sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{N}(i)} \sum_{u \in \mathcal{L}(k)} \lambda_{ij}^{k,u} \max\{G_{ij}^{k,u,k}(F_i^k, \alpha_u^{k'}, P_u^{k'}), 0\} Pr_{ij}^{k,u,k'} \tag{14a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{R}} F_i^k \le F^k, \tag{14b}$$

[4] In line with the current technology, one can use a VM or container to provide a service, which requires certain amount of resources (i.e., CPU, memory) from the host machine. Since the capacity of hard disk is much larger than that of memory, the service storage constraint that we considered in this work is mainly for the memory.

$$\text{Card}(\boldsymbol{F^k}) \le S^k/S, \tag{14c}$$

$$0 \le F_i^k \le F^{\text{MAX}}, \quad \forall i, \tag{14d}$$

where $\boldsymbol{F^k} = (F_1^k, F_2^k, \ldots, F_N^k)$, and $\text{Card}(\boldsymbol{F^k})$ is the cardinality function in $\boldsymbol{F^k}$, i.e., it equals to the number of non-zero elements in $\boldsymbol{F^k}$.

Based on (11), we can see that the above problem is an instance of problems of minimizing a sum of quasi-convex functions with a convex constraint together with a cardinality constraint (Observe that the function $\sum_{i\in\mathcal{R}}\sum_{j\in\mathcal{N}(i)}\sum_{u\in\mathcal{L}(k)}\lambda_{ij}^{k,u}G_{ij}^{k,u,k}(F_i^k,\alpha_u^{k'},P_u^{k'})Pr_{ij}^{k,u,k'}$ is concave in $\boldsymbol{F^k}$). This problem is NP-hard, since minimizing a sum of convex functions (which is also convex) with a convex constraint and a cardinality constraint has been shown NP-hard [10] [27].

Now suppose there exists a polynomial-time algorithm to problem (13). Setting $\mathcal{M} = \{k\}$, $S_1 = S_2 = \cdots = S$, we are able to obtain an optimal solution, say, $\{F_i^{k^*}, x_i^{k^*}, \alpha_u^{k^*}, P_u^{k^*}, Pr_{ij}^{k,u,k^*}\}$, where $x_i^{k^*} = 1$ if $F_i^{k^*} > 0$ and $x_i^{k^*} = 0$ otherwise. Obviously, this solution also solves problem (14) under the same $\{\alpha_u^{k^*}, P_u^{k^*}, Pr_{ij}^{k,u,k^*}\}$, and this contradicts the conclusion that (14) is NP-hard. □

## III. A Two-Stage Algorithm

### A. Preliminaries

Given that problem (13) is NP-hard, we seek efficient heuristic algorithms for approximate solution, by exploiting its structural properties. Observe that the hardness of the problem lies in the following facts: 1) lack of nice mathematical properties, i.e., it is not convex. In particular, the objective function is non-differentiable at the boundary of the feasible region, i.e., when $F_i^k = 0$, or $\alpha_u^k = 0$, or $P_u^k = 0$; and 2) the integer (0-1) constraints at each server.

To solve the problem efficiently, we closely approximate it with another one which is much easier to handle. More specifically, we first impose a positive lower bound $\epsilon > 0$ for the concerned variables, i.e., $F_i^k \ge \epsilon$, $P_u^k \ge \epsilon$, and $\alpha_u^k \ge \epsilon$. This ensures that the objective function is differentiable while at the same time, the optimal value of the modified problem is expected to be close to the original one, as long as $\epsilon \in (0, 1)$ is chosen sufficiently small. Next, rather than targeting the problem directly, we focus instead on the problem with no storage constraints. This leads us to the following problem variant

$$\underset{\{F_i^s,\alpha_u^k,P_u^k,Pr_{ij}^{k,u,s}\}}{\text{Maximize:}} \quad \sum_{i\in\mathcal{R}}\sum_{j\in\mathcal{N}(i)}\sum_{k\in\mathcal{M}}\sum_{u\in\mathcal{L}(k)}\sum_{s\in\mathcal{J}(k)}\lambda_{ij}^{k,u}G_{ij}^{k,u,s}(F_i^s,\alpha_u^k,P_u^k)Pr_{ij}^{k,u,s} \tag{15a}$$

$$\text{s.t.} \quad (13b),\ (13d),\ (13e),\ (13f),\ (13g) \tag{15b}$$

$$\epsilon \le P_u^k \le P_u^{\text{MAX}}, \quad \forall k, u, \tag{15c}$$

$$\epsilon \le F_i^k \le F^{\text{MAX}}, \quad \forall i, k, \tag{15d}$$

$$\epsilon \le \alpha_u^k \le 1, \quad \forall k, u, \tag{15e}$$

$$0 \le Pr_{ij}^{k,u,s} \le 1, \quad \forall i, j, k, u, s, \tag{15f}$$

**Algorithm 1:** Iterative Algorithm for Solving Problem (15).

1: Select an initial resource allocation and request routing $\{F_i^{k^0}, Pr_{ij}^{k,u,s^0}\}$ subject to (13b)(13d)(13e)(13f)(13i)(13k);
2: $T \leftarrow 0$; $G^0 \leftarrow \infty$;
3: **for** $T \le T^{\text{MAX}}$ **do**
4: Solve problem (16) with $\{F_i^{k^T}, Pr_{ij}^{k,u,s^T}\}$ and obtain $\{\alpha_u^{k^T}, P_u^{k^T}\}$;
5: Solve problem (23) with $\{\alpha_u^{k^T}, P_u^{k^T}\}$ and obtain $\{F_i^{k^{T+1}}, Pr_{ij}^{k,u,s^{T+1}}\}$, and $G^{T+1}$;
6: **if** $|G^{T+1} - G^T| < \xi$ **then**
7: break;
8: $T \leftarrow T + 1$.
9: Return $\{F_i^{s^T}, \alpha_u^{k^T}, P_u^{k^T}, Pr_{ij}^{k,u,s^T}\}$.

It is not hard to see that problem (15) is a nonlinear programming problem (NLP) with a continuously differentiable objective function and multiple linear constraints, i.e., the feasible region is a polyhedron. Furthermore, the constraints are not coupled with respect to different type of decision variables. This allows us to apply general NLP solvers leveraging the information of first-order or second-order derivatives. In this work, instead of relying on these existing tools, we propose the following algorithm which fully exploits the structural properties of the problem:

Let $G$ be the objective function of problem (15). Since $\underset{\{F_i^s,\alpha_u^k,P_u^k,Pr_{ij}^{k,u,s}\}}{\text{Maximize}} \ G = \underset{\{F_i^s,Pr_{ij}^{k,u,s}\}}{\text{Maximize}} \{\underset{\{\alpha_u^k,P_u^k\}}{\text{Maximize}} \ G\}$, we can solve problem (15) by decomposing it into two sub-problems, where the first (and lower-layer) problem is to determine the optimal $\{\alpha_u^k, P_u^k\}$ under a fixed $\{F_i^s, Pr_{ij}^{k,u,s}\}$, and the second (and upper-layer) problem is to obtain the optimal $\{F_i^s, Pr_{ij}^{k,u,s}\}$ given a fixed $\{\alpha_u^k, P_u^k\}$. The algorithm works by iteratively solving these two sub-problems until convergence, as depicted in Algorithm 1 (here $\xi > 0$ is the parameter of accuracy).

More specifically, the lower-layer sub-problem we consider can be formulated as follows:

$$\underset{\{\alpha_u^k,P_u^k\}}{\text{Maximize:}} \quad \sum_{i\in\mathcal{R}}\sum_{j\in\mathcal{N}(i)}\sum_{k\in\mathcal{M}}\sum_{u\in\mathcal{L}(k)}\sum_{s\in\mathcal{J}(k)}\lambda_{ij}^{k,u}G_{ij}^{k,u,s}(F_i^s,\alpha_u^k,P_u^k)Pr_{ij}^{k,u,s} \tag{16a}$$

$$\text{s.t.} \quad (13g),\ (13h),\ (13j),\ (15c),\ (15e) \tag{16b}$$

Note that since the objective function is separable and the constraints are not coupled in terms of individual BSs, we can tackle the above problem by solving $M$ sub-problems, each one for a particular BS. For example, the problem for BS $k$ is

$$\underset{\{\alpha_u^k,P_u^k\}}{\text{Maximize:}} \quad \sum_{i\in\mathcal{R}}\sum_{j\in\mathcal{N}(i)}\sum_{u\in\mathcal{L}(k)}\sum_{s\in\mathcal{J}(k)}\lambda_{ij}^{k,u}G_{ij}^{k,u,s}(F_i^s,\alpha_u^k,P_u^k)Pr_{ij}^{k,u,s} \tag{17a}$$

$$\text{s.t.} \quad \sum_{u\in\mathcal{L}(k)} \alpha_u^k \le 1, \tag{17b}$$

$$\epsilon \le P_u^k \le P_u^{\text{MAX}}, \quad \forall u, \tag{17c}$$

$$\epsilon \le \alpha_u^k \le 1, \quad \forall u, \tag{17d}$$

The above problem can be solved by the following nonlinear Gauss-Seidel method as proposed in [17] [36]

(1) Obtaining the optimal transmit power for a fixed bandwidth allocation. The optimal transmit power $\{P_u^k\}$, given $\{\alpha_u^k\}$, is achieved at either the stationary point of the objective function or one of the boundary points, which can be calculated as follows:

$$P_u^k = \underset{P_u^k\in\{v\in\mathcal{H}|\epsilon\le v\le P_u^{\text{MAX}}\}}{\text{argmax:}} \; G^{\text{k}}, \tag{18}$$

$$\mathcal{H} = \left\{\epsilon, P_u^{\text{MAX}}\right\} \cup \left\{P_u^k | \frac{dG^{\text{k}}}{dP_u^k} = 0\right\}, \tag{19}$$

where $G^{\text{k}} = \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{N}(i)} \sum_{s\in\mathcal{J}(k)} \lambda_{ij}^{k,u} G_{i,j}^{k,u,s} Pr_{ij}^{k,u,s}$.

