# Pricing-Driven Service Caching and Task Offloading in Mobile Edge Computing

Jia Yan, *Graduate Student Member, IEEE*, Suzhi Bi, *Senior Member, IEEE*, Lingjie Duan, *Senior Member, IEEE*, and Ying-Jun Angela Zhang, *Fellow, IEEE*

***Abstract*— Provided with mobile edge computing (MEC) services, wireless devices (WDs) no longer have to experience long latency in running their desired programs locally, but can pay to offload computation tasks to the edge server. Given its limited storage space, it is important for the edge server at the base station (BS) to determine which service programs to cache by meeting and guiding WDs' offloading decisions. In this article, we propose an MEC service pricing scheme to coordinate with the service caching decisions and control WDs' task offloading behavior in a cellular network. We propose a two-stage dynamic game of incomplete information to model and analyze the two-stage interaction between the BS and multiple associated WDs. Specifically, in Stage I, the BS determines the MEC service caching and announces the service program prices to the WDs, with the objective to maximize its expected profit under both storage and computation resource constraints. In Stage II, given the prices of different service programs, each WD selfishly decides its offloading decision to minimize individual service delay and cost, without knowing the other WDs' desired program types or local execution delays. Despite the lack of WD's information and the coupling of all the WDs' offloading decisions, we derive the optimal threshold-based offloading policy that can be easily adopted by the WDs in Stage II at the Bayesian equilibrium. In particular, a WD is more likely to offload when there are fewer WDs competing for the edge server's computation resource, or when it perceives a good channel condition or low MEC service price. Then, by predicting the WDs' offloading equilibrium, we jointly optimize the BS' pricing and service caching in Stage I via a low-complexity algorithm. In particular, we first study the differentiated pricing scheme and prove that the same price should be charged to the cached programs of the same workload. Motivated by this analysis, we further propose a low-complexity uniform pricing heuristics.**

Manuscript received August 11, 2020; revised December 23, 2020; accepted February 4, 2021. Date of publication February 24, 2021; date of current version July 12, 2021. This work was supported in part by the National Natural Science Foundation of China under Project 61871271, in part by the General Research Fund established by the Research Grants Council of Hong Kong under Project 14208017, in part by the Guangdong Province Pearl River Scholar Funding Scheme 2018 under Project 308/00003704, in part by the Key Project of Department of Education of Guangdong Province under Grant 2020ZDZX3050, and in part by the Foundation of Shenzhen City under Project JCYJ20170818101824392 and Project JCYJ20190808120415286. The associate editor coordinating the review of this article and approving it for publication was L. Zhao. *(Corresponding author: Suzhi Bi.)*

Jia Yan and Ying-Jun Angela Zhang are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: yj117@ie.cuhk.edu.hk; yjzhang@ie.cuhk.edu.hk).

Suzhi Bi is with the College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: bsz@szu.edu.cn).

Lingjie Duan is with the Engineering Systems and Design Pillar, Singapore University of Technology and Design, Singapore 487372 (e-mail: lingjie_duan@sutd.edu.sg).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2021.3059692.

Digital Object Identifier 10.1109/TWC.2021.3059692

***Index Terms*— Mobile edge computing, service caching and pricing, computation offloading, dynamic game under incomplete information.**

## I. Introduction

VARIETIES of modern mobile applications, such as face recognition, online gaming and augmented reality, have recently emerged into our daily life. Wireless devices (WDs) equipped with low-performance computation units often experience long latency to run these emerging computation-heavy applications. Alternatively, mobile edge computing (MEC) is a promising solution to provide high-performance computing for the WDs [1], [2]. Instead of forwarding tasks to the remote data center as traditional mobile cloud computing does, the WDs are able to offload their tasks to nearby edge servers, which efficiently reduces the high overhead and long backhaul latency. The global edge computing market is expected to reach $28.07 billion by 2027.[1] For example, Amazon offers many MEC services, such as AWS IoT Greengrass,[2] where users are charged based on the individual services they need.

In cellular networks, WDs can opportunistically offload their tasks to the edge server according to time-varying channel conditions and dynamic resource availability at the edge server. Most of the existing work on opportunistic computation offloading [3]–[7] assumes that the edge server has stored all the service programs. In practice, however, the service program acquiring process is time-consuming even during off-peak traffic hours. Compared with the task execution time at a millisecond level, the installation and loading time of a program takes tens of seconds for some common applications [8]. As such, the low latency requirement does not allow the edge server to fetch remotely from the program provider every time an MEC service is required. The edge server needs to pre-cache popular programs before repeatedly requested by WDs' offloaded tasks. Due to the limited caching storage capacity, the edge server needs to be selective and can only cache a subset of requested service programs before serving the WDs. For the uncached service programs, the edge server is unable to provide the real-time computation services to the corresponding WDs' online applications. This is referred to as *service caching* [9]–[14].

In this article, we consider an MEC system with a base station (BS) and multiple associated WDs. The MEC server is

[1] https://meticulousblog.org/top-10-companies-in-edge-computing-market/

[2] https://aws.amazon.com/greengrass/

co-located with the BS. Specific service programs are required to compute the tasks for the WDs. For instance, a human face recognition service program at the edge server can be repetitively called to process individual pictures of different WDs. Based on its storage and computational capacity, the BS determines which service programs to cache and what prices to charge for the MEC service provided to the WDs. Based on the BS' service caching and pricing decisions as well as the competition from the other WDs, each WD decides whether to compute its task locally or at the edge server.

Making the optimal service caching, pricing, and offloading decisions is a challenging task. Intuitively, the BS needs to know the WDs' offloading decisions, so that it can cache the service programs that are most popularly requested. Likewise, each WD's offloading decision is affected by not only the service caching and pricing of the BS, but also the offloading decisions of the other WDs due to the sharing of MEC server resources. However, in practice the WDs are unwilling to reveal their private information about their local computing capabilities, task offloading delays and requested service program types. As such, their offloading decisions cannot be accurately inferred by the BS and the other WDs.

To address the above problem, we model the practical interaction between the BS and WDs as a two-stage dynamic game of incomplete information (a.k.a Stackelberg game under incomplete information) [15], [16], where only the random distributions of each WD's characteristics, including requested service program type, local execution delay and offloading time, are known to the others. Specifically, in Stage I, the BS determines the service caching decisions and sets service prices for tasks requiring different programs to maximize its expected profit subject to the caching space and computation constraints. A higher price for a certain service program decreases the offloading willingness of the tasks requiring the program, but spares more computation resource to serve other types of tasks. The prices for different service programs are highly related to the total caching space and computing power at the BS, the size of each service program, the popularity of each service program, and the WDs' willingness to offload their tasks under a certain price (which is not known precisely due to the lack of complete information). In this regard, we are interested in answering the first key question: *What is the BS' optimal pricing and service caching strategy that maximizes its expected profit under incomplete information about the WDs?*

In Stage II, based on the given prices, the WDs compete for the computation resource at the BS and make task offloading decisions individually to minimize their own costs. Note that the WDs' optimal offloading decisions are coupled due to the sharing of the limited resources at the edge server. Under such negative externalities, a WD may choose not to offload its task to the edge server if it predicts that many other WDs are going to offload, leaving the edge server little computing power to execute its task. Accordingly, the second key question is raised: *How should each WD decide its offloading decision and how different WDs (with the same or different programs) affect each other's decision-making under incomplete information?*

The main contributions in this article are concluded as follows:

- *Two-stage dynamic game (a.k.a Stackelberg game) of incomplete information for managing service caching and task offloading:* To our best knowledge, this is the first work that studies two-stage dynamic game of incomplete information to jointly coordinate edge service caching and guide computation task offloading in the MEC systems. Besides selectively caching programs to admit the target WDs, we employ pricing [17], [18] as another degree of control to mitigate WDs' competition for limited computation resource and maximize the BS' profit. The BS can encourage offloading of a certain type of tasks by decreasing the price of the corresponding program or increasing the other programs' prices.
- *Bayesian equilibrium of WDs' offloading decisions:* For any given prices in Stage I, we analyze the WDs' optimal offloading decisions by considering their mutual competition and incomplete information. We show that each WD will follow a threshold-based task offloading policy at the Bayesian equilibrium, which is simple to implement in practice. More specifically, the threshold is a function of the programs' prices, the BS' CPU computation frequency, and the statistic characteristics of WDs' private information.
- *Optimal strategy of BS' pricing and service caching:* Based on the analysis of the Bayesian equilibrium in Stage II, we first derive the differentiated pricing scheme for Stage I. Differentiated pricing assumes that the BS sets different prices for the service programs. Based on the analysis of the optimal prices for the cached service programs, an efficient optimization algorithm is proposed to obtain the optimal prices and service caching decisions. Besides, in a special case where the valuation of each WD's personalized information is uniformly distributed, we obtain more engineering insights on the optimal pricing. More interestingly, we further show that the prices of two service programs are equal when they require the same computational workload, which motivates a low-complexity uniform pricing heuristics. Specifically, we suppose that the BS charges all the service programs with the same price and propose a low-complexity algorithm to jointly optimize the price and service caching decisions.

Compared to the existing studies, the distinct technical challenges are threefold. First, given the prices in Stage I, the WDs form a Bayesian subgame in Stage II under the incomplete information of WDs' local computing capacities, task offloading delays and requested service program types. Due to their mutual competition for the limited edge computing resources, each WD should infer the other WDs' strategic decisions for his own decision making under such incomplete information scenario. Therefore, it is technically challenging to find the Bayesian subgame equilibrium and analyze the structural properties of the obtained equilibrium in Stage II. Second, based on the derived Bayesian equilibrium in Stage II, finding the optimal strategy of BS' pricing and service caching in Stage I is a mixed integer stochastic optimization problem.

The problem is challenging due to the combinatorial nature of the service caching decisions and the strong coupling with the service prices. The third challenge is to further develop a reduced-complexity pricing scheme in Stage I for fast deployment in practical heterogeneous networks with a large number of service programs.

The rest of the article is organized as follows. In Section II, we introduce the system model. Section III formulates the two-stage dynamic game of incomplete information. We analyze the Bayesian subgame among the WDs in Stage II in Section IV. The general differentiated pricing scheme in Stage I is studied in Section V. We investigate the low-complexity uniform pricing heuristics in Section VI. Some model and result extensions are analyzed in Section VII. In Section VIII, numerical results are described. Finally, we conclude the article in Section IX.

### A. Related Work

Existing work has extensively studied opportunistic computation offloading, which is often jointly optimized with system computation and communication resource allocation [3]–[7]. Only recently has service caching started to attract research interests [9]–[14]. For a single-server MEC system, [9] proposed an online algorithm to dynamically schedule the cached services without the knowledge of task arrival patterns. For a multi-server MEC system, [10] studied the joint service caching and request scheduling problem. The problem of minimizing served traffic load was considered in [11]. [12] posed the service caching problem as a combinatorial bandit learning problem. In [13], each WD can offload its task to either the remote cloud center or a nearby edge node that has cached the required service program. Based on this, a joint optimization of service caching and task offloading was studied therein. Notice that the above work [9]–[13] has assumed that all tasks are computed at the edge server or/and remote cloud, neglecting the benefits of opportunistic computation offloading in MEC. For example, WDs may choose to compute locally when they perceive poor channel conditions. Very recently, [14] considered the joint optimization of service caching placement, task offloading, and resource allocation in a sequential task graph.

The above work has optimized the service caching, task offloading and resource allocation in a centralized manner. In terms of decentralized operation, previous work has proposed Stackelberg games [19], [20], priority pricing [21], multi-round resource trading [22] in MEC systems. Specifically, [19] considered a Stackelberg game, where the edge server acts as the leader and sets prices to maximize its revenue with computation capacity constraint. The WDs are the followers and locally make offloading decisions to minimize their own costs for given prices. [20] studied optimal pricing and edge node selection by adopting a Stackelberg game. [21] further proposed a priority pricing scheme, where users are served first for a higher price. The authors in [22] designed an online multi-round auction mechanism for profit maximization.

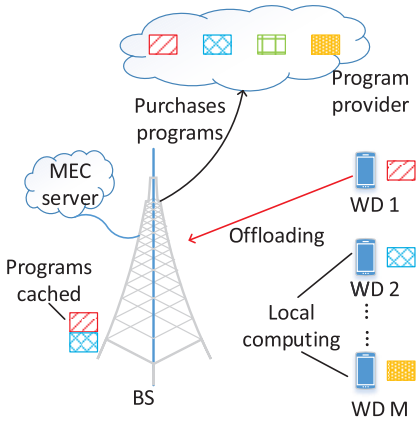



Fig. 1. System model of BS provision of MEC services to $M$ WDs who can choose to offload their desired programs' tasks to the edge server (i.e., BS) or compute locally. Given its limited caching storage, the BS can only serve those WDs if it caches such desired programs from the program provider beforehand. The BS cannot cache all potential programs from the program provider.

[19]–[22] have assumed that all programs are cached at the BS and none of these studies have taken service caching into account. In this article, we endeavor to design a Stackelberg game in MEC to coordinate the service caching and pricing decisions at the BS and the offloading behavior of the WDs. Besides, the existing work (e.g., [19], [20], [22]) has assumed that each WD's private information is known to all. In contrast, we take into account the information uncertainty when designing and analyzing the two-stage Stackelberg game.

## II. System Model

As shown in Fig. 1, we consider a multi-user MEC system with $M$ single-antenna active WDs, denoted by a set $\mathcal{M} = \{1, 2, \ldots, M\}$, and one single-antenna BS. MEC server is located at the BS to share the infrastructure such as stable power supply. The BS provides MEC services to the WDs with its limited computation resource and storage capacity.