(2) Obtaining the optimal wireless bandwidth allocation for a fixed transmit power. Given $\{P_u^k\}$, the optimal wireless bandwidth allocation $\{\alpha_u^k\}$ can be obtained by solving the following problem

$$\underset{\{P_u^k\}}{\text{Maximize:}} \quad \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{N}(i)} \sum_{u\in\mathcal{L}(k)} \sum_{s\in\mathcal{J}(k)} \lambda_{ij}^{k,u} G_{ij}^{k,u,s}(F_i^s, \alpha_u^k, P_u^k) Pr_{ij}^{k,u,s} \tag{20a}$$

$$\text{s.t.} \quad (17\text{b}), (17\text{d}) \tag{20b}$$

The structure of problem (20) suggests a Lagrangian-based method to solve it efficiently. More specifically, let $G^{\#}$ be the objective function and $\gamma > 0$ be the Lagrangian multiplier associated with the constraint (17b), the partial Lagrangian function can be written as follows:

$$\mathcal{L}\left(\{\alpha_u^k\}, \gamma\right) = -1.0 \times G^{\#} + \gamma \left(\sum_{u\in\mathcal{L}(k)} \alpha_u^k - 1\right), \tag{21}$$

Based on KKT conditions, the optimal allocation $\alpha_u^{k^*}$ and optimal Lagrangian multiplier $\gamma^*$ should satisfy

$$\alpha_u^{k^*} = \begin{cases} \max\{\epsilon, \psi_u(\gamma^*)\}, & \forall u, \gamma^* > 0 \\ \sum_{u\in\mathcal{L}(k)} \alpha_u^{k^*} \le 1 \end{cases}, \tag{22}$$

where $\psi_u(\gamma)$ denotes the root of $-1.0 \times \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{N}(i)} \sum_{s\in\mathcal{J}(k)} \lambda_{ij}^{k,u} \times \frac{dG_{ij}^{k,u,s}}{d\alpha_u} \times Pr_{ij}^{k,u,s} = \gamma$, which is positive and unique as $-1.0 \times \frac{dG_{ij}^{k,u,s}}{d\alpha_u} > 0$ decreases with $\alpha_u$. Let $\gamma_{\text{L}} = \max_u\{-1.0 \times \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{N}(i)} \sum_{s\in\mathcal{J}(k)} \lambda_{ij}^{k,u} \times \frac{dG_{ij}^{k,u,s}}{d\alpha_u} \times Pr_{ij}^{k,u,s}|_{\alpha_u=1}\}$, and define $\gamma_{\text{U}}$ such that $\max\{\epsilon, \psi_u(\gamma_{\text{U}})\} < 1$. A bisection search over $[\gamma_{\text{L}}, \gamma_{\text{U}}]$ can be leveraged for the optimal $\gamma^*$. The search process terminates whenever $|\sum_{u\in\mathcal{L}(k)} \max\{\epsilon, \psi_u(\gamma^*)\} - 1| < \eta$, where $\eta > 0$ is the accuracy of the algorithm.

In short, the Gauss-Seidel method iteratively updates the transmit power and wireless bandwidth allocation until convergence. The derived solution at each server is then adopted as the solution to the lower-layer problem (16).

On the other hand, the upper-layer problem we consider is

$$\underset{\{F_i^s, Pr_{ij}^{k,u,s}\}}{\text{Maximize:}} \quad \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{N}(i)} \sum_{k\in\mathcal{M}} \sum_{u\in\mathcal{L}(k)} \sum_{s\in\mathcal{J}(k)} \lambda_{ij}^{k,u} G_{ij}^{k,u,s}(F_i^s, \alpha_u^k, P_u^k) Pr_{ij}^{k,u,s} \tag{23a}$$

$$\text{s.t.} \quad (13\text{b}), (13\text{d}), (13\text{e}), (13\text{f}), (13\text{k}), (15\text{d}) \tag{23b}$$

Based on (11), it can be seen that the above problem belongs to the one of maximizing a sum of linear fractions with multiple linear constraints, to which efficient global optimization algorithms have been recently proposed [11]. Alternatively, we can also cast the problem as a Quadratically-Constrained Quadratic Programming problem (QCQP, here actually bilinear) that can be efficiently and approximately addressed [32] [7], through introducing auxiliary variables $z_i^s$ for each $F_i^s$, and new constraints $F_i^s \times z_i^s = 1, \forall i, s$. The objective function then becomes quadratic in $z_i^s$ and $Pr_{ij}^{k,u,s}$, and the constraints are also quadratic.

### B. 2-Stage Algorithm

We now elaborate on our 2-stage algorithm. Let $\{F_i^{k'}, \alpha_u^{k'}, P_u^{k'}, Pr_{ij}^{k,u,s'}\}$ be the solution to problem (15), which satisfies all CPU resource allocation, wireless bandwidth allocation, power management and request routing constraints in problem (13). Denote by $\mathcal{H}(k)$ be the set of services which have stringent QoS requirements at BS $k$, i.e., $\mathcal{H}(k) = \{i|h_i^k > 0\}$. Note that these services have priorities over the others and need to be preferentially cached. To have the remaining storage constraints satisfied, we formulate the following 0-1 programming problem for each server $k$

$$\underset{\{x_i^k\}}{\text{Maximize:}} \quad \sum_{i\in\mathcal{R}} G_i^k(F_i^{k'}) \times x_i^k \tag{24a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{N}} S_i \times x_i^k \le S^k, \tag{24b}$$

$$x_i^k = 1, \quad \forall i \in \mathcal{H}(k) \tag{24c}$$

$$x_i^k \in \{0,1\}, \quad \forall i \in \mathcal{R} \setminus \mathcal{H}(k) \tag{24d}$$

where $G_i^k(F_i^{k'})$ is the aggregate gain from hosting service $i$ at server $k$, which can be characterized as

$$G_i^k(F_i^{k'}) = \sum_{j\in\mathcal{N}(i)} \sum_{s\in\mathcal{J}(k)} \sum_{u\in\mathcal{L}(s)} \lambda_{ij}^{s,u} G_{ij}^{s,u,k}(F_i^{k'}, \alpha_u^{s'}, P_u^{s'}) Pr_{ij}^{s,u,k'} \tag{25}$$

The above problem (24) is a 0-1 knapsack problem if we regard services as items, the aggregate gain from hosting services as values/profits, and the size of services as weights. This problem

has been studied, and both greedy approximation algorithms and fully polynomial-time approximation schemes [2] have been proposed. In particular, if we adopt the greedy approximation algorithms based on the ratios between values/profits and weights, then this leads us to the following important quantity which we call *resource efficiency* $E_i^k(F_i^{k'})$ of hosting service $i$ at server $k$ with $F_i^{k'}$, defined as

$$E_i^k(F_i^{k'}) = \frac{G_i^k(F_i^{k'})}{S_i} \tag{26}$$

Eq. (26) states that a high resource efficiency for hosting a service can be expected when a large aggregate gain is achieved while at the same time it consumes less storage resources.

Our 2-stage algorithm is based on the above solutions of service placement at MEC servers. The main idea is to decompose the original problem (13) into two sub-problems solved at each stage, during which we ensure certain constraints are satisfied. More specifically, at the very first stage we try to satisfy the storage and 0-1 constraints (see (13c), (13l)), by solving problem (15) for the network and then evaluating resource efficiency for services at each server (Algorithm 2, line 1–2). The services obtained by solving the knapsack problem for each server $k$ are then selected for potential resource allocation. If there are ties, then we select the services with the least consumed CPU resources (line 3). Once services are selected, it remains to obtain the appropriate resource allocation, task scheduling, bandwidth allocation and power management for the network. Again this is achieved through solving problem (15) at the second stage (line 4). Notice the difference is that now the problem is restricted to the services we have selected at the first stage, i.e., the objective function becomes

$$\sum_{k\in\mathcal{M}}\sum_{i\in\mathcal{R}(k)}\sum_{j\in\mathcal{N}(i)}\sum_{u\in\mathcal{L}(k)}\sum_{s\in\mathcal{J}(k)}\lambda_{ij}^{k,u}G_{ij}^{k,u,s}(F_i^s,\alpha_u^k,P_u^k)Pr_{ij}^{k,u,s}$$

where $\mathcal{R}(k)$ is the set of services selected at server $k$. In addition, we introduce a new routing constraint

$$Pr_{ij}^{k,u,s} = 0, \quad \text{if} \quad i \notin \mathcal{R}(s) \tag{27}$$

which states that a request will not be directed to a server if the corresponding service is not cached.

Obviously, all constraints can be simultaneously satisfied when our algorithm terminates. Meanwhile, low complexity can be expected given that problem (15) can be efficiently solved, since evaluating resource efficiency for each service incurs negligible computation cost (see (25)–(26)).

*Remarks (insights behind our algorithm):* The sub-problems solving structure of our algorithm makes it suitable to be implemented at different layers. For example, the Gauss-Seidel iterative process for obtaining the appropriate wireless bandwidth allocation and power management can be carried out at the communication layer, whereas requests routing and service caching can be operated at the network and service/application layer, respectively. Meanwhile, it also admits a multi-time-scale framework that depends on system stability and operation cost, i.e., requests routing can be adjusted at a small time scale whereas service caching decisions may be made at a larger time scale.

**Algorithm 2:** A 2-Stage Algorithm.

1: Solve problem (15) and obtain a solution $\{F_i^{k'}, \alpha_u^{k'}, P_u^{k'}, Pr_{ij}^{k,u,s'}\}$;
2: Evaluate resource efficiency for services at each server based on (26);
3: Rank the services at each server $k$ according to their resource efficiency, and select the top most services sequentially until there is no storage resources left;
4: Solve problem (15) with the selected services at each server and new constraints (27), and obtain the solution $\{F_i^{s*}, \alpha_u^{k*}, P_u^{k*}, Pr_{ij}^{k,u,s*}\}$.

### C. Convergence and Complexity Analysis

The convergence of our two-stage algorithm is fully determined by the block nonlinear Gauss-Seidel (GS) method adopted for calculating bandwidth allocation and power management in the lower-layer sub-problem, given that the upper-layer sub-problem can be well solved by the global optimization algorithm [11]. Note that the GS method we adopted in this work is actually for two blocks (with two different blocks of variables updated iteratively), and it satisfies all the necessary conditions and requirements for convergence as presented in [22], which are as follows: 1) the objective function is continuously differentiable; 2) the feasible set is the Cartesian product of closed, nonempty and convex subsets; 3) the sub-problem for each block is well defined, i.e., it has an optimal solution; and 4) The Assumption 1 in [22] holds. As a result, the GS method adopted is convergent. Furthermore, Algorithm 1 can also be viewed as an instance of the two-block Gauss-Seidel method (with one block of variables being $\{F_i^{k^T}, Pr_{ij}^{k,u,s^T}\}$ and the other one $\{\alpha_u^{k^T}, P_u^{k^T}\}$), which again satisfies the above four conditions and requirements, leading to the fact that Algorithm 1 is also convergent. The convergence of Algorithm 2 is obvious since it merely involves calling Algorithm 1 twice, i.e., at the first stage we try to solve the problem with no storage constraints, and at the second stage we solve it with a fixed service placement.

The complexity of the algorithm comprises of two parts, namely, the iteration complexity (outer iterations) and the per-iteration computation (inner iterations). The complexity of the per-iteration computation can be easily characterized as all sub-problems are well defined and can be efficiently solved by existing algorithms, i.e., bi-section search over a line interval. The iteration complexity is relatively difficult to characterize. M. Patriksson in [44] has given a linear convergence rate of decomposition algorithms such as GS for continuously differentiable optimization problems over Cartesian products of convex sets, within the framework of cost approximation algorithms. Z.Q. Luo et al. [33] have proved that the sequence generated by iterative methods such as block coordinate decent converges at least linearly to a stationary point, by using the concept of error bounds and under some mild assumptions. Since the problem we address has identical structure properties, we expect the same convergence rate, although it still requires rigorous

mathematical proofs. All together, this implies polynomial-time complexity of the two-stage algorithm.

## IV. Decentralized Mechanism

In this section, we present a distributed algorithm to perform the joint optimization task for a MEC-based network, since the 2-stage algorithm proposed in the above section requires a central coordinator as well as the detailed information of workload and network conditions (e.g., task parameters, UEs/BSs configurations), which is often impossible as global information is generally hard to collect in many real-world applications due to security, management concerns, etc. On the other hand, decentralized mechanisms are able to achieve superior performance through nodes collaboration and local computation, and therefore they are more preferred. Moreover, the algorithm also needs to be adaptive to deal with the dynamics and uncertainty of both workload and network conditions.

### A. Distributed and Adaptive Algorithm

Our distributed and adaptive algorithm is based on the following three design principles: 1) minimum information sharing/exchange between a server and its neighbors; 2) confidentiality – each server should not expose its local traffic information to other servers; and 3) to leverage the offline problem (13) and its algorithmic solution as much as possible.

To start with, let $G_k(\boldsymbol{F_k}, \boldsymbol{Pr_k}, \boldsymbol{\alpha_k}, \boldsymbol{P_k}, \{\boldsymbol{F_l}\}_{l\in\mathcal{I}(k)})$ be the objective function at server $k$, i.e.,

$$G_k = \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{N}(i)} \sum_{u\in\mathcal{L}(k)} \sum_{s\in\mathcal{J}(k)} \lambda_{ij}^{k,u} G_{ij}^{k,u,s} Pr_{ij}^{k,u,s}$$

where $\{\boldsymbol{F_k}, \boldsymbol{Pr_k}, \boldsymbol{\alpha_k}, \boldsymbol{P_k}\}$ are local decision variables and $\{\boldsymbol{F_l}\}_{l\in\mathcal{I}(k)}$ are computing resource allocations from nearby nodes. Also, to design a fully distributed algorithm, we decouple the constraint (13e) by introducing $W^{s,k}$, which denotes the available network bandwidth at server $s$ that reserved to its neighbor $k \in \mathcal{I}(s)$. We thus get the following problem

$$\underset{\{\boldsymbol{F_k},\boldsymbol{\alpha_k},\boldsymbol{Pr_k},\boldsymbol{P_k}\}}{\text{Maximize:}} \sum_{k\in\mathcal{M}} G_k(\boldsymbol{F_k}, \boldsymbol{\alpha_k}, \boldsymbol{Pr_k}, \boldsymbol{P_k}, \{\boldsymbol{F_l}\}_{l\in\mathcal{I}(k)}) \tag{28a}$$

$$\text{s.t.} \quad (13b) \sim (13d), (13f) \sim (13l) \tag{28b}$$

$$\sum_{u\in\mathcal{L}(s)} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}(i)} \lambda_{ij}^{k,u,s} L_{ij} Pr_{ij}^{k,u,s} \le W^{s,k}, \forall k, s \in \mathcal{I}(k), \tag{28c}$$

where the constraint (13e) is replaced by (28c), which tells that the requests from server $k$ routed to server $s \in \mathcal{I}(k)$ should not exceed $s$'s network bandwidth allocated to $k$.

It is not hard to see that the above problem is the one with coupled objectives but the constraints are not coupled. One way to decompose it is to introduce auxiliary variables and add additional equality constraints, and then apply the Lagrangian-dual based approaches [51]. More specifically, for each server $k$ we introduce auxiliary variables $\{\boldsymbol{F_{kl}}\}_{l\in\mathcal{I}(k)}$ for the coupled arguments $\{\boldsymbol{F_l}\}_{l\in\mathcal{I}(k)}$, and the equality constraints to enforce consistency

$$\underset{\{\boldsymbol{F_k},\boldsymbol{\alpha_k},\boldsymbol{Pr_k},\boldsymbol{P_k},\{\boldsymbol{F_{kl}}\}\}}{\text{Maximize:}} \sum_{k\in\mathcal{M}} G_k(\boldsymbol{F_k}, \boldsymbol{\alpha_k}, \boldsymbol{Pr_k}, \boldsymbol{P_k}, \{\boldsymbol{F_{kl}}\}_{l\in\mathcal{I}(k)}) \tag{29a}$$

$$\text{s.t.} \quad (13b) \sim (13d), (13f) \sim (13l), (28c) \tag{29b}$$

$$\boldsymbol{F_{kl}} = \boldsymbol{F_l}, \quad \forall k, \quad l \in \mathcal{I}(k) \tag{29c}$$

Let $\boldsymbol{\gamma_{kl}}$ be the *consistency prices* [51] for the equality constraint (29c), we get the following relaxed problem

$$\underset{\{\boldsymbol{F_k},\boldsymbol{\alpha_k},\boldsymbol{Pr_k},\boldsymbol{P_k},\{\boldsymbol{F_{kl}}\}\}}{\text{Maximize:}} \sum_{k\in\mathcal{M}} G_k - \boldsymbol{\gamma_{kl}}^T(\boldsymbol{F_{kl}} - \boldsymbol{F_l}) \tag{30a}$$

$$\text{s.t.} \quad (13b) \sim (13d), (13f) \sim (13l), (28c) \tag{30b}$$

Next, denote by $g(\boldsymbol{\gamma_{kl}})$ the optimal value of problem (30), the dual problem is given by

$$\underset{\{\boldsymbol{\gamma_{kl}}\}}{\text{Minimize:}} \; g(\{\boldsymbol{\gamma_{kl}}\}) \tag{31}$$

It is well known that the dual problem is convex even when the original problem is not [12].