Suppose that each WD has a computationally intensive task to compute. The computation of each task requires a service program, e.g., human face recognition program. We refer to a task as a type-$j$ task if it is processed by the service program $j, j \in \mathcal{N} = \{1, 2, \ldots, N\}$, where $N$ is the total number of service programs. We define a binary indicator $u_{i,j}$ such that $u_{i,j} = 1$ when the task of WD $i$ is of type $j$, and 0 otherwise. Accordingly, $\sum_j u_{i,j} = 1, \forall i \in \mathcal{M}$, implying that a WD cannot run two programs at the same time. Besides, we denote the type of the WD $i$'s task as $\varphi_i \in \{1, \ldots, N\}$. In particular, $\varphi_i = j$ if $u_{i,j} = 1$.

Each WD needs to decide whether to compute its task locally or remotely at the BS. Define a binary indicator variable $a_i$ such that $a_i = 1$ when WD $i$ decides to offload its task for edge computing and $a_i = 0$ when WD $i$ decides to compute the task locally. A task can be served at the BS only if the BS has pre-cached the corresponding service program from the program provider. Take Fig. 1 for example. The BS has cached the required programs of WD 1 and WD 2. After comparing the costs of local computing and edge computing,

WD 1 decides to offload its task for edge computing, while WD 2 decides to compute locally despite the availability of its required service program at the BS. WD $M$, on the other hand, has no choice but to compute locally, because its required service program is not cached at the BS. To avoid trivial cases, we assume that the WDs have the service programs to run their own tasks locally. Otherwise, they will always offload the tasks to the edge server. In the following, we introduce the service caching, communication, and computation models in detail.

### A. BS' Service Caching Model

Suppose that the cost for the BS to acquire the $j$-th service program from the program provider is $r_j$. After obtaining the program data and configuration from the program provider, the edge server installs and caches the service programs (e.g., executable.EXE files). We use a binary indicator $x_j$ to denote the caching decision of the $j$-th program at the BS. Specifically, $x_j = 1$ tells that the $j$-th program is cached at the BS, and $x_j = 0$ otherwise. Given the limited storage space, the BS cannot cache all the potential programs. We model the caching capacity constraint at the BS as

$$\sum_{j=1}^{N} x_j c_j \leq C, \tag{1}$$

where $c_j$ is the size of the $j$-th generated program and $C$ is the caching space at the BS. Besides, the BS needs the input task data (e.g. individual photos for the human face recognition program) from the WDs to run the cached programs.

### B. WDs' Communication Model With BS

We assume that the WDs access the uplink spectrum through FDM or OFDM to avoid mutual interferences. Each WD is fairly allocated an orthogonal channel of identical bandwidth $W$.[3] Let $p_i$ denote the transmit power of WD $i$ when offloading its task to the BS. The wireless channel gain between WD $i$ and the BS is denoted as $h_i$. Besides, we assume additive white Gaussian noise (AWGN) with zero mean and identical variance $\sigma^2$ at all the receivers. The offloading data rate of the task from WD $i$ to the BS is

$$R_i^u = W \log_2 \left(1 + \frac{p_i h_i}{\sigma^2}\right). \tag{2}$$

Then, the transmission time of WD $i$ when offloading its task is expressed as

$$\tau_i^u = \frac{I_i}{R_i^u}, \tag{3}$$

where $I_i$ is the size (in bits) of the input data of WD $i$'s task. In this article, we suppose that different computation tasks under the same service program can have different data inputs and outputs. For example, when running the human face recognition application, WDs need to input their photos of different sizes and definitions and expect the program to return specific results.

[3]Note that there are mature ways such as spectrum management to control interference, which is out of the scope of this article.

Finally, we assume that the time spent on downloading the task computation result from the BS to the WD is negligible due to the strong transmit power of the BS and the relatively small output data size (as compared to the input data size). For instance, the human face recognition application outputs the person name with only a few bytes, which is much smaller than the size of the corresponding input photo (of several mega bytes).

### C. Computation Model at the BS and the WDs

If WD $i$ computes its task locally, i.e., $a_i = 0$, then the local execution time is

$$\tau_i^l = \frac{\sum_{j=1}^{N} u_{i,j} L_j}{f_i^l}, \tag{4}$$

where $f_i^l$ is the CPU computation frequency of WD $i$ and $L_j$ denotes the computational workload in CPU cycles to execute the type-$j$ task.

Alternatively, WD $i$ can choose to offload its task in MEC services, i.e., $a_i = 1$. Suppose that the edge server creates multiple virtual machines (VMs) to execute the offloaded tasks in parallel and each VM is assigned to handle one task. In practice, the edge server with powerful computation capability can actually assign an individual VM for each offloaded task, thus allowing parallel computation of multiple tasks [3-6]. Some popular techniques to support such parallel computation at the edge server include serverless computing [8] and container [23] technologies, where the computation resources at the edge server (including memory, CPU, disk, and networking resources) are partitioned into VMs that do not interfere with each other. For simplicity, we assume that the total computation resource at the edge server (i.e., the total CPU frequency $f^c$) is equally partitioned and allocated to the VMs. Accordingly, the task processing time at the edge server is

$$\tau_i^c(m) = \frac{\sum_{j=1}^{N} u_{i,j} L_j}{f^c/m}, \tag{5}$$

where $m$ is the number of WDs' tasks offloaded to the edge server for edge computing, i.e., $m = \sum_{i=1}^{M} a_i$. Notice that $m$ is a random variable depending on WDs' offloading decisions $a_i$. $\tau_i^c(m)$ is an increasing function of $m$. In Section VII, we will extend the investigation to a general case where the edge CPU frequency is proportionally allocated to the offloaded tasks according to their computation workloads.

## III. Two-Stage Dynamic Game Formulation Under Incomplete Information for MEC Service Provision

The BS and the WDs interact with each other in a two-stage dynamic game of incomplete information as shown in Fig. 2, where the BS is the leader and the WDs are the followers. The profit-seeking BS first determines the caching decisions and sets the program prices for task executions in Stage I. Then, the delay-sensitive WDs optimize their offloading decisions individually in Stage II based on the prices announced from the BS and the WDs' mutual competition to share the limited

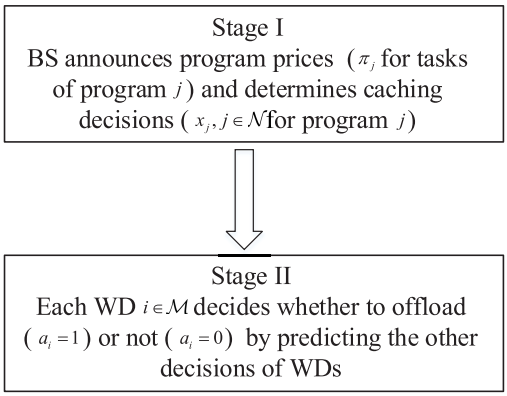



Fig. 2. Our proposed two-stage dynamic game for the interaction between the BS and the WDs.

computation resource at the BS. We will analyze this dynamic game by backward induction. In the following, we first detail the problem formulation in each stage.

### A. WDs' Problem Formulation in Stage II

Each WD aims to make an optimal offloading decision to minimize its total cost, defined as its task execution delay plus the payment to the BS. In particular, the task execution time of WD $i$, denoted by $T_i$, is

$$T_i(m) = (1 - a_i)\tau_i^l + a_i(\tau_i^u + \tau_i^c(m)). \tag{6}$$

Note that if $a_i = 0$, the total delay $T_i$ in (6) simply equals the local execution time $\tau_i^l$ in (4). Otherwise, $T_i$ consists of the data uploading time $\tau_i^u$ in (3) and the edge computing time $\tau_i^c$ in (5), which increases with $m$, the total number of tasks offloaded to the BS. If WD $i$ chooses to offload its task, i.e., $a_i = 1$, then it needs to pay the BS for the service. Suppose that the price the BS charges for program $j$'s task execution is $\pi_j$ per CPU cycle. Then, the total amount WD $i$ pays is $a_i \sum_{j=1}^{N} u_{i,j} L_j \pi_j$. As such, the total cost WD $i$ aims to minimize is

$$U_i(m) = T_i(m) + a_i \sum_{j=1}^{N} u_{i,j} L_j \pi_j. \tag{7}$$

In an ideal case where WD $i$ has the knowledge of $m$, it can simply compare $U_i(m|a_i = 1)$ and $U_i(m|a_i = 0)$ and choose the offloading decision that yields the smaller value of $U_i$. In practice, however, a WD is not able to infer the other WDs' offloading decisions, as their private information including local computing time $\tau_i^l$, offloading time $\tau_i^u$, computation workload $L_{\varphi_i}$ and program type $u_{i,j}$ is not revealed. As such, the exact value of $m$ is unknown. To address the issue, we will derive the Bayesian equilibrium to understand each WD's best response under incomplete information in Section IV.

### B. BS' Problem Formulation in Stage I

The BS aims to maximize its overall profit by optimizing the service caching $\mathbf{x} = \{x_j, j \in \mathcal{N}\}$ and program pricing $\boldsymbol{\pi} = \{\pi_j, j \in \mathcal{N}\}$. In particular, the profit is the difference between the payments received from the WDs and the cost of acquiring the programs from the program provider:

$$u_B = \sum_{i=1}^{M} a_i \sum_{j=1}^{N} x_j u_{i,j} L_j \pi_j - \sum_{j=1}^{N} x_j r_j. \tag{8}$$

The first term in the right hand side of (8) is the total payments received from the WDs. Here, the multiplicative factor $x_j$ corresponds to the fact that the BS can serve a type-$j$ task only when the $j$-th program is cached. The second term in (8) is the total program acquiring cost from the program provider.

Under the assumption of incomplete information, the BS does not know each WD's private information, including local execution time $\tau_i^l$, offloading time $\tau_i^u$, computation workload $L_{\varphi_i}$ and program type $u_{i,j}$. Instead, the BS only knows the distribution of each WD's characteristics. As such, the BS infers the offloading probabilities for the WDs and computes the expected profit as

$$U_B = \mathbb{E}[u_B] = \mathbb{E}\left[\sum_{i=1}^{M} a_i \sum_{j=1}^{N} x_j u_{i,j} L_j \pi_j - \sum_{j=1}^{N} x_j r_j\right], \tag{9}$$

where the expectation is taken over the distributions of service program types $u_{i,j}$ and the offloading decisions $a_i$ the WDs made in the Stage II game.

Mathematically, the optimization problem at the BS is formulated as

$$\begin{aligned} \text{(P1)} \quad & \max_{(\boldsymbol{\pi}, \mathbf{x})} \; U_B, \\ & \text{s.t.} \; \sum_{j=1}^{N} x_j c_j \le C, \\ & \quad x_j \in \{0, 1\}, \quad \forall j = 1, \ldots, N. \end{aligned} \tag{10}$$

In the following, we analyze the proposed two-stage dynamic game using backward induction. In Section IV, we first start with the Stage II game, when the service price $\boldsymbol{\pi}$ is fixed. In particular, we will analyze the Bayesian subgame among the WDs and obtain the equilibrium decision policy for all the WDs. Then in Section V and VI, we analyze the Stage I where the BS optimizes programs' prices $\boldsymbol{\pi}^*$ and caching decisions $\mathbf{x}^*$ to maximize its expected profit, by predicting the WDs' equilibrium offloading behavior.

## IV. Analysis of the WDs' Offloading Equilibrium in Stage II

By observing the prices announced by the BS in Stage I, the WDs determine the offloading decisions individually by estimating the other WDs' decisions, which leads to a Bayesian subgame in Stage II. An equilibrium is reached if no WD can improve its cost by changing its offloading strategy unilaterally. In this section, we manage to derive an optimal threshold-based offloading strategy for all the WDs at the Bayesian subgame equilibrium (as a stable outcome of the WDs' interactions) and analyze some interesting properties of the optimal threshold.

To derive the Bayesian subgame equilibrium under incomplete information, we first try to understand the properties of

the optimal offloading decisions in an ideal case where the private information of each WD is known by each other. That is, each WD exactly knows the number $m$ of tasks offloaded to the BS by calculating the offloading decisions on other WDs' behalf.

*Lemma 4.1 (Complete Information Scenario):* The optimal offloading decision of WD $i$ with type-$\varphi_i$ task is given by

$$a_i^*(\pi_{\varphi_i}) = \begin{cases} 1, & \pi_{\varphi_i} \leq \theta_i - \dfrac{m}{f^c}; \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where

$$\theta_i = \frac{\tau_i^l - \tau_i^u}{L_{\varphi_i}}. \tag{12}$$

Lemma 4.1 implies that WD $i$ will choose to offload if its total payment is smaller than the difference between the local execution time and the edge computing time, i.e., $L_{\varphi_i}\pi_{\varphi_i} \leq L_{\varphi_i}\theta_i - \frac{mL_{\varphi_i}}{f^c}$. Note that the WD $i$'s willingness-to-offload increases when its local computation time $\tau_i^l$ is long, the data uploading to the BS incurs short delay (small $\tau_i^u$), or there are a smaller number $m$ of WDs competing for the BS' computation resource $f^c$.

*Remark 4.1:* Our proposed pricing mechanisms can be directly applied to the scenario considering WDs' energy consumptions of local computing and task offloading. Suppose that the energy consumption for local computing of WD $i$ is $e_i^l = \kappa L_{\varphi_i}(f_i^l)^2$, where $\kappa$ is the effective switched capacitance parameter related to the chip architecture. Besides, we assume that the energy consumed on task offloading from WD $i$ to the BS is $e_i^u = p_i\frac{I_i}{R_i^u}$. Suppose that the goal of each WD is to optimize its offloading decision to minimize its task execution delay and energy consumption plus the payment to the BS. In this case, the integrated private information of WD $i$ is $\theta_i = \frac{\tau_i^l + e_i^l - \tau_i^u - e_i^u}{L_{\varphi_i}}$.

Now we turn to the incomplete information scenario where neither $m$ nor $\theta_i$ is publicly known. Instead, $\theta_i$ appears random to other WDs and the BS. We assume that $\theta_i$'s are independent and identically distributed with probability density function (PDF) $f(\cdot)$ and cumulative distribution function (CDF) $F(\cdot)$. The distribution function is a common prior knowledge to all the WDs and the BS.

*Remark 4.2:* For the private information $\theta_i$ of WD $i$, we have $\theta_i = \frac{\tau_i^l - \tau_i^u}{L_{\varphi_i}} = (\frac{L_{\varphi_i}}{f_i^l} - \frac{I_i}{R_i^u})/L_{\varphi_i}$. In practice, we consider a linear relation between the input data size $I_i$ and the computation workload $L_{\varphi_i}$ for the task of WD $i$, i.e., $I_i = \xi_i L_{\varphi_i}$ [3,4,6]. Accordingly, we have $\theta_i = \frac{1}{f_i^l} - \frac{\xi_i}{R_i^u}$, which depends on the local CPU frequency and the channel condition between the BS and WD $i$. *Because the variations of wireless channels are independent between different users, we assume that $\theta_i$ is independent across different WD $i$. Besides, we assume $\theta_i$ is identically distributed across the WDs for the simplicity in Section IV, V and VI. In Section VII, we extend the investigation to a general case where $\theta_i$'s are non-identically distributed.*

As in the wide literatures of pricing and mechanism designs [24], we suppose that the distribution of $\theta_i$ is regular as defined below.

*Assumption 1 (Regular Distribution):* $y(\theta) = \theta - \frac{1-F(\theta)}{f(\theta)}$ is an increasing function of continuous random variable $\theta$, where $F(\theta)$ and $f(\theta)$ are the CDF and PDF of $\theta$, respectively.

Note that many random distributions, such as uniform, normal, and exponential distributions, are indeed regular distributions. Likewise, we assume that WD $i$ does not know the other WDs' task types, so that the prices that the other WDs need to pay for offloading are unknown. In this regard, we assume that the program indicator $u_{i,j}$ for each WD $i$ appears random to other WDs with probability $q_j$. In particular, $q_j$ represents the $j$-th program's popularity. We suppose that all the WDs and BS can estimate the distribution of WD $i$'s integrated private information $\theta_i$ and the programs' popularity $q_j$ from historical data.

Then, the PDF of $\beta_i = \theta_i - \pi_{\varphi_i}$ is given by

$$g(\beta_i) = \sum_{j=1}^{N} q_j f(\beta_i + \pi_j), \tag{13}$$

and the corresponding CDF $G(\beta_i)$ is

$$\begin{aligned} G(\beta_i) &= \int_{-\infty}^{\beta_i} g(x)dx = \int_{-\infty}^{\beta_i} \sum_{j=1}^{N} q_j f(x + \pi_j)dx \\ &= \sum_{j=1}^{N} q_j F(\beta_i + \pi_j). \end{aligned} \tag{14}$$

Using the definition above and based on Lemma 4.1, we obtain the optimal offloading strategy for each WD at the Bayesian equilibrium under the incomplete information scenario in the following Theorem 1.

*Theorem 1 (Incomplete Information Scenario):* A WD $i$ with type-$\varphi_i$ task will offload its task to the BS if and only if

$$\pi_{\varphi_i} \leq \theta_i - \delta^*(\boldsymbol{\pi}). \tag{15}$$

Here, the equilibrium decision parameter $\delta^*(\boldsymbol{\pi})$ is the same for all the WDs and is the unique solution to

$$\Phi(\delta) := \delta - \frac{(M-1)(1-G(\delta)) + 1}{f^c} = 0. \tag{16}$$

Besides, $\delta^*(\boldsymbol{\pi})$ satisfies $\frac{1}{f^c} \leq \delta^*(\boldsymbol{\pi}) \leq \frac{M}{f^c}$.

*Proof:* Due to symmetry, we assume that all the WDs other than WD $i$ choose to offload their tasks to the edge server if and only if their valuations $\beta_k, k \neq i$, are larger than a decision parameter $\delta > 0$. By extending Lemma 4.1 to the incomplete information case, we know that when the following inequality holds, WD $i$ would prefer to offload its task for edge computing.

$$\theta_i - \pi_{\varphi_i} \geq \mathbb{E}\left[\frac{m' + 1}{f^c}\right],$$

where $m'$ follows a binomial distribution $B(M-1, 1-G(\delta))$ and represents the number of the WDs other than WD $i$ that prefer edge computing. Here, $\mathbb{E}\left[\frac{m'+1}{f^c}\right] = \frac{(M-1)(1-G(\delta))+1}{f^c}$. At the equilibrium, we have the common decision parameter $\delta$. That is,

$$\delta = \frac{\mathbb{E}_{m'}[m'] + 1}{f^c} = \frac{(M-1)(1-G(\delta)) + 1}{f^c}.$$

Next, we want to show that there exists a unique solution $\delta^*$ to (16). Since $G(\delta)$ in (14) is an increasing function with respect to $\delta$, $\Phi(\delta)$ is a monotonically increasing function with respect to $\delta$, i.e., $\frac{\partial G(\delta)}{\partial \delta} > 0$. When $\delta = \frac{1}{f^c}$, we have $\Phi(\delta) \leq 0$. When $\delta = \frac{M}{f^c}$, we have $\Phi(\delta) \geq 0$. Together with the result that $\Phi(\delta)$ is a monotonically increasing function, $\Phi(\delta) = 0$ has a unique solution $\delta^* \in [\frac{1}{f^c}, \frac{M}{f^c}]$. ■

According to Theorem 1, we can obtain $\delta^*$ through a bi-section search over $\delta^* \in [\frac{1}{f^c}, \frac{M}{f^c}]$ that satisfies $\Phi(\delta^*) = 0$. $\delta^*(\boldsymbol{\pi})$ can be viewed as the expectation of $\frac{m}{f^c}$ in (11) under incomplete information. Despite WDs' heterogeneity in local execution time, communication delay and program type, the proposed policy integrates all such personal information in a single parameter $\theta_i$. To decide whether to offload, a WD just needs to compare its private term $\theta_i$ with the decision threshold, which is defined as the equilibrium parameter $\delta^*$ plus price $\pi_{\varphi_i}$ for its desired program (i.e., $\theta_i \geq \pi_{\varphi_i} + \delta^*(\boldsymbol{\pi})$). In the following, we derive some interesting properties of decision threshold $\delta^* + \pi_{\varphi_i}$ for WD $i$.

*Proposition 4.1:* The WD $i$'s decision threshold $\delta^* + \pi_{\varphi_i}$ increases in its own program's price $\pi_{\varphi_i}$ and decreases in any other program's price $\pi_k, k \in \mathcal{N} \setminus \varphi_i$.

*Proof:* Please refer to Appendix A. ■

The Proposition 4.1 indicates that the BS can incentivize the WDs with type-$j$ tasks to offload by setting a lower price for the $j$-th program or a higher price for the other programs. Next, we further study the impacts of the edge server's CPU computation frequency $f^c$ and the total number of users $M$ on the offloading decision threshold $\delta^* + \pi_{\varphi_i}$.

*Proposition 4.2:* The WD $i$'s decision threshold $\delta^* + \pi_{\varphi_i}$ decreases in $f^c$, and increases in $M$.

*Proof:* Please refer to Appendix B. ■

It follows from Proposition 4.2 that WDs are more likely to offload when the BS has larger computational capability. Besides, as more WDs compete for the limited computation resource, each WD tends to offload with lower probability to avoid long computation latency at the BS.

By predicting the WDs' equilibrium strategies in Stage II through Theorem 1, we are ready to turn to Stage I in the following section.

## V. Differentiated Pricing to Cope With MEC Service Caching

The optimal decision in Stage I involves combinatorial optimization due to the binary service caching decisions $\mathbf{x}$ in (P1). Due to the variation in the programs' properties (i.e., the programs' popularity and workloads, the storage sizes of programs and the cost of acquiring different programs), the BS can set different prices for different programs to increase its profit (a.k.a differentiated pricing scheme). In this section, we first consider the general case of Problem (P1) where the BS is allowed to charge different prices for computing different types of tasks.

### A. BS' Profit Maximization in Problem (P1)

Based on the Bayesian equilibrium policy derived in Stage II, the BS can effectively predict the WDs' offloading behaviors by calculating the offloading probability of each program's tasks. Specifically, the offloading probability of type-$j$ tasks is equal to $1 - F(\delta^*(\boldsymbol{\pi}) + \pi_j)$. Then, Problem (P1) in Stage I is expressed as

$$\begin{aligned} \text{(P2)} \quad \max_{(\boldsymbol{\pi}, \mathbf{x})} \ & \sum_{j=1}^{N} (1 - F(\delta^*(\boldsymbol{\pi}) + \pi_j)) x_j q_j M \pi_j L_j - \sum_{j=1}^{N} x_j r_j, \\ \text{s.t.} \ & \sum_{j=1}^{N} x_j c_j \leq C, \\ & x_j \in \{0, 1\}, \quad \forall j = 1, \ldots, N. \end{aligned} \tag{17}$$

Problem (P2) is challenging due to the strong coupling between the caching decisions and each program's task price. To tackle this problem, we first suppose that the caching decisions are given and derive some important properties of the optimal prices in the following proposition, based on which we propose an efficient algorithm to optimize the prices.

*Proposition 5.1:* Suppose that a subset $\hat{\mathcal{N}} \subseteq \mathcal{N}$ of programs are cached in the BS. Then, the optimal program price $\pi_j^* > 0, \forall j \in \hat{\mathcal{N}}$ satisfies (18), shown at the bottom of the next page.

Here, $\delta^*(\boldsymbol{\pi})$ in Stage II is obtained in (16).

*Proof:* The expected profit of the BS is

$$U_B = \sum_{j \in \hat{\mathcal{N}}} (1 - F(\delta^* + \pi_j)) q_j M \pi_j L_j - \sum_{j \in \hat{\mathcal{N}}} r_j.$$

To find the optimal price of program $j$, we calculate the derivative of $U_B$ with respect to $\pi_j$ as

$$\begin{aligned} \frac{\partial U_B}{\partial \pi_j} = & -f(\delta^* + \pi_j)(1 + \frac{\partial \delta^*}{\partial \pi_j}) q_j M \pi_j L_j \\ & + [1 - F(\delta^* + \pi_j)] q_j M L_j \\ & + \sum_{k \neq j, k \in \hat{\mathcal{N}}} -f(\delta^* + \pi_k) \frac{\partial \delta^*}{\partial \pi_j} q_k M \pi_k L_k. \end{aligned}$$

According to the analysis in Stage II, we have

$$\frac{\partial \delta^*}{\partial \pi_j} = \frac{-(M-1) q_j f(\delta^* + \pi_j)}{f^c + (M-1) \sum_{j \in \hat{\mathcal{N}}} q_j f(\delta^* + \pi_j)}.$$

By equating $\frac{\partial U_B}{\partial \pi_j} = 0$, we have (19), shown at the bottom of the next page. By multiplying $[f^c + (M-1) \sum_{j \in \hat{\mathcal{N}}} q_j f(\delta^* + \pi_j)]$ in both sides of (19) for program $j$, we have (20), shown at the bottom of the next page, which yields $\omega_d(\pi_j) = 0$. ■

Accordingly, we propose an alternating algorithm that alternately optimizes the prices of the cached programs given any feasible $\hat{\mathcal{N}}$. Specifically, in the $t$-th iteration, the algorithm finds the optimal $\{\pi_j^{(t)}, j \in \hat{\mathcal{N}}\}$ according to Proposition 5.1 given $\boldsymbol{\pi}_{-j}^{(t-1)} = \{\pi_k^{(t-1)}, k \in \hat{\mathcal{N}} \setminus j\}$ and the corresponding $\delta^*(\boldsymbol{\pi}^{(t-1)})$. In the following, we show that there exists a unique $\{\pi_j^{(t)}, j \in \hat{\mathcal{N}}\}$ in each iteration $t$.

*Corollary 5.1:* Suppose that the other prices $\boldsymbol{\pi}_{-j}^{(t-1)}$ and $\delta^*(\boldsymbol{\pi}^{(t-1)})$ in the last iteration are given. There exists a unique solution $\pi_j^{(t)}, j \in \hat{\mathcal{N}}$, that satisfies $\omega_d(\pi_j^{(t)}) = 0$ in the $t$-th iteration.

*Proof:* If the distribution of $\theta_i$ is regular, then $\frac{[1 - F(\delta^*(\boldsymbol{\pi}^{(t-1)}) + \pi_j)]}{f(\delta^*(\boldsymbol{\pi}^{(t-1)}) + \pi_j)} - \pi_j$ is a decreasing function in $\pi_j$ given

$\delta^*(\boldsymbol{\pi}^{(t-1)})$. Besides, $(M-1)q_j[1-F(\delta^*(\boldsymbol{\pi}^{(t-1)})+\pi_j)]$ decreases in $\pi_j$. Therefore, $\omega_d(\pi_j)$ is a decreasing function of $\pi_j$ given $\boldsymbol{\pi}_{-j}^{(t-1)}$ and $\delta^*(\boldsymbol{\pi}^{(t-1)})$.