A key observation of problem (30) is that it is separable in terms of individual servers. That is, given $\{\boldsymbol{\gamma_{kl}}\}_{l\in\mathcal{I}(k)}$ and $\boldsymbol{F_l}$, server $k$ can fully determines its local variables $(\boldsymbol{F_k}, \boldsymbol{\alpha_k}, \boldsymbol{Pr_k}, \boldsymbol{P_k}, \{\boldsymbol{F_{kl}}\})$. This property together with the standard dual-based decomposition method leads us to the following *fully distributed* algorithm:

*Step 1:* at each time $t$, based on the received $\boldsymbol{F_l}(t)$ from neighbors, each server $k$ updates its consistency prices according to the following rule (here $\beta$ is the step size)

$$\boldsymbol{\gamma_{kl}}(t+1) = \boldsymbol{\gamma_{kl}}(t) + \beta(\boldsymbol{F_l}(t) - \boldsymbol{F_{kl}}(t)), \quad \forall l \in \mathcal{I}(k)$$

*Step 2:* at each time $t$, based on the received consistency prices from neighbors, each server $k$ locally solves its optimization problem:

$$\underset{\{\boldsymbol{F_k},\boldsymbol{\alpha_k},\boldsymbol{Pr_k},\boldsymbol{P_k},\{\boldsymbol{F_{kl}}\}\}}{\text{Maximize:}} G_k + \left(\sum_{l:k\in\mathcal{I}(l)} \boldsymbol{\gamma_{lk}}\right)^T \boldsymbol{F_k} - \sum_{l\in\mathcal{I}(k)} \boldsymbol{\gamma_{kl}}^T \boldsymbol{F_{kl}} \tag{32a}$$

$$\text{s.t.} \quad (13b) \sim (13d), (13f) \sim (13l), (28c) \tag{32b}$$

$$\sum_{i\in\mathcal{R}} F_i^{kl} \le F^l, \forall l \in \mathcal{I}(k) \tag{32c}$$

$$\epsilon \le F_i^{kl} \le F^{\text{MAX}}, \forall i \in \mathcal{R}, l \in \mathcal{I}(k) \tag{32d}$$

here, the constraints (13b)∼(13d), (13f)∼(13l) and (28c) are for server $k$ only.

*Step 3:* at each time $t$, after calculation each server $k$ broadcasts to its neighbors the following three messages: 1) its computing capacity $F^k$ and service resource allocations $\{F_i^k(t)\}$, i.e., the resource allocated to each service $i$ at time $t$; 2) available network bandwidth reserved to its neighbors $\{W^{k,s}(t)\}$; and 3) the consistency prices $\{\boldsymbol{\gamma_{kl}}(t)\}$, as shown in Fig. 2.

Note that problem (32) essentially has the same properties as problem (13) (the added terms in the objective function are linear). As a result, the same 2-stage algorithms proposed in

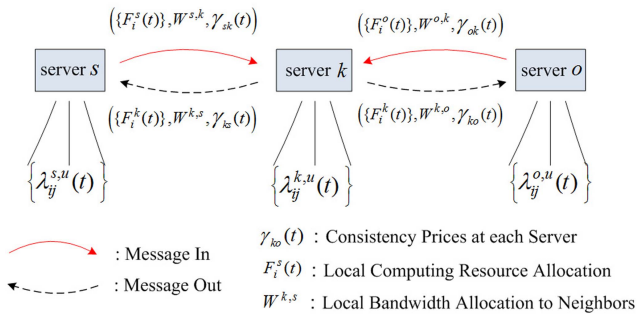



Fig. 2. Information exchange between neighbors in the proposed distributed mechanism.

Section III with the following two adaptations, can be applied: 1) the algorithm is executed locally at server $k$ and incurs less computational cost, and 2) the aggregate gain $G_i^k(F_i^{k\prime})$ for evaluating resource efficiency becomes the objective function value of (32), which reflects the fact that in the distributed environment each server $k$ can only use its local traffic information.

It remains to specify the bandwidth allocations $\{W^{s,k}\}$ in order to implement the distributed algorithm. There are several ways to do this. One possible choice is for each server $s$ to equally allocate its bandwidth among its neighbors $\mathcal{I}(s)$, i.e., $W^{s,k} = \frac{W^s}{|\mathcal{I}(s)|}, \forall k \in \mathcal{I}(s)$. A more reasonable choice is to distribute its bandwidth based on bandwidth allocations from its neighbors $\mathcal{I}(s)$, i.e., $W^{s,k} = W^s \times \frac{W^{k,s}}{\sum_{p\in I(s)} W^{p,s}}, \forall k \in \mathcal{I}(s)$. Alternatively, we can also design bandwidth allocation schemes with incentive mechanisms such as in P2P-like systems [55] [29], so that MEC servers from different network operators are willing to collaborate and contribute their resources. Whereas we recognize that bandwidth allocation can essentially impact system performance, the design of optimal bandwidth allocation scheme is out of the scope of this paper, and we leave it for our future work.

### B. Cost-Aware Solution

We now focus on the jointly optimizing task in a more realistic scenario where *service switches* can occur frequently in a stochastic MEC-based network, i.e., when some new services "replace" the existing ones in an MEC server due to workload variations or change of network condition. According to the current technology, services are often hosted by virtual machines or containers, whose image (data and code) needs to be fetched and loaded into system before the service is available. Note that during this period of time, tasks cannot be executed at the server but instead they can only be performed at UE or be offloaded to other servers if the service is present. Likewise, allocating a service with different amount of resources also causes delays of the service. As a result, frequent service switches can significantly degrade system performance. The cost of service switches requires optimal control policy be *cost-aware*, i.e., it achieves high system performance while at the same time incurs less service switches.

To capture the cost of service switches into our algorithm design, we first give a formal definition of service switches.

*Definition IV.1:* We define switch of a service $i$ at server $k$ at time $t$ as allocating resources to service $i$ from $F_i^{k^t}$ to $F_i^{k^{t+1}}$.

Service switches can be classified into two categories according to their operation cost. The first one is to download a new service from remote cloud or a nearby server and then start it, i.e., $F_i^{k^t} = 0$ and $F_i^{k^{t+1}} \neq 0$. The second one is to re-allocate resources to an existing service, i.e., $F_i^{k^t} \neq 0$ and $F_i^{k^{t+1}} \neq 0$. Denote by $\Delta T_i^{k,t,\mathrm{dl}}$ and $\Delta T_i^{k,t,\mathrm{rl}}$ the time needed for these two kinds of switches for service $i$ at server $k$ at time $t$, respectively[5]. In general, we have $\Delta T_i^{k,t,\mathrm{rl}} < \Delta T_i^{k,t,\mathrm{dl}}$.

Accordingly, at arbitrary time $t$ we can divide the set of services at server $k$ into two disjoint sets $A_k^t$ and $B_k^t$ ($A_k^t \cap B_k^t = \emptyset$ and $A_k^t \cup B_k^t = \mathcal{R}$), which corresponds to the set of services that are hosted at the current time $t$ and that are not present but may be hosted at time slot $t+1$, respectively. Let $\Delta T$ be the length of a time slot. To ensure system stability and low operation cost, we set $\Delta T > \Delta T_i^{k,t,\mathrm{dl}}$. Given that during service switches tasks can only be processed at UEs or nearby servers, we can now formulate a *cost-aware* optimization problem for server $k$ at time $t$ as follows:

$$\underset{\left\{F_i^{k^{t+1}}, x_i^{k^{t+1}}, Pr_{ij}^{k,u,s^{t+1}}, \alpha_u^{k^{t+1}}, P_u^{k^{t+1}}\right\}}{\text{Maximize:}}$$

$$\sum_{i\in\mathcal{R}}\sum_{j\in\mathcal{N}(i)}\sum_{u\in\mathcal{L}(k)}\sum_{s\in\mathcal{I}(k)} \lambda_{ij}^{k,u^t} \times G_{ij}^{k,u,s^t} \times Pr_{ij}^{k,u,s^{t+1}} \tag{33a}$$

$$+\sum_{i\in A_k^t}\sum_{j\in\mathcal{N}(i)}\sum_{u\in\mathcal{L}(k)} \lambda_{ij}^{k,u^t} G_{ij}^{k,u,k^{t+1}} Pr_{ij}^{k,u,k^{t+1}} \frac{\Delta T - \Delta T_i^{k,t,\mathrm{rl}}}{\Delta T} \tag{33b}$$

$$+\sum_{i\in B_k^t}\sum_{j\in\mathcal{N}(i)}\sum_{u\in\mathcal{L}(k)} \lambda_{ij}^{k,u^t} G_{ij}^{k,u,k} Pr_{ij}^{k,u,k^{t+1}} \frac{\Delta T - \Delta T_i^{k,t,\mathrm{dl}}}{\Delta T} \tag{33c}$$

$$\text{s.t.,} \quad (13\text{b}) \sim (13\text{d}),\ (13\text{f}) \sim (13\text{l}),\ (28\text{c}) \tag{33d}$$

It is not hard to see that problem (33) is identical with the problem we consider in Section IV-A, with the only difference being in the objective functions. The first term (33a) of the objective function denotes the gains from offloading tasks to the nearby servers, whereas the sum of the rest terms denotes the gains from performing tasks at server $k$, which is further split into two separate ones ((33a) and (33c)) that respectively corresponds to the set $A_k^t$ and $B_k^t$. Notice that $\frac{\Delta T - \Delta T_i^{k,t,\mathrm{rl}}}{\Delta T}$ $\left(\frac{\Delta T - \Delta T_i^{k,t,\mathrm{dl}}}{\Delta T}\right)$ represents the fraction of time that requests from UEs can be served by the MEC server $k$ due to resource re-allocation (caching a new service), and thus it captures the cost of service switches during service selection. Problem (33) has exactly the same properties as the problem we addressed in Section IV-A and therefore the same algorithm can be applied. Here we want to emphasize that whereas services are relatively stable, $\Delta T_i^{k,t,\mathrm{dl}}$ ($\Delta T_i^{k,t,\mathrm{rl}}$)

[5] $\Delta T_i^{k,t,\mathrm{rl}}$ and $\Delta T_i^{k,t,\mathrm{dl}}$ can be specified by the network operator, or be measured online at each server as system operates.

can vary from server to server and time to time, depending on the network conditions, server computing workload, or their hardware capacity, etc.