When $\pi_j = 0$, we have $\omega_d(\pi_j) > 0$. Besides, when $\pi_j \to +\infty$, $\omega_d(\pi_j) \to -\infty$. Hence, there exists a unique $\pi_j^* \in (0,+\infty)$ that satisfies $\omega_d(\pi_j^*) = 0$ given $\boldsymbol{\pi}_{-j}^{(t-1)}$ and $\delta^*(\boldsymbol{\pi}^{(t-1)})$. ∎

According to Corollary 5.1, given the prices $\boldsymbol{\pi}_{-j}^{(t-1)}$ and the equilibrium parameter $\delta^*(\boldsymbol{\pi}^{(t-1)})$ in (16) in Stage II, we can obtain the optimal price $\pi_j^{(t)}$ for the cached program $j, j \in \hat{\mathcal{N}}$, by an efficient bi-section search method over $\pi_j^{(t)} \in [0,\phi]$ that satisfies $\omega_d(\pi_j^{(t)}) = 0$ in the $t$-th iteration. Here, $\phi$ is a sufficiently large real number. We summarize the proposed alternating algorithm for differentiated pricing given any feasible $\hat{\mathcal{N}}$ in Algorithm 1.

Then, the remaining optimization of Problem (P2) is to find the optimal caching decisions $\mathbf{x}$. In general, when the total number of programs is moderate, we can enumerate all feasible service caching decisions $\mathbf{x}$ that satisfy the caching space constraints in (P2) and choose the best service caching decision that yields the maximal $U_B$. When the dimension of $\mathbf{x}$ is high, many meta-heuristic methods, such as Gibbs sampling [5] and particle swarm optimization [25], can be applied to effectively find the optimal solution. Take Gibbs sampling for example. By starting from an initial feasible solution $\mathbf{x}^{(0)}$, we update the service caching decision to $\mathbf{x}^{(t)}$ in the $t$-th sampling according to the probability distributions $\boldsymbol{\Lambda}(\mathbf{x}) = \{\Lambda(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}_{\mathbf{x}^{(t-1)}}\}$ with

$$\Lambda(\mathbf{x}) = \frac{\exp(-\mu/U_B(\mathbf{x}))}{\sum_{\mathbf{x}' \in \mathcal{X}_{\mathbf{x}^{(t-1)}}} \exp(-\mu/U_B(\mathbf{x}'))}. \tag{21}$$

In (21), $\mu$ is a temperature parameter and $\mathcal{X}_{\mathbf{x}^{(t-1)}}$ is the sampling set generated from $\mathbf{x}^{(t-1)}$. In particular, given $\mathbf{x}^{(t-1)}$, $\mathcal{X}_{\mathbf{x}^{(t-1)}}$ consists of binary vectors that differ from $\mathbf{x}^{(t-1)}$ in at most one position (i.e., changing at most one entry of $\mathbf{x}^{(t-1)}$ from 0 to 1 or 1 to 0). Moreover, the vectors in $\mathcal{X}_{\mathbf{x}^{(t-1)}}$ satisfy the caching storage constraint $\sum_{j=1}^{N} x_j c_j \le C$. According to (21), a service caching decision $\mathbf{x}$ that yields a larger expected profit $U_B$ is more likely to be picked. According to the proof in Section IV, [26], a Gibbs sampling algorithm achieves the optimal solution when it converges.

**Algorithm 1** Computation of Differentiated Pricing Given a Feasible $\hat{\mathcal{N}}$ for Problem (P2)

1: The BS initializes the prices as $\{\pi_j^{(0)}, j \in \hat{\mathcal{N}}\}$ and calculates the $\delta^*(\boldsymbol{\pi}^{(0)})$ in (16);
2: Set $t = 1$;
3: **repeat**
4: **for** Each cached program $j \in \hat{\mathcal{N}}$ **do**
5: Apply bi-section search method to find $\pi_j^{(t)}$ that satisfies $\omega_d(\pi_j^{(t)}) = 0$ in (18) with given $\boldsymbol{\pi}_{-j}^{(t-1)}$ and $\delta^*(\boldsymbol{\pi}^{(t-1)})$;
6: **end for**
7: Obtain $\{\pi_j^{(t)}, j \in \hat{\mathcal{N}}\}$ and calculate the corresponding $\delta^*(\boldsymbol{\pi}^{(t)})$ in (16);
8: Set $t = t+1$.
9: **until** $\{\pi_j^*, j \in \hat{\mathcal{N}}\}$ converges.

Though complicated, we manage to derive some interesting properties of the optimal prices in the differentiated pricing scheme under special cases to deliver more engineering insights.

First, we have the following corollary when $f^c \to \infty$.

*Corollary 5.2:* In the case where the BS has very large computational power, i.e., $f^c \to \infty$, we have

$$\pi_j^* = \frac{1-F(\pi_j^*)}{f(\pi_j^*)}, \forall j \in \hat{\mathcal{N}}.$$

$$\omega_d(\pi_j) := q_j M L_j \Bigg\{ (f^c + (M-1) \sum_{k \neq j, k \in \hat{\mathcal{N}}} q_k f(\delta^*(\boldsymbol{\pi}) + \pi_k)) \left[ \frac{[1-F(\delta^*(\boldsymbol{\pi}) + \pi_j)]}{f(\delta^*(\boldsymbol{\pi}) + \pi_j)} - \pi_j \right] + (M-1) q_j [1 - F(\delta^*(\boldsymbol{\pi}) + \pi_j)] \Bigg\} + (M-1) q_j \sum_{k \neq j, k \in \hat{\mathcal{N}}} f(\delta^*(\boldsymbol{\pi}) + \pi_k) q_k M \pi_k L_k = 0, \forall j \in \hat{\mathcal{N}}. \tag{18}$$

$$-(1 + \frac{-(M-1) q_j f(\delta^* + \pi_j)}{f^c + (M-1) \sum_{j \in \hat{\mathcal{N}}} q_j f(\delta^* + \pi_j)}) q_j M \pi_j L_j + \frac{[1 - F(\delta^* + \pi_j)]}{f(\delta^* + \pi_j)} q_j M L_j = \sum_{k \neq j, k \in \hat{\mathcal{N}}} f(\delta^* + \pi_k) \frac{-(M-1) q_j}{f^c + (M-1) \sum_{j \in \hat{\mathcal{N}}} q_j f(\delta^* + \pi_j)} q_k M \pi_k L_k, \forall j \in \hat{\mathcal{N}}. \tag{19}$$

$$\begin{aligned} & q_j M L_j \left[ [f^c + (M-1) \sum_{j \in \hat{\mathcal{N}}} q_j f(\delta^* + \pi_j)] \frac{[1 - F(\delta^* + \pi_j)]}{f(\delta^* + \pi_j)} - (f^c + (M-1) \sum_{k \neq j, k \in \hat{\mathcal{N}}} q_k f(\delta^* + \pi_k)) \pi_j \right] \\ & = q_j M L_j \left\{ (f^c + (M-1) \sum_{k \neq j, k \in \hat{\mathcal{N}}} q_k f(\delta^* + \pi_k)) \left[ \frac{[1 - F(\delta^* + \pi_j)]}{f(\delta^* + \pi_j)} - \pi_j \right] + (M-1) q_j [1 - F(\delta^* + \pi_j)] \right\} \\ & = -(M-1) q_j \sum_{k \neq j, k \in \hat{\mathcal{N}}} f(\delta^* + \pi_k) q_k M \pi_k L_k. \end{aligned} \tag{20}$$

*Proof:* When $f^c \to \infty$, we have $\delta^* = 0$ at the Stage II according to Theorem 1. Then, according to Proposition 5.1, we have $\pi_j^* = \frac{1-F(\pi_j^*)}{f(\pi_j^*)}, \forall j \in \hat{\mathcal{N}}$. ∎

From Corollary 5.2, the optimal prices only depend on the PDF and CDF of the WDs' valuation of $\theta_i$ when the $f^c$ is very large. It is intuitive as the WDs do not compete for the computation resource of the BS when it is abundant. Notice that even if $f^c$ goes to infinity, some WDs still choose not to offload due to the long communication delay (i.e., $\tau_i^u > \tau_i^l$ in (12)).

*Proposition 5.2:* If the task workloads of all the cached programs are the same ($L_j = L_k, \forall j,k \in \hat{\mathcal{N}}$), then the edge server should set the same price ($\pi_j^* = \pi^*, \forall j \in \hat{\mathcal{N}}$) for all the cached programs as the unique solution to

$$\pi - \frac{1 - F(\delta^*(\pi) + \pi)}{f(\delta^*(\pi) + \pi)} - \frac{(M-1)(\sum_{j\in\hat{\mathcal{N}}} q_j)(1 - F(\delta^*(\pi) + \pi))}{f^c} = 0. \quad (22)$$

*Proof:* Please refer to Appendix C. ∎

The above proposition indicates that if the cached programs have the same workload, then the optimal prices per CPU cycle are equal for these programs.

### B. Uniform Distribution of $\theta_i$ in (12)

To further obtain some engineering insights from our two-stage game analysis, we consider a special case where $\theta_i$ follows a uniform distribution within $[\underline{\theta}, \bar{\theta}]$. Without loss of generality, we assume that $\bar{\theta} > \frac{1}{f^c}$. Otherwise, none of the WDs will offload its task to the BS. Besides, we generally assume that $\underline{\theta} < 0$, implying that $\tau_i^l < \tau_i^u$ for some WDs facing bad channel conditions.

*Proposition 5.3:* Suppose that there are two programs $j$ and $k$ cached at the BS. Then, with uniform distribution of $\theta_i$, we have

- If $L_j = L_k$, then
$$\pi_j^* = \pi_k^* = \frac{\bar{\theta}}{2} - \frac{1}{2f^c}; \quad (23)$$
- If $L_j > L_k$, then $\pi_j^* < \pi_k^*$, implying that a higher unit price is charged to the program with the smaller workload.

*Proof:* Please refer to Appendix D. ∎

From the above proposition, we have the following observations:

- If the workloads of different cached programs are equal, the BS sets the same price per CPU cycle for the programs, which is in consistence with Proposition 5.2. For the obtained equal price, we have the following insights:
  - The BS tends to set a higher price when it has more computation power, i.e., a larger $f^c$. This is because a larger $f^c$ increases the WDs' willingness to offload. Thus, the BS can charge a higher price to obtain higher profit.
  - Increasing $\bar{\theta}$, the upper bound of the $\theta_i$, leads to a higher optimal price. It is because a larger average $\theta_i$ (i.e., a larger difference between local execution time $\tau_i^l$ and offloading transmission delay $\tau_i^u$ according to (12)) represents higher probability to offload for WD $i$. Accordingly, the BS can increase the price for higher profit.
- If the workloads are different, the program with the larger workload has a lower price. Notice that the programs' prices are charged for unit computation workload. Intuitively, the BS can increase its profit by setting a lower price for the program with the larger workload to encourage more workloads offloaded from the WDs.

Besides the relation between the optimal prices and the program workloads, Proposition 5.4 further discusses the impact of program popularity $q_j$ on the optimal prices.

*Proposition 5.4:* Suppose that the BS caches two programs $j$ and $k$. With uniform distribution of $\theta_i$, we have the following properties:

- If $L_j > L_k$, $\pi_j^*$ increases with popularity $q_j$ of program $j$, regardless of the value of $q_k$.
- If $L_j < L_k$, $\pi_j^*$ decreases with $q_j$ when $q_k \in (0, \frac{2f^c(\bar{\theta}-\underline{\theta})L_j}{(L_k-L_j)(M-1)})$. Otherwise, when $q_k \geq \frac{2f^c(\bar{\theta}-\underline{\theta})L_j}{(L_k-L_j)(M-1)}$, $\pi_j^*$ increases with $q_j$.

*Proof:* Please refer to Appendix E. ∎

From Proposition 5.4, we obtain the following insights:

- The optimal price $\pi_j$ increases with $q_j$ if the program $j$ has the larger workload. As the program $j$'s popularity increases, more WDs are interested in program $j$ with more demands. Thus, the BS can charge higher price for higher profit.
- If program $j$ has a smaller workload, the relation between $\pi_j$ and $q_j$ depends on the other larger workload's program popularity $q_k$. Specifically, when $q_k$ is small, i.e., the $k$-th program is not popular, the BS has more spare computation resource to serve type-$j$ tasks. Thus, when $q_j$ increases, the BS has the incentive to decrease the price $\pi_j$ to encourage the offloading of more type-$j$ tasks. On the other hand, when $q_k$ is large, the BS has the incentive to discourage the type-$j$ tasks' offloading by setting a higher price $\pi_j$ with the increase of $q_j$, so that it can spare more computation resource for the type-$k$ tasks.

## VI. Low-Complexity Uniform Pricing Heuristics to Cope With MEC Service Caching

Based on Proposition 5.2 in Section V, by approximating the term $\sum_{j\in\hat{\mathcal{N}}} q_j = 1$ in (22), we can decouple the optimization for the common price $\pi$ from that for the caching decision $\mathbf{x}$. In this case, we can obtain $\pi^*$ by solving (22). Having obtained $\pi^*$, the remaining problem for optimizing caching decision $\mathbf{x}$ is a standard Knapsack problem. This motivates a reduced-complexity pricing scheme in this section.

Specifically, we propose in this section, an efficient low-complexity algorithm to optimize the program prices and service caching decisions under a uniform pricing heuristics, where the BS sets the same unit price to all the programs, i.e., $\pi = \pi_1 = \ldots = \pi_N$. The identical pricing among all the programs simplifies our analysis of Problem (P1).

Based on Stage II's Bayesian equilibrium derived in Theorem 1, a task is offloaded with probability $1 - F(\delta^*(\pi) + \pi)$ regardless of its type due to the uniform pricing.[4] Then, we can rewrite Problem (P1) in Stage I as

$$\begin{aligned}(\text{P3}) \ \max_{(\pi,\mathbf{x})} \ & (1 - F(\delta^*(\pi) + \pi))\pi \sum_{j=1}^{N} L_j x_j q_j M - \sum_{j=1}^{N} x_j r_j, \\ \text{s.t.} \ & \sum_{j=1}^{N} x_j c_j \le C, \\ & x_j \in \{0,1\}, \quad \forall j = 1, \ldots, N. \end{aligned} \tag{24}$$

Note that the optimization for the price $\pi$ is decoupled from the caching decisions $\mathbf{x}$ in Problem (P3). Thus, we can separately optimize $\pi$ by maximizing the term $(1-F(\delta^*(\pi)+\pi))\pi$ in the objective function of Problem (P3). In particular, the following Proposition 6.1 shows that the optimal price $\pi^*$ can be efficiently obtained using bi-section search method.

**Algorithm 2** Computation of Uniform Pricing Heuristics and Service Caching for Problem (P3)

1: Set $\phi$ as a sufficiently large real number and $\varepsilon = 10^{-6}$;
2: $\pi^{UB} = \phi$, $\pi^{LB} = 0$;
3: **repeat**
4: Set $\pi = \frac{\pi^{UB}+\pi^{LB}}{2}$;
5: Calculate $\delta^*(\pi)$ in Stage II according to (16);
6: **if** $\omega_u(\pi) < 0$ **then**
7: $\pi^{LB} = \pi$;
8: **else**
9: $\pi^{UB} = \pi$;
10: **end if**
11: **until** $|\omega_u(\pi)| < \varepsilon$.
12: Obtain the optimal caching decisions $\mathbf{x}^*$ by solving knapsack problem via toolbox and using obtained $\pi^*$.

*Proposition 6.1:* The optimal uniform price $\pi^*$ of Problem (P3) is the unique solution to

$$\begin{aligned}\omega_u(\pi) := \pi - & \frac{[1 - F(\delta^*(\pi) + \pi)]}{f(\delta^*(\pi) + \pi)} \\ & - \frac{(M-1)\,[1 - F(\delta^*(\pi) + \pi)]}{f^c} = 0, \end{aligned} \tag{25}$$

where $\delta^*(\pi)$ is given in (16).

*Proof:* We first prove the optimality of the solution in (25). The derivative of $(1 - F(\delta^*(\pi) + \pi))\pi$ with respect to $\pi$ is

$$\begin{aligned}\frac{\partial[(1 - F(\delta^*(\pi) + \pi))\pi]}{\partial \pi} = & \left[\pi \left[-f(\delta^* + \pi)(\frac{\partial \delta^*}{\partial \pi} + 1)\right]\right. \\ & \left. + [1 - F(\delta^* + \pi)]\right]. \end{aligned} \tag{26}$$

[4]Here, we approximate the equilibrium parameter $\delta^*(\pi)$ in Stage II assuming that all the WDs are only informed of the uniform price. In this case, the offloading probability inferred by the BS is unrelated to the caching decisions. Actually, in the optimal differentiated pricing scheme, the Bayesian equilibrium in Stage II is based on the prices and caching decisions announced by the BS, where the WDs can treat the prices of the uncached programs as a sufficiently large price to guarantee zero offloading probability for the corresponding computation tasks.

According to the analysis in Stage II, we have

$$\frac{\partial \delta^*}{\partial \pi} = \frac{-(M-1)f(\delta^* + \pi)}{f^c + (M-1)f(\delta^* + \pi)}. \tag{27}$$

By substituting (27) into (26) and letting $\frac{\partial U_B}{\partial \pi} = 0$, we have

$$\pi^* - \frac{[1 - F(\delta^* + \pi^*)]}{f(\delta^* + \pi^*)} \frac{f^c + (M-1)f(\delta^* + \pi^*)}{f^c} = 0, \tag{28}$$

which yields $\omega_u(\pi^*) = 0$.

Next, by analyzing the property of $\omega_u(\pi)$, we demonstrate the existence and uniqueness of the optimal price. According to Theorem 1, we rewrite $\omega_u(\pi)$ as

$$\begin{aligned}\omega_u(\pi) = \delta^*(\pi) + \pi - & \frac{[1 - F(\delta^*(\pi) + \pi)]}{f(\delta^*(\pi) + \pi)} \\ & - \frac{2(M-1)\,[1 - F(\delta^*(\pi) + \pi)] + 1}{f^c}. \end{aligned} \tag{29}$$

Based on Assumption 1, $\delta^*(\pi) + \pi - \frac{[1-F(\delta^*(\pi)+\pi)]}{f(\delta^*(\pi)+\pi)}$ in (29) is an increasing function in $\delta^*(\pi) + \pi$. Besides, the last term $-\frac{2(M-1)[1-F(\delta^*(\pi)+\pi)]+1}{f^c}$ in (29) is an increasing function in $\delta^*(\pi) + \pi$. Therefore, all the terms in $\omega_u(\pi)$ increase with $\delta^*(\pi) + \pi$, implying that $\omega_u(\pi)$ increases in $\delta^*(\pi) + \pi$. Then, according to Proposition 4.1, $\delta^*(\pi) + \pi$ increases in $\pi$. Thus, we have $\omega_u(\pi)$ is an increasing function in $\pi$. Meanwhile, when $\pi = 0$, we have

$$\omega_u(\pi = 0) = -\frac{[1 - F(\delta^*)]}{f(\delta^*)} - \frac{(M-1)\,[1 - F(\delta^*)]}{f^c} < 0.$$

When $\pi \to +\infty$, we have $\omega_u(\pi) \to +\infty$. Together with the result that $\omega_u(\pi)$ is an increasing function, there must exist a unique $\pi^* \in [0, +\infty)$ that satisfies $\omega_u(\pi^*) = 0$. ■

With Proposition 6.1, given WDs' equilibrium response function $\delta^*(\pi)$, the optimal $\pi^*$ as the unique solution to (25) can be efficiently obtained via a bi-section search over the feasible price range $\pi^* \in [0, \phi]$, where $\phi$ is a sufficiently large real number. After obtaining the optimal uniform price in (P3), the remaining optimization of (P3) is a standard Knapsack problem, where off-the-shelf toolboxes can be applied to solve the optimum in pseudo-polynomial time. For example, we can adopt kp01$(\cdot)$ software package in MATLAB [19]. The details of the proposed algorithm are summarized in Algorithm 2.

## VII. Model and Result Extensions

In this section, we extend the investigation to a general case where $\theta_i$'s are non-identically distributed. Specifically, we denote the PDF and CDF of $\theta_i$ for the WDs with type-$j$ specific tasks as $f_j(\cdot)$ and $F_j(\cdot)$, respectively. Similar to Assumption 1 in Section IV, we assume that the distribution of $\theta_i$ is regular. That is, $y(\theta) = \theta - \frac{1-F_j(\theta)}{f_j(\theta)}$ is an increasing function of continuous random variable $\theta$ for all $j \in \mathcal{N}$.

Suppose that the edge CPU frequency allocated to each offloaded task is proportional to its computation workload. In this case, the task processing time of WD $i$ at the edge server is changed from (5) to

$$\tau_i^c(m) = \frac{L_{\varphi_i}}{L_{\varphi_i} f^c / (\Sigma_j m_j L_j)} = \frac{\Sigma_j m_j L_j}{f^c}, \tag{30}$$

where $m_j$ is the number of type-$j$ tasks offloaded to the edge server for edge computing, i.e., $m_j = \Sigma_{i=1}^{M} u_{i,j} a_i$.

In the following Theorem 2, we derive the optimal offloading strategy for each WD with each task type at the Bayesian equilibrium.

*Theorem 2:* A WD $i$ with type-$\varphi_i$ task will offload its task to the BS if and only if

$$\pi_{\varphi_i} \leq \theta_i - \delta^*_{\varphi_i}(\boldsymbol{\pi}), \tag{31}$$

where the equilibrium decision parameter $\delta^*_{\varphi_i}(\boldsymbol{\pi})$ is the same for all the WDs with the same type-$\varphi_i$ task, and the equilibrium decision parameters $\{\delta^*_{\varphi_i}(\boldsymbol{\pi}), \forall \varphi_i \in \mathcal{N}\}$ are the solutions to (32), shown at the bottom of the page.

*Proof:* Note that all the WDs with type-$j$ tasks other than WD $i$ ($\varphi_i = j$) choose to offload their tasks to the edge server if and only if their valuations $\beta_z, z \neq i$, are larger than the decision parameter $\delta_j > 0, j \in \mathcal{N}$. Then, WD $i$ with type-$j$ task will offload its task if

$$\theta_i - \pi_j \geq \mathbb{E}\left[\frac{m'_j + 1 + \Sigma_{k \neq j} m_k \frac{L_k}{L_j}}{f^c}\right], \tag{33}$$

where $m'_j$ is the number of the WDs with type-$j$ tasks other than WD $i$ that choose edge computing, and follows a binomial distribution $B(q_j M - 1, 1 - F_j(\delta_j + \pi_j))$. Besides, $m_k, k \neq j$ represents the number of WDs with type-$k$ tasks that prefer edge computing and follows a binomial distribution $B(q_k M, 1 - F_k(\delta_k + \pi_k))$. Accordingly, for the RHS of the inequlity (33), we have (34), shown at the bottom of the page. At the equilibrium, we have the common decision parameter $\delta_j$ for the WDs with type-$j$ tasks as shown in (35), shown at the bottom of the page. ■

Unlike Theorem 1, here we need to jointly determine $N$ equilibrium decision parameters by solving (32) in Stage II. According to Theorem 2, we propose an iterative algorithm to iteratively find the equilibrium decision parameters $\{\delta_j, j \in \mathcal{N}\}$. Specifically, in the $t$-th iteration, the algorithm finds the optimal $\{\delta_j^{(t)}, j \in \mathcal{N}\}$ according to (32) given $\boldsymbol{\delta}_{-j}^{(t-1)} = \{\delta_k^{(t-1)}, k \in \mathcal{N} \setminus j\}$. In the following corollary, we show that there exists a unique $\{\delta_j^{(t)}, j \in \mathcal{N}\}$ in each iteration $t$.

*Corollary 7.1:* Suppose that the other equilibrium decision parameters $\boldsymbol{\delta}_{-j}^{(t-1)}$ in the $(t-1)$-th iteration are given. There exists a unique $\delta_j^{(t)}, j \in \mathcal{N}$, that satisfies $\Phi_j(\delta_j^{(t)}) = 0$ in the $t$-th iteration.

*Proof:* The proof follows a similar technique in Theorem 1 and we omit the details here. ■

According to Corollary 7.1, given $\boldsymbol{\delta}_{-j}^{(t-1)}$, we can efficiently obtain the optimal equilibrium decision parameter $\delta_j^{(t)}$ for $j, j \in \mathcal{N}$, by a bi-section search method. The details of the proposed iterative algorithm to determine the equilibrium decision parameters are summarized in Algorithm 3.

**Algorithm 3** Computation of the Equilibrium Decision Parameters in Stage II Under the Extended General Case

1: Initializes the equilibrium decision parameters as $\{\delta_j^{(0)}, j \in \mathcal{N}\}$;
2: Set $t = 1$;
3: **repeat**
4: **for** Each task's type $j \in \mathcal{N}$ **do**
5: Apply bi-section search method to find $\delta_j^{(t)}$ that satisfies $\Phi_j(\delta_j^{(t)}) = 0$ with given $\boldsymbol{\delta}_{-j}^{(t-1)}$;
6: **end for**
7: Obtain $\{\delta_j^{(t)}, j \in \mathcal{N}\}$;
8: Set $t = t + 1$.
9: **until** $\{\delta_j^*, j \in \mathcal{N}\}$ converges.

$$\begin{cases} \Phi_1(\delta_1) := \delta_1 - \dfrac{(q_1 M - 1)(1 - F_1(\delta_1 + \pi_1)) + 1 + \sum_{k \in \mathcal{N} \setminus \{1\}} q_k M (1 - F_k(\delta_k + \pi_k)) \frac{L_k}{L_1}}{f^c} = 0, \\ \Phi_2(\delta_2) := \delta_2 - \dfrac{(q_2 M - 1)(1 - F_2(\delta_2 + \pi_2)) + 1 + \sum_{k \in \mathcal{N} \setminus \{2\}} q_k M (1 - F_k(\delta_k + \pi_k)) \frac{L_k}{L_2}}{f^c} = 0, \\ \ldots, \\ \Phi_j(\delta_j) := \delta_j - \dfrac{(q_j M - 1)(1 - F_j(\delta_j + \pi_j)) + 1 + \sum_{k \in \mathcal{N} \setminus j} q_k M (1 - F_k(\delta_k + \pi_k)) \frac{L_k}{L_j}}{f^c} = 0, \\ \ldots, \\ \Phi_N(\delta_N) := \delta_N - \dfrac{(q_N M - 1)(1 - F_N(\delta_N + \pi_N)) + 1 + \sum_{k \in \mathcal{N} \setminus \{N\}} q_k M (1 - F_k(\delta_k + \pi_k)) \frac{L_k}{L_N}}{f^c} = 0. \end{cases} \tag{32}$$

$$\mathbb{E}\left[\frac{m'_j + 1 + \Sigma_{k \neq j} m_k \frac{L_k}{L_j}}{f^c}\right] = \frac{(q_j M - 1)(1 - F_j(\delta_j + \pi_j)) + 1 + \sum_{k \in \mathcal{N} \setminus j} q_k M (1 - F_k(\delta_k + \pi_k)) \frac{L_k}{L_j}}{f^c}. \tag{34}$$

$$\delta_j = \frac{(q_j M - 1)(1 - F_j(\delta_j + \pi_j)) + 1 + \sum_{k \in \mathcal{N} \setminus j} q_k M (1 - F_k(\delta_k + \pi_k)) \frac{L_k}{L_j}}{f^c}, \forall j \in \mathcal{N}. \tag{35}$$

Based on the Bayesian equilibrium derived in Theorem 2, we turn to the optimization of service caching and pricing in Stage I. In this case, Problem (P1) in Stage I is expressed as

$$\text{(P4)} \quad \max_{(\boldsymbol{\pi},\mathbf{x})} \sum_{j=1}^{N}(1-F_j(\delta_j^*(\boldsymbol{\pi})+\pi_j))x_j q_j M \pi_j L_j - \sum_{j=1}^{N} x_j r_j,$$
$$\text{s.t.} \sum_{j=1}^{N} x_j c_j \le C,$$
$$x_j \in \{0,1\}, \quad \forall j = 1,\ldots,N. \tag{36}$$

Note that once the prices are given, we apply the proposed Algorithm 3 in Stage II to find the equilibrium decision parameters $\{\delta_j^*, j \in \mathcal{N}\}$. Then, the remaining optimization problem in (P4) for the service caching decisions is a standard Knapsack problem. Based on that, we can apply some meta-heuristic methods, e.g., particle swarm optimization [25], to search the optimal service prices that achieve the maximum objective value (the expected profit of BS) in Problem (P4) in Stage I.