## V. Performance Evaluation

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

### A. Benchmarks

We adopt the following three benchmarks for a comprehensive performance comparison:

(1) *Random-Caching*: This is the algorithm that randomly selects services to accommodate at each server (within its storage capacity).

(2) *Most-Caching*: This algorithm caches at each MEC server as many services as possible so as to keep the highest service diversity.

(3) *NLP-Solver*: This algorithm solves problem (15) using a standard NLP solver – Ipopt [54].

Note that for benchmarks (1) and (2), resource allocations, power management and requests routing are determined optimally by running our proposed algorithms, i.e., line 4 in Algorithm 2. For benchmark (3), we adopt the same procedure for selecting services as in Algorithm 2 (line 3), but with the difference that resource allocation, power management and requests routing are determined by the Ipopt solver.

### B. Evaluation Setup

For synthetic-workload simulations, we assume there are 50 services in system, i.e., $N = 50$. Among them, 3 services have QoS requirements with offloading rate randomly set from [0.1, 0.3]. The number of sub-types of each task is randomly picked from $\{1, 2, 3, 4, 5\}$, and these sub-types follow a Zipf distribution with skewdness parameter 1.2. Task parameters are configured as follows: the data size of each task $L_{ij}$ is randomly selected from {500 KB, 2000 KB, 3000 KB, 5000 KB, 10000KB}, and the computation intensity $C_{ij}$ from {100 b/cycle, 200 b/cycle, 300 b/cycle, 400 b/cycle, 500 b/cycle}. Note that these settings represent workload of a typical face recognition application [16] leveraging MEC. Meanwhile, we assume the number of UEs at each BS is chosen from $30 \sim 50$, and their CPU frequency is picked randomly from {0.5 GHz, 0.8 GHz, 1.0 GHz, 1.2 GHz}; The maximum transmission power of UEs is 2Watt, which is typical for a smartphone uploading data in commercial 4 G LTE networks [24]. The energy coefficient of all UEs are set equally as $K_u^k = K = 0.18 * 10^{-12}$.

For task arrival process, we assume the aggregate request rate at each MEC server is chosen uniformly from the interval $[0.5 \times 10^4$req/s, $1 \times 10^4$req/s], and these requests are for a subset $\lceil N/4 \rceil$ of services following a Zipf distribution with skewdness parameter 0.8. For tasks – UEs mapping, we assume requests for a specific service at each MEC server are generated by a subset of UEs that are randomly selected.

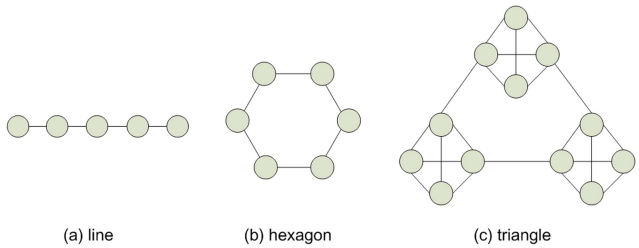


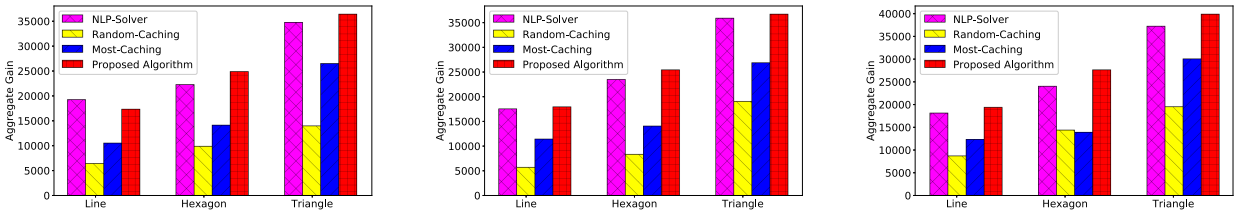

Fig. 3. Three network topologies adopted in simulation.

CPU resources and storage capacity of MEC servers are uniformly drawn from [50 GHz, 100 GHz] and [50 GB, 100 GB], respectively. The size of each service is picked from [3 GB, 10 GB]. In addition, we set the maximum CPU resource that a service can be allocated to as $F^{\text{MAX}} = 10$ GHz.

Communication-related parameters are configured as follows: the wireless bandwidth of each BS is set 40 MHz, and UEs are located at a distance of 100 m $\sim$150 m away from BS. The small scale fading channel power gains are assumed to be exponentially distributed with unit mean. As in [36], we set $g_0^k = g_0 = -40$ dB, $d_0^k = d_0 = 1$m, $\theta^k = \theta = 4$, $N_0^k = N_0 = -174$ dBm/Hz. Moreover, the communication capacity of each MEC server is drawn uniformly from [5 Gbps, 20 Gbps]. Unless otherwise specified, the weight for each task is set as $\alpha_{ij} = \alpha = 0.5$. Finally, to fully assess the performance, we adopt three different network topologies – a line with 5 nodes, a hexagon, and a triangle, as shown in Fig. 3, where we allow each server to forward its local traffic to nodes that are one-hop away. Note that each node in the triangle represents a 4-node mesh, which can reflect more realistic settings on geographical relationship and the possibility of clustering servers.

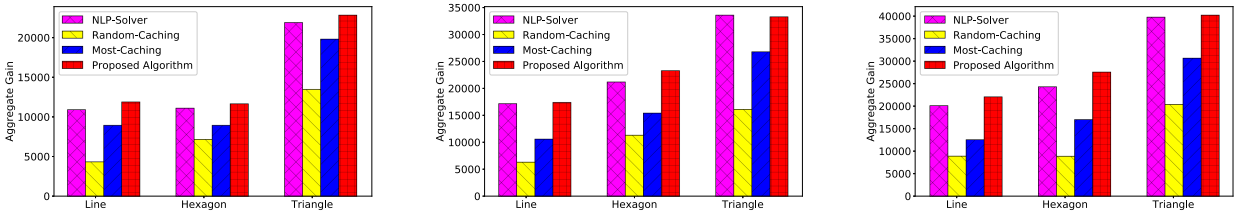
### C. Simulation Results

*1) Performance of Centralized Mechanism:* We first evaluate the performance of our algorithm in a *centralized* environment, i.e., when task parameters are given a priori and only a single time slot is considered (therefore no cost of service switch is incurred). Figs. 4–7 show system gains achieved by the four algorithms, with varying computing capacity, communication capacity, network bandwidth and weight parameter. From these figures, we can see that: 1) The NLP solver does not always provide the optimal performance, which is out of our expectation. In fact, in most cases it does not perform any better than our 2-stage algorithm; 2) Our proposed algorithm gives the best performance on average among the four algorithms, and as much as 18% improvement (see Fig. 6(a)) can be observed when it is compared with NLP solver; 3) The Most-Caching algorithm, which caches as many services as possible at each MEC server, also does not provide satisfactory performance as compared to the optimal solution; and 4) The Random-Caching algorithm, as we expected, performs worst. Moreover, according to Figs. 4–6, it is clear that except for Random-Caching, the



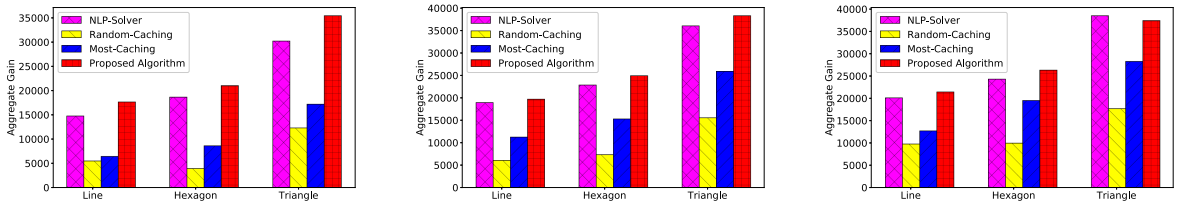

(a) $W^k = W$ = 5GHz (b) $W^k = W$ = 10GHz (c) $W^k = W$ = 20GHz

Fig. 4. Performance comparison of our algorithm and the baselines under varying communication capacity.