## VIII. Simulation Results

In this section, we conduct numerical simulations to evaluate the performances of our proposed two-stage dynamic game of incomplete information for service caching, pricing, and task offloading in MEC systems. For simplicity of illustration, we normalize the unit cost for acquiring each program as $r_j = 1$. Besides, we assume that the program sizes $c_j$ are equal for all programs $j$, such that the caching space $C$ at the BS can easily tell the number of cached programs. $\theta_i$ is assumed to follow a uniform distribution between $-10\times10^{-8}$ and $10\times10^{-8}$. In general, most of the parameters chosen in the simulation are based on the parameter setting of a typical wireless network [27] and practical computing model [28]. Specifically, regarding the range of the random variable $\theta_i$, we consider that the peak computational CPU frequency of WDs is $10^9$ cycles/second as in [6]. Besides, for the wireless channel gain $h_i$ following the free-space path loss model, we have $h_i = A_d(\frac{3\cdot10^8}{4\pi f_c d_i})^{PL}$, where $f_c = 915$ MHz is the carrier frequency, $A_d = 4.11$ denotes the antenna gain, $PL = 3$ denotes the pass loss exponent, and $d_i$ in meters denotes the distance between WD $i$ and the BS. We assume that the transmit power of the WDs is 100 mW, the bandwidth $W = 2$ MHz, and the noise power $\sigma^2 = 10^{-10}$ W as in [4,5]. In addition, the input data size and the computation workload of a task are usually several thousand Kbyte and several hundred Mcycles, respectively [3-6]. Hence, by substituting the above parameters into the equation (12), we find that the quantity class of $\theta_i$ is at $10^{-8}$. Accordingly, we set the range of the random variable $\theta_i$ as $[-10*10^{-8}, 10*10^{-8}]$.

### A. WDs' Task Offloading Behaviors in Stage II

We first investigate the WDs' task offloading behaviors in Stage II. For illustration purpose, we consider a two-program case (i.e., $N = 2$) in the following. We assume that there are $M = 100$ WDs and the computation capability at the BS $f^c = 10^8$ cycles/second.

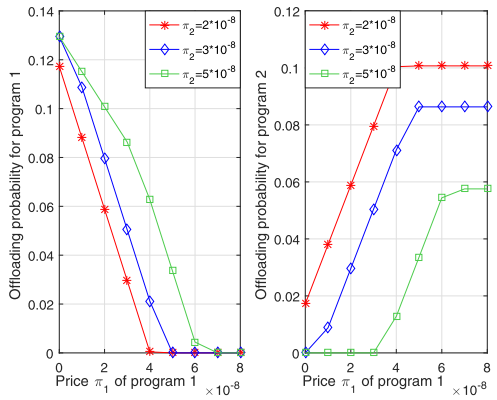



Fig. 3. Offloading probability as a function of $\pi_1$ under different $\pi_2$.

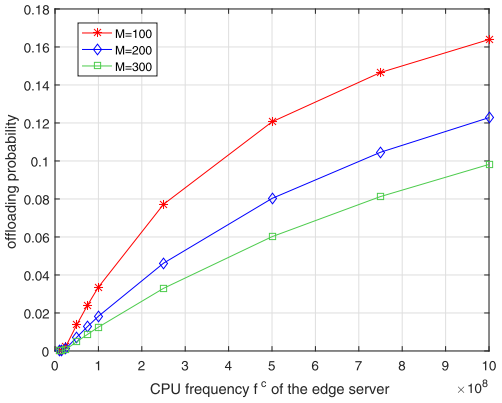
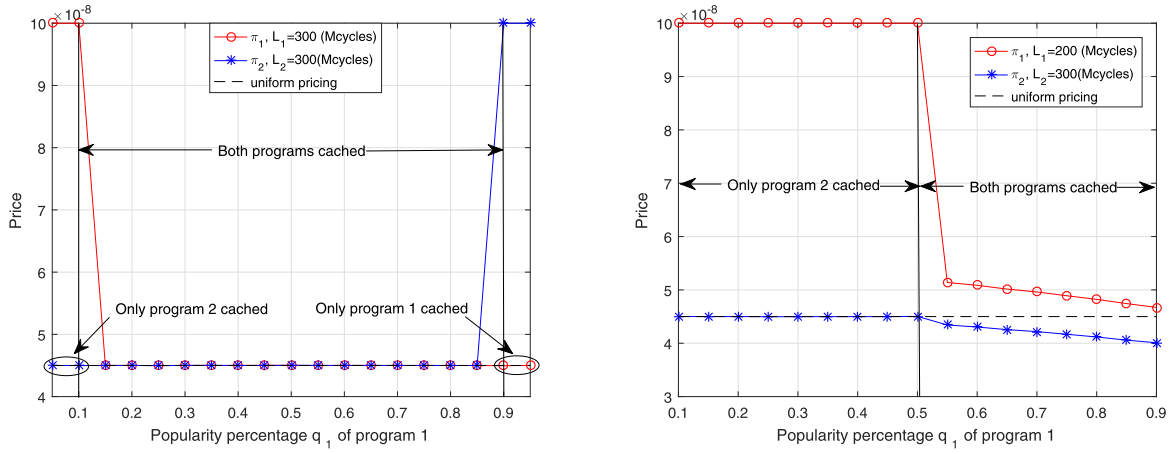



Fig. 4. Offloading probability as a function of $f^c$ under different numbers of WDs $M$.

Fig. 3 illustrates the impact of different programs' prices on the offloading probabilities of two types of tasks, where the program popularity $\{q_j\} = [0.5,\ 0.5]$. Given the price of program 2 in the first subfigure of Fig. 3, the offloading probability of type-1 tasks decreases with $\pi_1$. Besides, increasing $\pi_2$ leads to a higher offloading probability of type-1 tasks. It is due to the fact that increasing $\pi_2$ reduces the offloading probability of type-2 tasks, which makes more computation resource available for computing the type-1 tasks. Nevertheless, the offloading probability of type-2 tasks in the second subfigure of Fig. 3 shows an opposite trend, where the offloading probability increases with $\pi_1$ and decreases with $\pi_2$. An interesting observation is that when $\pi_1$ is high[5] (e.g., above $4\times10^{-8}$ when $\pi_2 = 2\times10^{-8}$), the offloading probability of type-2 tasks is fixed. It is because the offloading probability of type-1 tasks drops to zero as shown in the left subfigure of Fig. 3. In this case, further increasing the price of program 1 does not affect the offloading probability of type-2 tasks.

Fig. 4 studies the impact of the computation capability on the offloading probability of each program's tasks under different number of WDs $M$, where the program popularity $\{q_j\} = [0.5,\ 0.5]$. We set equal price $5\times10^{-8}$ for both

[5]In this article, the program prices are charged for unit CPU cycle. The total payment $\pi_1 L_1$ from the WD with type-1 task is nontrivial.




(a) Equal workloads of the two programs' tasks

(b) Different workloads of the two programs' tasks

Fig. 5. The price for each program versus the popularity $q_1$ in the two-program case.

programs, which leads to the same offloading probabilities of different types of tasks. We observe that the WDs' offloading probability under each program increases in the edge server's CPU frequency $f^c$. Besides, increasing $M$ or WDs' competition in sharing $f^c$ reduces the offloading probability. These also coincide with our results in Proposition 4.2.

### *B. BS' Service Caching and Pricing Strategies in Stage I*

Then, we show the properties of BS' service caching and pricing strategies in Stage I for the considered two-program case, where the BS is able to cache at most two programs, i.e., $C = 2$.

In Fig. 5, we demonstrate the impact of program popularity on the optimal prices, where $f^c = 10^8$ cycles/s and $M = 100$. For the uniform pricing heuristics, the uniform price is fixed regardless of the program popularity as shown in Proposition 6.1. For the differentiated pricing scheme, when the workloads of two programs are equal (i.e., $\{L_j\} = [300,\ 300]$ Mcycles), we observe from Fig. 5(a) that if both programs are cached, the prices of the two programs are the same and equal to that in the uniform pricing heuristics. When the popularity of program 1 is small (i.e., below 0.1), the BS does not cache program 1 and sets a sufficiently large price (i.e., $\pi_1 = 10 \times 10^{-8}$) for program 1 to guarantee zero offloading probability of type-1 tasks. It is because in this case, the BS' profit obtained by caching program 1 cannot compensate for the program 1's acquiring cost charged by the program provider. Besides, as shown in Fig. 5(b), when the workloads of two programs are different (i.e., $\{L_j\} = [200,\ 300]$ Mcycles), we can see that the BS always caches program 2 with larger workload. It is due to the fact that higher total revenue $L_2\pi_2$ is obtained for executing one type-2 task with larger workload. As $q_1$ is larger than 0.5, both programs 1 and 2 are cached and the price of program 1 (lower workload) is higher than that of program 2. Besides, we observe that, when both programs are cached, with the increase of $q_1$, both $\pi_1$ and $\pi_2$ decrease. It is because $q_2 = 1 - q_1$ in this case and equivalently, $\pi_2$ increases in $q_2$, which coincides with our analytical results in Proposition 5.4.

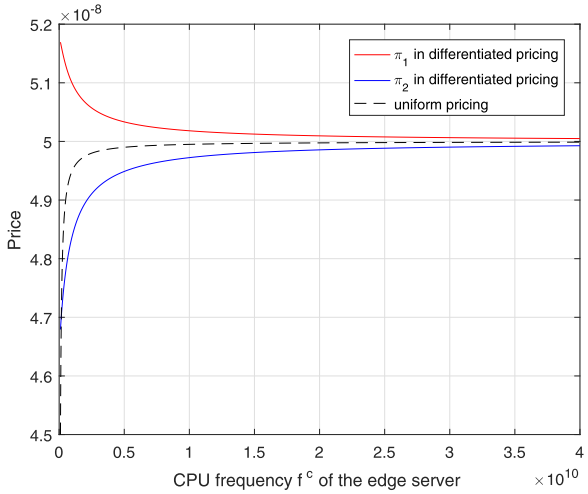
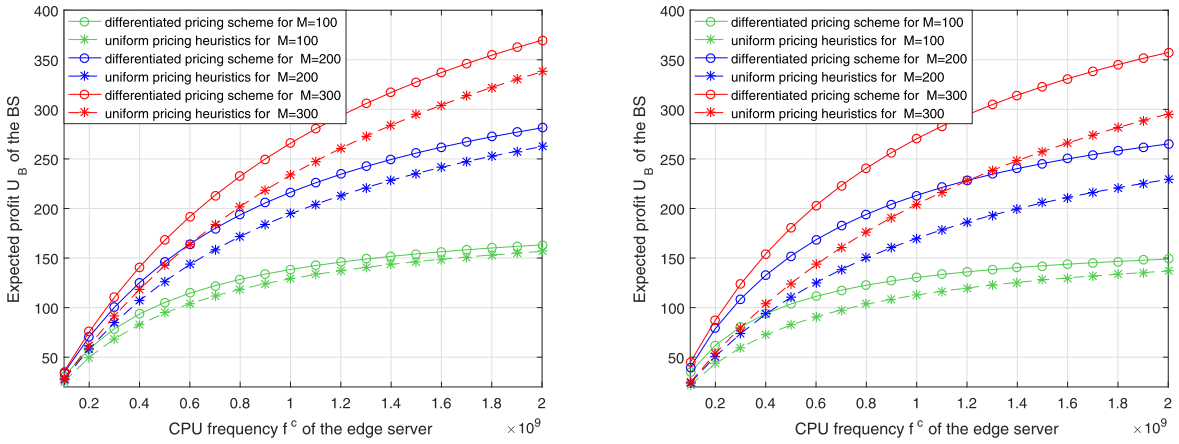



Fig. 6. The price for each program versus the edge computation capability $f^c$ in the two-program case.

Furthermore, Fig. 6 illustrates the impact of edge computation capability $f^c$ on the optimal prices, where $\{L_j\} = [200,\ 300]$ Mcycles, $\{q_j\} = [0.6,\ 0.4]$ and $M = 100$. For the differentiated pricing scheme, it is observed that when $f^c$ increases, the optimal price of the program 2 with larger workload increases. It is because the BS always sets a lower price for the program with larger workload in order to attract more WDs interested in this program to offload. Accordingly, as $f^c$ increases, the BS can decide a higher price $\pi_2$ and obtain a larger profit. However, the price $\pi_1$ of the program 1 with lower workload decreases in $f^c$ so as to incentivize offloading of type-1 tasks since the BS has more spare computation resource. In addition, we find that for a sufficiently large $f^c$, the prices of both programs tend to be the same, which is consistence with Corollary 5.2. Besides, we observe that the uniform price is in between the differentiated prices $\pi_1$ and $\pi_2$, as the uniform pricing heuristics is like an average way to coordinate the two cached programs and guide task offloading.




(a) Equal workloads (b) Different workloads

Fig. 7. The expected profit of the BS versus edge computation capability $f^c$ under different WDs number $M$.

### C. Performance Evaluation

We now evaluate and compare the performances of the proposed pricing algorithms in term of gaining MEC service profit for the BS. We assume that the WDs' tasks belong to $N = 3$ programs with popularity $\{q_j\} = [0.2,\ 0.4,\ 0.4]$. Here, we set $C = 2$ for at most two programs to cache.

Note that the uniform pricing heuristics (though easier and more fair to implement in practice) yields smaller expected profit for the BS. In Fig. 7, we present the expected profit of the BS using the proposed differentiated pricing and uniform pricing heuristics algorithms versus CPU frequency $f^c$ at the edge server and the number of WDs $M$ when the computing workloads for the programs are $\{L_j\} = [200,\ 200,\ 200]$ (Mcycles) and $\{L_j\} = [300,\ 200,\ 100]$ (Mcycles), respectively. We observe that as $f^c$ or $M$ increases, higher expected profit of the BS is obtained under both differentiated pricing scheme and uniform pricing heuristics, and uniform pricing heuristics still gains most profit of the differentiated pricing for different settings, telling the value to adopt simpler uniform pricing heuristics in practice. The profit gap between differentiated pricing upperbound and uniform pricing heuristics increases as there are more WDs with greater $M$ to tailor for each program's WDs. Besides, we observe that the performance gap between the differentiated pricing scheme and the uniform pricing heuristics in the equal workload scenario is smaller than that in the different workload scenario, e.g., $9.43\%$ and $20.96\%$ larger expected profit can be achieved by the proposed differentiated pricing scheme compared to the uniform pricing heuristics when $M = 300, f^c = 2 * 10^9$ and the programs have the same and different computation workload, respectively. It is because when the programs have the same workload, the optimal prices of the cached programs are equal. In this case, the performance loss caused by uniform pricing heuristics is only due to the approximation of the equilibrium parameter $\delta^*(\pi)$ in Stage II assuming that all the programs are cached and all the WDs are only informed of the uniform price.

Furthermore, we use real-world data to compare the performance of the proposed pricing algorithms with that of the following two representative benchmarks used in the literatures [29]–[31].

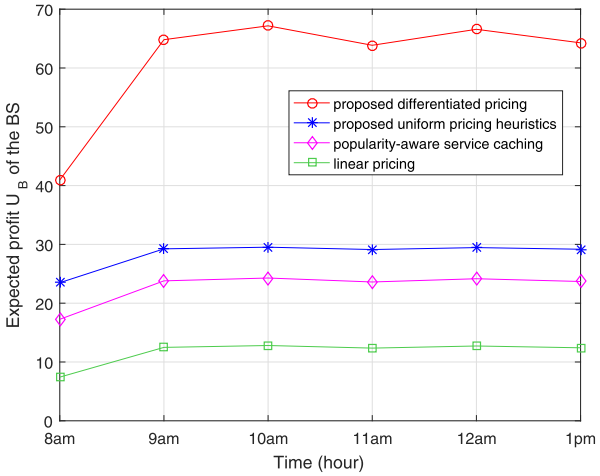



Fig. 8. Comparisons of the BS' expected profit performance for different pricing algorithms under the traffic monitored at the BS with cell ID 10012 from 8 am to 1 pm on April 19th, 2004, at Milan, Italy [32].

- Popularity-aware Service Caching: The BS caches the most popular service programs to fully occupy its storage [29].
- Linear Pricing: The BS charges the WDs with different programs through a linear pricing model [30], [31]. Specifically, the price $\pi_j$ of program $j$ is proportional to the computation workload of program $j$, i.e., $\pi_j = \chi L_j$. In this article, we set $\chi = 10^{-16}$. Note that the linear pricing model is commonly used on Amazon EC2 and Google [30], [31].

We test the performance on the real-world data traffic monitored at a typical BS with cell ID 10012 from 8 am to 1 pm on April 19th, 2004, at Milan, Italy [32]. It represents the number of served WDs by the BS at each specific time slot, which is time-varying within the period of interest. For each time slot, the BS does not need to obtain new service data for program $j$ if it is already cached in the previous time slot, i.e., the acquiring cost for program $j$ is zero. Otherwise, the BS

needs to acquire program $j$ from the program provider with the cost $r_j$.

In Fig. 8, we compare the BS' expected profit performances under different pricing algorithms, where the edge server CPU frequency $f^c = 10^8$ cycles/second and the computing workloads for the programs are $\{L_j\} = [500,\ 150,\ 100]$ (Mcycles). We observe from Fig. 8 that our proposed differentiated pricing algorithm achieves $137.23\%$ and $451.42\%$ higher expected profit than the popularity-aware service caching and linear pricing schemes at $8$ am, respectively. Besides, as a simplified version of differentiated pricing scheme, our proposed uniform pricing heuristics also outperforms the two representative benchmarks, e.g., $36.05\%$ and $216.24\%$ higher expected profit of the BS than the popularity-aware service caching and linear pricing schemes at $8$ am, respectively. This demonstrates the effectiveness and the benefits by jointly adapting selective program caching and pricing coordination for our proposed pricing algorithms on the real-world scenario.

## IX. Conclusion and Future Work

This article has studied a pricing mechanism to coordinate service caching and guide task offloading in an MEC system with one BS and multiple associated WDs. We have proposed a two-stage dynamic game of incomplete information to capture the interaction between the BS and WDs. In Stage I, the BS aims to maximize its expected profit by optimizing its service caching decisions and the programs prices for WDs' task executions under the limited computation resource and caching storage capacity. In Stage II, for given prices of service programs, the WDs play a Bayesian subgame and selfishly optimize offloading decisions to minimize their own costs by estimating the other WDs' decisions. We have first derived the threshold-based offloading strategy among the WDs at the Bayesian equilibrium. Then, by predicting the WDs' offloading equilibrium, in Stage I, we have developed the differentiated pricing algorithm and low-complexity uniform pricing heuristics to optimize the prices and service caching decisions at the BS. Simulation results have validated our analysis and shown the effectiveness of our proposed pricing mechanism.

Finally, we conclude the article with some future directions. First, in a multiple edge server scenario, the multiple BSs can share their cached service programs and collaboratively serve the WDs. As the dimensionality of the problem increases exponentially with the number of BSs, it is challenging to extend the proposed pricing mechanism to the scenario with cooperative service caching among the multiple BSs.[6] Second, it is challenging to consider the I/O interference for parallel computing at the edge server [33], [34]. Third, it is also interesting to consider the asynchronous offloading issue in MEC systems [35].

[6]One tractable way to handle the multiple edge server scenario is that suppose $M$ WDs are equally divided into $S$ groups and there are a number $S$ of BSs. Each WD can equally associate with these BSs. We define the WDs in group $s$ as those served by the $s$-th BS. Given the WDs in each group, we similarly apply the same pricing mechanisms proposed in the article to coordinate task offloading and service caching.

## Appendix A
## Proof of Proposition 4.1

According to Theorem 1, we have $\Phi(\pi_j, \delta^*(\pi_j)) = 0$. By applying implicit function theorem, we have $\frac{\partial \Phi(\delta^*)}{\partial \pi_j} + \frac{\partial \Phi(\delta^*)}{\partial \delta^*}\frac{\partial \delta^*}{\partial \pi_j} = 0, \forall j \in \mathcal{N}$. That is,

$$\frac{M-1}{f^c}\left[q_j \frac{\partial F(\delta^* + \pi_j)}{\partial(\delta^* + \pi_j)}\right] + \frac{\partial \delta^*}{\partial \pi_j}\left[1 + \frac{M-1}{f^c}\sum_j q_j \frac{\partial F(\delta^* + \pi_j)}{\partial(\delta^* + \pi_j)}\right] = 0. \quad (37)$$

Hence,

$$\frac{\partial \delta^*}{\partial \pi_j} = \frac{-(M-1)q_j f(\delta^* + \pi_j)}{f^c + (M-1)\sum_j q_j f(\delta^* + \pi_j)}, \quad \forall j \in \mathcal{N}.$$

Therefore, for the relation between $\delta^* + \pi_j$ and $\pi_j$, we have

$$\frac{\partial(\delta^* + \pi_j)}{\partial \pi_j} = 1 - \frac{(M-1)q_j f(\delta^* + \pi_j)}{f^c + (M-1)\sum_j q_j f(\delta^* + \pi_j)} > 0.$$

Besides, for the relation between $\delta^* + \pi_j$ and $\pi_k, k \in \mathcal{N} \setminus j$, we have

$$\frac{\partial(\delta^* + \pi_j)}{\partial \pi_k} = \frac{\partial \delta^*}{\partial \pi_k} = \frac{-(M-1)q_k f(\delta^* + \pi_k)}{f^c + (M-1)\sum_j q_j f(\delta^* + \pi_j)} < 0.$$

## Appendix B
## Proof of Proposition 4.2

We first prove the relation between $\delta^*$ and $f^c$. Equation (16) shows that $\Phi(\delta^*)$ is increasing in $f^c$. Recall that $\Phi(\delta^*)$ is increasing in $\delta^*$. Based on $\Phi(f^c, \delta^*(f^c)) = 0$, by applying implicit function theorem, we have $\frac{\partial \Phi(\delta^*)}{\partial f^c} + \frac{\partial \Phi(\delta^*)}{\partial \delta^*}\frac{\partial \delta^*}{\partial f^c} = 0$. Accordingly, we can derive $\frac{\partial \delta^*}{\partial f^c} = -\frac{\partial \Phi(\delta^*)}{\partial f^c} / \frac{\partial \Phi(\delta^*)}{\partial \delta^*} < 0$. Thus, $\delta^*$ is decreasing in $f^c$. Accordingly, given the price $\pi_j$, $\delta^* + \pi_j$ also decreases in $f^c$.

Then, we prove the relation between $\delta^*$ and $M$. Equation (16) shows that $\Phi(\delta^*)$ decreases with $M$. Suppose that $M_1 < M_2$ and $\Phi(M_1, \delta_1^*) = 0$. Because $\Phi(\delta^*)$ is increasing in $\delta^*$, to maintain $\Phi(M_2, \delta_2^*) = 0$, $\delta_2^* > \delta_1^*$ must hold. Therefore, $\delta^*$ is increasing in $M$. Accordingly, given the price $\pi_j$, $\delta^* + \pi_j$ also increases in $M$.

## Appendix C
## Proof of Proposition 6.2

We prove the existence, uniqueness and optimality of the solution, respectively.

*Existence and uniqueness:* According to (22), we define (38), shown at the bottom of the next page.

According to Proposition 4.1 in Stage II, $\delta(\pi) + \pi$ increases in $\pi$. Therefore, under regular distribution assumption, $\delta(\pi) + \pi - \frac{1 - F(\delta(\pi)+\pi)}{f(\delta(\pi)+\pi)}$ is an increasing function with respect to $\pi$. Besides, $-\frac{2(M-1)(\sum_{j\in\mathcal{N}} q_j)(1-F(\delta(\pi)+\pi))+1}{f^c}$ also increases in $\pi$. Hence, all terms in $\omega_{eq}(\pi)$ increase in $\pi$, thus $\omega_{eq}(\pi)$ is an increasing function in $\pi$. Meanwhile, when $\pi = 0$, we have $\omega_{eq} < 0$. When $\pi \to +\infty$, $\omega_{eq} \to +\infty$. Thus, there exists a unique $\pi^* > 0$ that satisfies $\omega_{eq}(\pi^*) = 0$.

*Optimality:* By substituting (22) into $\omega_d(\pi_j)$ for each cached program $j$, we have (39), shown at the bottom of the page. Since the price and workload for each program are equal, we have $L_1\pi_1^* = L_2\pi_2^* = \ldots = L_j\pi_j^*, \forall j \in \hat{\mathcal{N}}$. Therefore, we have $\omega_d(\pi_j^*) = 0, \forall j \in \hat{\mathcal{N}}$. It completes the proof.

## APPENDIX D
## PROOF OF PROPOSITION 6.3

Suppose that there are two programs $j$ and $k$ cached in the BS with popularity $q_j$ and $q_k$. From (17), the expected profit of the BS is

$$U_B = \frac{\bar{\theta} - (\delta^* + \pi_j)}{\bar{\theta} - \underline{\theta}} q_j M \pi_j L_j + \frac{\bar{\theta} - (\delta^* + \pi_k)}{\bar{\theta} - \underline{\theta}} q_k M \pi_k L_k.$$

According to Theorem 1, we can obtain the common equilibrium parameter $\delta^*$, (40) as shown at the bottom of the page.

Then, for the optimal price of program $j$, the derivative of $U_B$ with respect to $\pi_j$ can be expressed as

$$\frac{\partial U_B}{\partial \pi_j} = \frac{-(\frac{\partial \delta^*}{\partial \pi_j} + 1)}{\bar{\theta} - \underline{\theta}} q_j M \pi_j L_j + \frac{\bar{\theta} - (\delta^* + \pi_j)}{\bar{\theta} - \underline{\theta}} q_j M L_j + \frac{-\frac{\partial \delta^*}{\partial \pi_j}}{\bar{\theta} - \underline{\theta}} q_k M \pi_k L_k,$$

where $\frac{\partial \delta^*}{\partial \pi_j} = \frac{-(M-1)q_j}{f^c(\bar{\theta} - \underline{\theta}) + (M-1)(q_j + q_k)}$. By equating $\frac{\partial U_B}{\partial \pi_j} = 0$, we have

$$\pi_j^* = \frac{q_k(M-1)(L_j + L_k)}{2L_j[q_k(M-1) + f^c(\bar{\theta} - \underline{\theta})]} \pi_k + \frac{A}{2[q_k(M-1) + f^c(\bar{\theta} - \underline{\theta})]}, \quad (41)$$

where $A = f^c\bar{\theta}(\bar{\theta} - \underline{\theta}) + (M-1)(q_j + q_k)(\bar{\theta} - \underline{\theta}) - [(M-1)(q_j + q_k) + 1](\bar{\theta} - \underline{\theta})$.