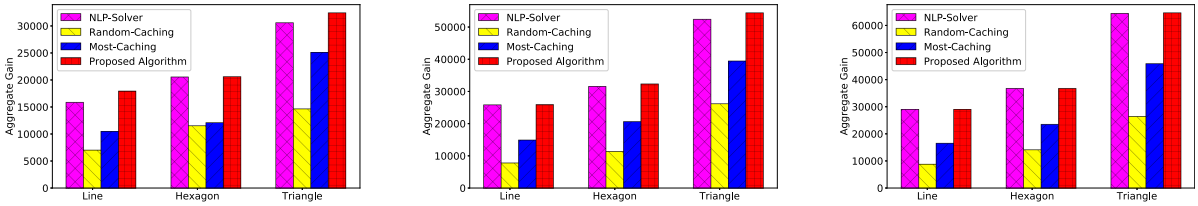

(a) $F^k = F$ = 20GHz (b) $F^k = F$ = 50GHz (c) $F^k = F$ = 100GHz

Fig. 5. Performance comparison of our algorithm and the baselines under varying computing capacity.




(a) $S^k = S$ = 40GB (b) $S^k = S$ = 70GB (c) $S^k = S$ = 90GB

Fig. 6. Performance comparison of our algorithm and the baselines under varying storage capacity.

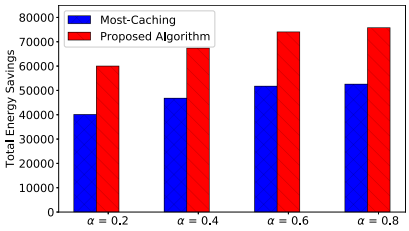



(a) $\alpha_{ij} = \alpha$ = 0.2 (b) $\alpha_{ij} = \alpha$ = 0.6 (c) $\alpha_{ij} = \alpha$ = 0.8

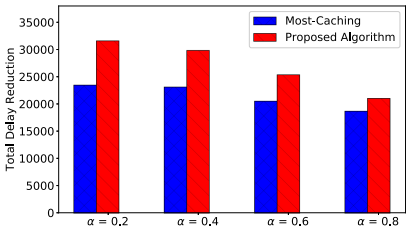
Fig. 7. Performance comparison of our algorithm and the baselines under varying weight parameter.

aggregate gains achieved by each algorithm over each topology increases as the capacity of the network grows. It is also clear that for each algorithm, the larger network scale, the higher aggregate gains.

Fig. 7 illustrates that system gains achieved by each algorithm can be impacted by the weight parameter $\alpha$. In particular, it is observed that gains grow as $\alpha$ becomes larger. Recall that $\alpha$ is the weight denoting the relative importance of optimizing energy consumption (see (11)), which implies that the larger $\alpha$, the more total energy savings. This property is validated in Fig. 8 where the actual delay reductions and energy savings are depicted. It can be seen that total energy savings is an increasing function in $\alpha$, whereas total delay reductions is non-increasing. In our experiment, since energy savings is much larger than

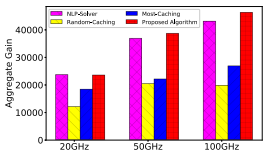



(a) Energy

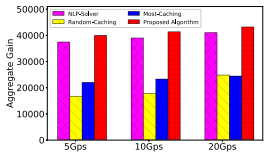



(b) Delay

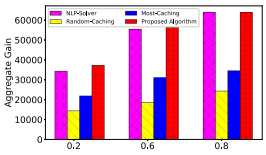
Fig. 8. Delay and energy consumption achieved by our algorithm and Most-Caching over the triangle topology.



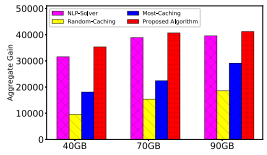

(a) with different CPU frequency




(b) with different bandwidth




(c) with different weight




(d) with different storage capacity

Fig. 9. Performance comparison of different algorithms under Abilene topology.

delay reductions, increasing $\alpha$ naturally leads to an increase in system gains, as shown in Fig. 7. Similar trends can be observed over other topologies, and these observations suggest that in practice we can tune system performance through regulating this parameter $\alpha$, i.e., balancing delays and energy consumption.

We also evaluate the performance of our algorithm under a real-world (and more complex) network topology – the Abilene network, with the same parameter settings as described above. The results are depicted in Fig. 9, where we can see that our algorithm again provides the best performance among these benchmarks, and the results are in accordance with that under the three synthetic topologies.

Since the original problem is NP-hard, it would be interesting to see the gap between our solution and the optimal one. To this end, we compare it with the solution to another optimization problem which is formulated as the one without the network

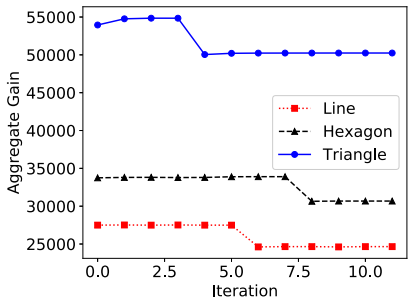
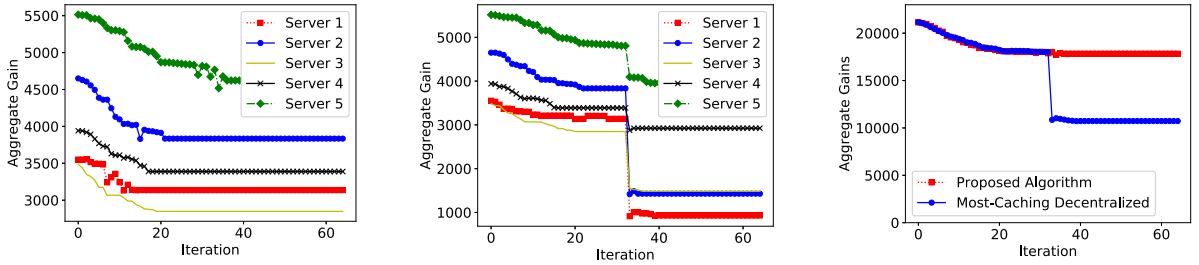



Fig. 10. Convergence of our algorithm over three network topologies.

bandwidth constraints and QoS requirements. Obviously, the solution to this optimization problem provides an upper bound for the original problem. Note that the MEC-based network then can be conceived as a Single-Server Multi-User system, where the set of MEC servers is regarded as a super-server whose computing, storage and radio resources are the sum of resources of individual servers, and the set of UEs consists of UEs from all servers. Meanwhile, it is also clear that for this problem, the optimal routing strategy is binary offloading, i.e., each UE either computes its task locally, or offloads it to the super-server. Although the problem is still NP-hard, for small scale problems the optimal solution can be found through exhaustive search. We make the following observations about this "super-server" system which facilitates the search of optimal solution: 1) A task will not be offloaded to the MEC server if the corresponding service is not hosted; 2) Given communication bandwidth and transmit power allocation at UEs, the problem of determining the optimal computing resource allocation among a given set of services is convex. The search of optimal solution to this "super-server" system then involves solving a series of convex optimization problems (and running GS algorithms), with each one corresponding to a specific task offloading strategy. Our numerical studies indicate that, for the network with 2 servers, 8 services (the tasks of each service are of the same sub-type), and 8 UEs at each BS/server, our algorithm achieves $82\% \sim 91\%$ of the optimal performance.

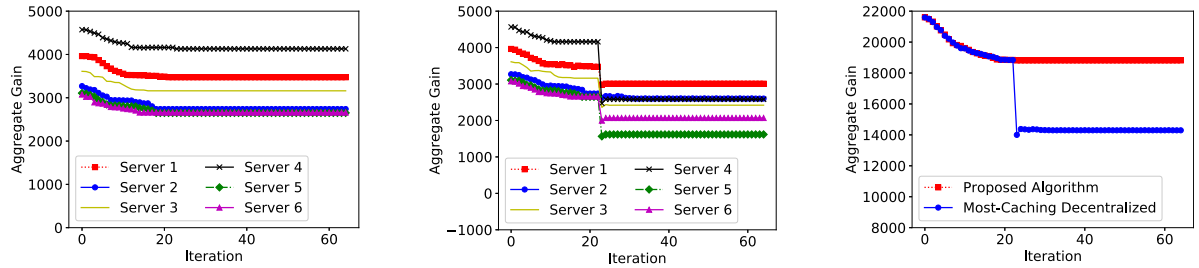
Next, to see how fast our algorithm is, we trace the aggregate gains returned by Algorithm 1, as shown in Fig. 10, where the drop of gains corresponds to selecting a subset of services to accommodate at each server. It can be seen that over all topologies our algorithm is stable and that it takes less than 8 iterations to converge.