Similarly, we can obtain the optimal price for the cached program $k$, i.e.,

$$\pi_k^* = \frac{q_j(M-1)(L_j + L_k)}{2L_k[q_j(M-1) + f^c(\bar{\theta} - \underline{\theta})]} \pi_j + \frac{A}{2[q_j(M-1) + f^c(\bar{\theta} - \underline{\theta})]}. \quad (42)$$

By combining (41) and (42), we have (43), shown at the bottom of the page, and (44), shown at the bottom of the page. According to (43) and (44), when $L_j = L_k$, we have $\pi_j^* = \pi_k^* = \frac{\bar{\theta}}{2} - \frac{1}{2f^c}$. Besides, we calculate (45), shown at the bottom of the page. Therefore, when $L_j > L_k$, we have $\pi_j^* < \pi_k^*$. If $L_j < L_k$, $\pi_j^* > \pi_k^*$.

## APPENDIX E
## PROOF OF PROPOSITION 6.4

According to (43), we calculate the first derivative with respect to $q_j$, as shown in (46), as shown at the top of the next page. If $L_j > L_k$, we have $\frac{\partial \pi_j^*}{\partial q_j} > 0, \forall q_k > 0$. If $L_j < L_k$,

$$\begin{aligned}\omega_{eq}(\pi) &= \pi - \frac{1 - F(\delta(\pi) + \pi)}{f(\delta(\pi) + \pi)} - \frac{(M-1)(\sum_{j \in \hat{\mathcal{N}}} q_j)(1 - F(\delta(\pi) + \pi))}{f^c} \\ &= \delta(\pi) + \pi - \frac{1 - F(\delta(\pi) + \pi)}{f(\delta(\pi) + \pi)} - \frac{2(M-1)(\sum_{j \in \hat{\mathcal{N}}} q_j)(1 - F(\delta(\pi) + \pi)) + 1}{f^c}. \end{aligned} \quad (38)$$

$$\begin{aligned}\omega_d(\pi_j) &= q_j M L_j \Bigg[ [f^c + (M-1) \sum_{j \in \hat{\mathcal{N}}} q_j f(\delta^* + \pi_j)] \frac{f^c}{f^c + (M-1) \sum_{j \in \hat{\mathcal{N}}} q_j f(\delta^* + \pi_j)} \pi_j \\ &\quad - (f^c + (M-1) \sum_{k \neq j, k \in \hat{\mathcal{N}}} q_k f(\delta^* + \pi_k)) \pi_j \Bigg] + (M-1) q_j \sum_{k \neq j, k \in \hat{\mathcal{N}}} f(\delta^* + \pi_k) q_k M \pi_k L_k \\ &= -(M-1) q_j \sum_{k \neq j, k \in \hat{\mathcal{N}}} q_k f(\delta^* + \pi_k) M \pi_j L_j + (M-1) q_j \sum_{k \neq j, k \in \hat{\mathcal{N}}} q_k f(\delta^* + \pi_k) M \pi_k L_k, \forall j \in \hat{\mathcal{N}}. \end{aligned} \quad (39)$$

$$\delta^* = \frac{[(M-1)(q_j + q_k) + 1](\bar{\theta} - \underline{\theta}) - (M-1)[q_j(\pi_j - \underline{\theta}) + q_k(\pi_k - \underline{\theta})]}{f^c(\bar{\theta} - \underline{\theta}) + (M-1)(q_j + q_k)}. \quad (40)$$

$$\pi_j^* = \frac{L_k[2L_j q_j(M-1) + 2L_j f^c(\bar{\theta} - \underline{\theta}) + q_k(M-1)(L_j + L_k)]A}{4L_j L_k[q_j q_k(M-1)^2 + f^c(\bar{\theta} - \underline{\theta})(M-1)(q_j + q_k) + (f^c)^2(\bar{\theta} - \underline{\theta})^2] - q_j q_k(M-1)^2(L_j + L_k)^2}, \quad (43)$$

$$\pi_k^* = \frac{L_j[2L_k q_k(M-1) + 2L_k f^c(\bar{\theta} - \underline{\theta}) + q_j(M-1)(L_j + L_k)]A}{4L_j L_k[q_j q_k(M-1)^2 + f^c(\bar{\theta} - \underline{\theta})(M-1)(q_j + q_k) + (f^c)^2(\bar{\theta} - \underline{\theta})^2] - q_j q_k(M-1)^2(L_j + L_k)^2}. \quad (44)$$

$$\pi_j^* - \pi_k^* = \frac{A[q_j L_j(L_k - L_j) + q_k L_k(L_k - L_j)]}{4L_j L_k[q_j q_k(M-1)^2 + f^c(\bar{\theta} - \underline{\theta})(M-1)(q_j + q_k) + (f^c)^2(\bar{\theta} - \underline{\theta})^2] - q_j q_k(M-1)^2(L_j + L_k)^2}. \quad (45)$$

$$\frac{\partial \pi_j^*}{\partial q_j} = \frac{A(M-1)^2 L_k (L_j - L_k)(L_j + L_k) q_k [2L_j f^c(\bar{\theta} - \underline{\theta}) + (M-1)(L_j - L_k) q_k]}{\left[4L_j L_k [q_j q_k (M-1)^2 + f^c(\bar{\theta} - \underline{\theta})(M-1)(q_j + q_k) + (f^c)^2(\bar{\theta} - \underline{\theta})^2] - q_j q_k (M-1)^2 (L_j + L_k)^2\right]^2}. \quad (46)$$

when $0 < q_k < \frac{2f^c(\bar{\theta}-\underline{\theta})L_j}{(L_k - L_j)(M-1)}$, $\frac{\partial \pi_j^*}{\partial q_j} < 0$. Otherwise, we have $\frac{\partial \pi_j^*}{\partial q_j} > 0$.

## REFERENCES


[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

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

[3] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

[4] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

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

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

[7] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[8] E. Jonas *et al.*, "Cloud programming simplified: A Berkeley view on serverless computing," 2019, *arXiv:1902.03383*. [Online]. Available: http://arxiv.org/abs/1902.03383

[9] T. Zhao, I.-H. Hou, S. Wang, and K. Chan, "Red/LeD: An asymptotically optimal and scalable online algorithm for service caching at the edge," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1857–1870, Aug. 2018.

[10] T. He, H. Khamfroush, S. Wang, T. L. 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. (ICDCS)*, Jul. 2018, pp. 365–375.

[11] Q. Xie, Q. Wang, N. Yu, H. Huang, and X. Jia, "Dynamic service caching in mobile edge networks," in *Proc. IEEE 15th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2018, pp. 73–79.

[12] L. Chen, J. Xu, S. Ren, and P. Zhou, "Spatio–temporal edge service placement: A bandit learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8388–8401, Dec. 2018.

[13] 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.*, Apr. 2018, pp. 207–215.

[14] S. Bi, L. Huang, and Y.-J.-A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.

[15] D. Fudenberg and J. Tirole, *Game Theory*. Cambridge, MA, USA: MIT Press, 1991.

[16] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, "Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1701–1709.

[17] C. A. Gizelis and D. D. Vergados, "A survey of pricing schemes in wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 1, pp. 126–145, 1st Quart., 2011.

[18] L. Duan, J. Huang, and B. Shou, "Economics of femtocell service provision," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2261–2273, Nov. 2013.

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

[20] X. Li, C. Zhang, B. Gu, K. Yamori, and Y. Tanaka, "Optimal pricing and service selection in the mobile cloud architectures," *IEEE Access*, vol. 7, pp. 43564–43572, 2019.

[21] L. Li, M. Siew, T. Q. Quek, J. Ren, Z. Chen, and Y. Zhang, "Learning-based priority pricing for job offloading in mobile edge computing," 2019, *arxiv:1905.07749*. [Online]. Available: http://arxiv.org/abs/1905.07749

[22] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit maximization incentive mechanism for resource providers in mobile edge computing," *IEEE Trans. Services Comput.*, early access, Jun. 24, 2019, doi: 10.1109/TSC.2019.2924002.

[23] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[24] C. Ewerhart, "Regular type distributions in mechanism design and $\rho$-concavity," *Econ. Theory*, vol. 53, pp. 591–603, May 2012.

[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov./Dec. 1995, pp. 1942–1948.

[26] L. P. Qian, Y. J. A. Zhang, and M. Chiang, "Distributed nonconvex power control using Gibbs sampling," *IEEE Trans. Commun.*, vol. 60, no. 12, pp. 3886–3898, Dec. 2012.

[27] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. Ylä-Jääski, "Practical power modeling of data transmission over 802.11g for wireless applications," in *Proc. 1st Int. Conf. Energy-Efficient Comput. Netw.*, 2010, pp. 75–84.

[28] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX HotCloud*, Jun. 2010, pp. 1–7.

[29] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud RAN," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6118–6131, Sep. 2016.

[30] E. Ibrahim, N. A. El-Bahnasawy, and F. A. Omara, "Task scheduling algorithm in cloud computing environment based on cloud pricing models," in *Proc. World Symp. Comput. Appl. Res. (WSCAR)*, Mar. 2016, pp. 65–71.

[31] R. Sahal and F. A. Omara, "Effective virtual machine configuration for cloud environment," in *Proc. 9th Int. Conf. Informat. Syst.*, Dec. 2014, pp. 32–33.

[32] C. Ratti, D. Frenchman, R. M. Pulselli, and S. Williams, "Mobile landscapes: Using location data from cell phones for urban analysis," *Environ. Planning B, Planning Des.*, vol. 33, no. 5, pp. 727–748, Oct. 2006.

[33] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 560–569, Mar. 2014.

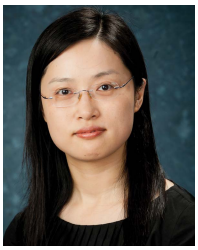
[34] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang, "Multiuser computation offloading and downloading for edge computing with virtualization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4298–4311, Sep. 2019.

[35] C. You, Y. Zeng, R. Zhang, and K. Huang, "Asynchronous mobile-edge computation offloading: Energy-efficient resource management," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7590–7605, Nov. 2018.


**Jia Yan** (Graduate Student Member, IEEE) received the B.Eng. degree from the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China, in 2017. He is currently pursuing the Ph.D. degree in information engineering with The Chinese University of Hong Kong, Hong Kong. His research interests include optimization and machine learning techniques in wireless communications and networking, particularly in mobile edge computing and edge intelligence.

**Suzhi Bi** (Senior Member, IEEE) received the B.Eng. degree in communications engineering from Zhejiang University in 2009 and the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2013. From 2013 to 2015, he was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore. Since 2015, he has been with the College of Electronics and Information Engineering, Shenzhen University, China, where he is currently an Associate Professor. He is also an Adjunct Research Associate with the Peng Cheng Laboratory, Shenzhen, China. His research interests include optimizations in wireless information and power transfer, mobile computing, and smart power grid communications. He received the Guangdong Province "Pearl River Young Scholar" Award in 2018, the IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award in 2019, and two times Shenzhen University Outstanding Young Faculty Award. He was a co-recipient of the IEEE SmartGridComm 2013 Best Paper Award. He is an Associate Editor of the IEEE WIRELESS COMMUNICATIONS LETTERS.

**Lingjie Duan** (Senior Member, IEEE) received the Ph.D. degree from The Chinese University of Hong Kong in 2012. In 2011, he was a Visiting Scholar with the University of California at Berkeley, Berkeley, CA, USA. He is currently an Associate Professor of engineering systems and design with the Singapore University of Technology and Design (SUTD). His research interests include network economics and game theory, cognitive communications, UAV networking, and energy harvesting wireless communications. He was also the finalist of the Hong Kong Young Scientist Award under Engineering Science Track in 2014. He received the 10th IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award in 2015 and the SUTD Excellence in Research Award in 2016. He was an Editor of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS. He also served as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Special Issue on Human-in-the-Loop Mobile Networks and the *IEEE Wireless Communications Magazine*. He is currently an Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.



**Ying-Jun Angela Zhang** (Fellow, IEEE) received the Ph.D. degree from The Hong Kong University of Science and Technology, Hong Kong, in 2004. She is currently a Professor with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. Her research interests include optimization and learning in communication and information systems.

She is currently a fellow of IET. She was a recipient of the 2011 Young Researcher Award of The Chinese University of Hong Kong, and a co-recipient of the 2014 IEEE Comsoc Asia Pacific Outstanding Paper Award, the 2013 IEEE SmartGridComm Best Paper Award, and the 2011 IEEE Marconi Prize Paper Award on Wireless Communications. As the only winner from Engineering Science, she received the Hong Kong Young Scientist Award 2006, conferred by the Hong Kong Institution of Science. She was the Chair of the Executive Editor Committee of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and the Chair of IEEE Technical Committee on Smart Grid Communications. She is currently a member of the Steering Committee of the IEEE WIRELESS COMMUNICATIONS LETTERS and of the IEEE SmartGridComm Conference, and an Associate Editor-in-Chief of the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY (OJ-COMS). She served many years on the Editorial Boards for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and the IEEE TRANSACTIONS ON COMMUNICATIONS. She also served as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE INTERNET OF THINGS JOURNAL, and the *IEEE Communications Magazine*. She is a Distinguished Lecturer of IEEE ComSoc.