*2) Performance of Decentralized Mechanism:* We use simulations to evaluate the performance of our algorithm in a *decentralized* environment. To this end, we set the length of each time slot as $\Delta T = 200$ sec. Both the gains by different algorithms and the number of service switches incurred in each time slot are investigate. Figs. 11–13 give the aggregate gains achieved in one slot at each server as the iteration process goes on, where the network bandwidth of each server is assumed to be evenly distributed among its neighbors, and there is no cost of service switches. From these figures, we can see that over all topologies and at all servers, our algorithm outperforms



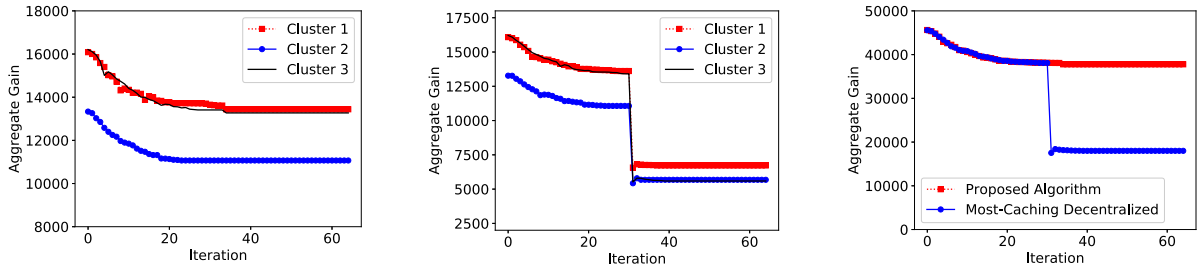

(a) gain at each server by our algorithm (b) gain at each server by Most-Caching (c) comparison with Most-Caching

Fig. 11. Performance of our decentralized algorithm over line topology.




(a) gain at each server by our algorithm (b) gain at each server by Most-Caching (c) comparison with Most-Caching

Fig. 12. Performance of our decentralized algorithm over hexagon topology.



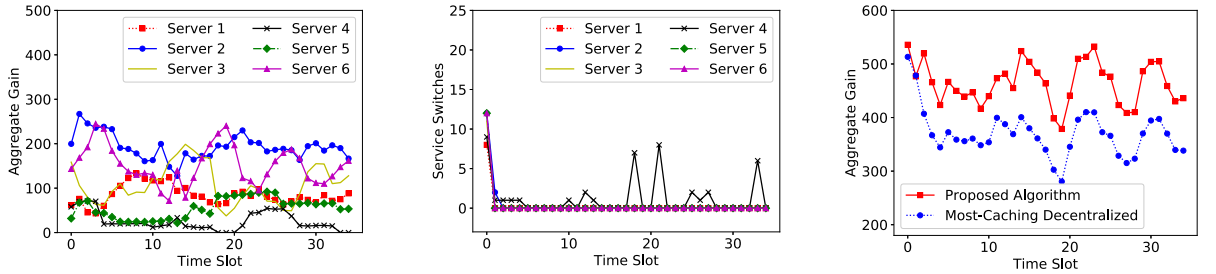

(a) gain at each server by our algorithm (b) gain at each server by Most-Caching (c) comparison with Most-Caching

Fig. 13. Performance of our decentralized algorithm over triangle topology.

Most-Caching in that: 1) it converges faster, for example, it takes less than 20 iterations to get an optimal service selection and resource allocation; and 2) it is much smoother as there is no sudden drop in aggregate gains when selecting a subset of services to host, which, on the other hand, can be observed in other baseline algorithms such as Most-Caching. Moreover, the proposed algorithm is effective that up to 110% (with the average 75%) performance enhancement can be achieved as compared with Most-Caching.

To investigate the performance of our cost-aware online algorithm in real-world setting, we adopt trace-driven simulations where the dataset consists of taxicab traces from [46] and wireless trace from [52]. More specifically, from taxicab we extract 36 mobile users and their location updates every 100 seconds, and assign them to 6 Voronoi cells that are at least 5 km apart. The wireless trace is used to generate requests, which contains packet inter-arrival times generated by 5 applications from 36 wireless devices. As in [20], we associate each device with a user in taxicab, and each application with a service. To have enough services in system, we further divide each application into 10 different services so that we have 50 services in total. Meanwhile, to simulate the task arrival process, we stretch time axis in each trace by 60 (so a second in trace now becomes a minute). The length of each time slot is set as $\Delta T = 100$ sec, and the time for a service switch (i.e., the time to load a new service from cloud or nearby servers) as $\Delta T = 20$ sec.

Fig. 14 gives performance of our decentralized cost-aware algorithm in this trace-driven simulation. We can see that due to user mobility and non-stationary workload arrivals, the system is not stable and that gains achieved at each server fluctuate




(a) gain at each server (b) service switches at each server (c) comparison with Top-Rate-Caching

Fig. 14. Performance of our decentralized algorithm in trace-driven simulation.

over time. However, we are still able to maintain low service switches at each server while at the same time retain high system performance, i.e., up to 45% gain enhancement can be expected by our algorithm as compared with Most-Caching. These results indicate that in practice, our mechanism can be adopted by the MEC service providers to improve their network performance.

## VI. Related Work

The issues of service placement, resource allocation and task offloading/scheduling in MEC have been extensively studied for the past few years. Below we introduce some representative work on these topics.

Early work on service placement and request scheduling mainly focus on offline solutions. [47] [23] propose near-optimal algorithms with a constant approximation factor to address the joint service placement and request scheduling problem. [18] considers data-intensive applications, in particular, content caching, but ignores constraints from other resources such as computation and communication. To address this issue, [47] and [23] propose to take into account multi-dimensional constraints when they try to maximize the number of satisfied requests by edge servers. In [9], the authors consider latency requirements on different services, and develop a set of uncoordinated strategies. [57] models the collaborative service placement problem as a matching problem and propose an efficient decentralized algorithm based on matching theory. Their goal is to minimize traffic load directed to cloud. Chen et al. [14] propose a parallel Gibbs-sampling based decentralized solution, taking into account service heterogeneity, spatial demand coupling and decentralized coordination.

Most recent work on service placement and request scheduling are primarily about online solutions. [38] [5] assume predictable system dynamics such as user mobility and propose online service placement schemes. However, in practice these assumptions are too strong. Xue et al. [56] model the considered task as a queue control/stabilization problem, and adopt a Lyapunov optimization approach. [20] proposes a two-time-scale framework to balance the cost of system reconfiguration and performance of serving requests, and develop a greedy algorithm based on set function optimization theory. Similarly, [42] [20] consider the operation cost of service migration and develop efficient mechanisms to trade off the cost and performance. Online learning schemes such as MAB-based optimization approaches and Thompson-sampling based learning algorithms [41] are also applied.

A significant amount of work is dedicated to resource allocation in MEC. Again, these work can be categorized into offline solutions and online solutions. Offline solutions generally assume availability of tasks and users, and then make radio or computation resource allocations according to different optimization objectives, i.e., to minimize task delay [19], energy consumption [50], or the weighted sum of them [15]. Online solutions, on the other hand, do not require prior knowledge and make resource allocation decisions dynamically as the system operates. For example, [28] [30] study the problem of minimizing energy consumption of all users while satisfying their delay requirements, through jointly optimizing radio resource allocation and task offloading. They propose a MDP-based approach. Authors in [35] [37] develop a Lyapunov-based online algorithm for radio and computation resource allocation, with the aim to minimize the long-term weighted sum of energy consumption of user devices and MEC servers. Machine learning techniques were recently applied. For example, [48] and [25] develop a model-free reinforcement learning based online task offloading approach to optimize the communication resource utilization. Regularization techniques [60] and MAB-based optimization theory [13] were also used to design efficient online algorithms for resource allocation.

Given that there has been a flurry of recent researches on MEC optimization, relatively few work has been dedicated to the problem of joint resource allocation, service caching and computation offloading in a *multi-user multi-server* environment. Authors in [39] study the joint communication, computation, caching and control in Big Data MEC systems, where the block successive upper bound minimization method is applied to solve the formulated problem. Apostolopoulos et al. [8] consider users' risk-seeking behavior when offloading their tasks, and use the principles of Prospect Theory and Tragedy of the Commons to formulate the problem. They propose a low-complexity algorithm based on a non-cooperative game formulation. Zhou et al. [59] investigate the problem of joint optimization of computation offloading and service caching in MEC-based smart grid, and propose CCORAM method, which comprises of a gradient descent allocation algorithm and a

game-based computing strategy. However, their approach cannot be applied in a distributed environment. Li et al. [31] aim at minimizing the energy consumption at UEs, through controlling power management and computing resources at UEs, in addition to task offloading decisions. They propose a two-stage heuristic based on genetic algorithms. Again, their solution is centralized. In [20], authors study the problem of maximizing the satisfied requests for a MEC-based network through jointly optimizing request routing and service placement, and propose a polynomial-time algorithm based on set function optimization theory. However, they do not consider the communication models in their system. Likewise, Ren et al. [49] study the problem of optimally offloading tasks between different Edge Service Providers (ESPs) and within each ESP, and propose an efficient two-layer offloading scheme to maximize the revenue of ESPs. Shen et al. [58] investigate the distributed computation offloading problem with delay constraints, and formulate the problem as a delay-constrained long-term stochastic optimization problem without prior knowledge. They propose TODG, a distributed online algorithm, to address the joint optimization problem.

Our work differs from the above in that: 1) we adopt a new QoS requirement called *offloading rate* requirement, while most existing work use delay constraints; 2) we consider a generic MEC-based network where each MEC server can collaborate with neighboring servers, whereas in some work an edge-cloud architecture is adopted and there's no collaboration among MEC nodes; 3) we consider all the four layers of communication, networking, computing and storage, and provide a unified framework for the joint optimization task; on the other hand, only 2 or 3 of the four layers are involved in most work; 4) we propose a novel concept called resource efficiency for service selection, and adopt a nested two-block Gauss-Seidel (GS) method to address the joint optimization problem; moreover, we devise a fully distributed algorithm based on the centralized solution, which at the same time guarantees that only limited information (instead of the raw task profile) needs to be exchanged between neighboring nodes to achieve a network-wide optimum. This provides us a sense of security/privacy protection; 5) We incorporate the cost of service switches from a real MEC system into the online algorithm design, which makes our mechanism more practical.

## VII. Conclusion

We study the problem of maximizing users' QoE in a MEC-based network through jointly optimizing service caching, resource allocation and task offloading. We adopt a multi-layer optimization approach and propose a novel two-stage algorithm based on approximation and decomposition theory. Distributed and online algorithms are also developed, with the special emphasis on limited/confidential information exchange between nodes and tradeoff between the cost of service switches and system performance. The efficiency of our algorithms are validated via numerical studies and trace-driven simulations.

## References


[1] Cloud ar/vr whitepaper. Accessed: Apr. 26, 2019. [Online]. Available: https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/

[2] Knapsack problem. Accessed: Apr. 8, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Knapsack_problem

[3] Open edge computing. Accessed: Apr. 23, 2023. [Online]. Available: http://openedgecomputing.org

[4] Openfog. Accessed: Apr. 23, 2023. [Online]. Available: https://opcfoundation.org/markets-collaboration/openfog/

[5] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5302–5316, Jun. 2018.

[6] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4004–4022, Mar. 2020.

[7] K. M. Anstreicher, "On convex relaxations for quadratically constrained quadratic programming," *Math. Program.*, vol. 136, no. 2, pp. 233–251, 2012.

[8] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1405–1418, Jun. 2020.

[9] O. Ascigil, T. K. Phan, A. G. Tasiopoulos, V. Sourlas, I. Psaras, and G. Pavlou, "On uncoordinated service placement in edge-clouds," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci.*, 2017, pp. 41–48.

[10] G. Banjac and P. J. Goulart, "A novel approach for solving convex problems with cardinality constraints," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13182–13187, 2017.

[11] M. Borza and A. S. Rambely, "A linearization to the sum of linear ratios programming problem," *Mathematics*, vol. 9 no. 9, 2021, Art. no. 1004.

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

[13] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4233–4248, Dec. 2021.

[14] L. Chen and J. Xu, "Collaborative service caching for edge computing in dense small cell networks," 2017, *arXiv: 1709.08662*.

[15] M.-H. Chen, M. Dong, and B. Liang, "Joint offloading decision and resource allocation for mobile cloud with computing access point," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 3516–3520.

[16] 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.

[17] W. Chu, P. Yu, Z. Yu, J. C. Lui, and Y. Lin, "Online optimal service selection, resource allocation and task offloading for multi-access edge computing: A utility-based approach," *IEEE Trans. Mobile Comput.*, early access, Feb. 18, 2022, doi: 10.1109/TMC.2022.3152493.

[18] M. Dehghan et al., "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1635–1648, Jun. 2016.

[19] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

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

[21] A. F. Florian Scheck and A. Freyberg, "5G: A key requirement for autonomous driving–really?," Jan. 2020. [Online]. Available: https://www.kearney.com/communications-media-technology/article/-/insights/5g-a-key-requirement-for-autonomous-driving-really-

[22] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear Gauss–Seidel method under convex constraints," *Operations Res. Lett.*, vol. 26, no. 3, pp. 127–136, 2000.

[23] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 365–375.

[24] 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.

[25] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[26] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Fourth Quarter 2015.

[27] Z. Jiang, B. Wu, and Q. Hu, "Optimality conditions for cardinality-constrained programs and a SCA method," 2019, *arXiv: 1906.00504*.
[28] M. Kamoun, W. Labidi, and M. Sarkiss, "Joint resource allocation and offloading strategies in cloud enabled cellular networks," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 5529–5534.
[29] X. Kang and Y. Wu, "Incentive mechanism design for heterogeneous peer-to-peer networks: A Stackelberg game approach," *IEEE Trans. Mobile Comput.*, vol. 14, no. 5, pp. 1018–1030, May 2015.
[30] W. Labidi, M. Sarkiss, and M. Kamoun, "Joint multi-user resource scheduling and computation offloading in small cell networks," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput., Netw. Commun.*, 2015, pp. 794–801.
[31] H. Li, H. Xu, C. Zhou, X. Lü, and Z. Han, "Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10214–10226, Sep. 2020.
[32] J. Linderoth, "A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs," *Math. Program.*, vol. 103, no. 2, pp. 251–282, 2005.
[33] Z.-Q. Luo and P. Tseng, "Error bounds and convergence analysis of feasible descent methods: A general approach," *Ann. Operations Res.*, vol. 46, no. 1, pp. 157–178, 1993.
[34] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tut.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.
[35] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
[36] Y. Mao, J. Zhang, S. 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.
[37] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.
[38] A. Nadembega, A. S. Hafid, and R. Brisebois, "Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.
[39] A. Ndikumana et al., "Joint communication, computation, caching, and control in Big Data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.
[40] S. Nunna et al., "Enabling real-time context-aware collaboration through 5G and mobile edge computing," in *Proc. 12th Int. Conf. Inf. Technol.-New Gener.*, 2015, pp. 601–605.
[41] 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.
[42] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[43] M. Patel et al., "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-Edge Comput. Ind. Initiative*, vol. 29, pp. 854–864, 2014.
[44] M. Patriksson, "Decomposition methods for differentiable optimization problems over Cartesian product sets," *Comput. Optim. Appl.*, vol. 9, no. 1, pp. 5–42, 1998.
[45] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Commun. Surveys Tut.*, vol. 16, no. 1, pp. 414–454, First Quarter 2013.
[46] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009–02-24)," Feb. 2009. [Online]. Available: https://crawdad.org/epfl/mobility/20090224
[47] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 10–18.
[48] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.
[49] J. Ren et al., "An efficient two-layer task offloading scheme for MEC system with multiple services providers," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1519–1528.
[50] S. Sardellitti, S. Barbarossa, and G. Scutari, "Distributed mobile cloud computing: Joint optimization of radio and computational resources," in *Proc. IEEE Globecom Workshops*, 2014, pp. 1505–1510.
[51] C. W. Tan, D. P. Palomar, and M. Chiang, "Distributed optimization of coupled systems with applications to network utility maximization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. Proc.*, 2006, pp. V–V.
[52] A. S. Uluagac, "CRAWDAD dataset gatech/fingerprinting (v. 2014–06-09)," Jun. 2014. [Online]. Available: https://crawdad.org/gatech/fingerprinting/20140609/isolatedtestbed
[53] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Leveraging cloudlets for immersive collaborative applications," *IEEE Pervasive Comput.*, vol. 12, no. 4, pp. 30–38, Fourth Quarter 2013.
[54] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
[55] W. Wu, J. C. Lui, and R. T. Ma, "On incentivizing upload capacity in P2P-VoD systems: Design, analysis and evaluation," *Comput. Netw.*, vol. 57, no. 7, pp. 1674–1688, 2013.
[56] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.
[57] 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.
[58] S. Yue et al., "TODG: Distributed task offloading with delay guarantees for edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1650–1665, Jul. 2021.
[59] H. Zhou, Z. Zhang, D. Li, and Z. Su, "Joint optimization of computing offloading and service caching in edge computing-based smart grid," *IEEE Trans. Cloud Comput.*, Apr. 12, 2022, doi: 10.1109/TCC.2022.3163750.
[60] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, 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. 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. From 2011-2012 he worked as a visiting researcher with Microsoft Research Asia, Beijing. From 2013 he was with the School of Computer Science and Technology, Northwestern Polytechnical University. His research interests include internet measurement and modeling, traffic analysis and performance evaluation.

**Xinming Jia** received the BE degree from Northwestern Polytechnical University, Xi'an, China, in 2021. He is currently working toward the MS degree with the School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an. His research interests include resource management and incentive mechanisms design for edge computing systems.

**Zhiwen Yu** (Senior Member, IEEE) 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 UCLA. He is currently a professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong. His current research interests include communication networks, network/system security (e.g., cloud security, mobile security, etc.), network economics, network sciences (e.g., online social networks, information spreading, etc.), cloud computing, large-scale distributed systems and performance evaluation theory. He serves in the editorial board of *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*. He was the chairman of the CSE Department from 2005 to 2011. He received various departmental teaching awards and the CUHK vice-chancellor's Exemplary Teaching Award. He is also a corecipient of the 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, 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 of the School of Computer Science, Northwestern Polytechnical University. His research interests include data storage and software engineering.