Optimal Service Caching and Pricing in Edge Computing: a Bayesian Gaussian Process Bandit Approach

Feridun Tütüncüoğlu and György Dán
Division of Network and Systems Engineering, School of Electrical Engineering and Computer Science
KTH, Royal Institute of Technology, Stockholm, Sweden
Email: {feridun, gyuri}@kth.se

***Abstract*—Motivated by the emergence of function-as-a-service (FaaS) as a programming abstraction for edge computing, we consider the problem of caching and pricing applications for edge computation offloading in a dynamic environment where *Wirelesss Devices* (WDs) can be active or inactive at any point in time. We model the problem as a single leader multiple-follower Stackelberg game, where the service operator is the leader and decides what applications to cache and how much to charge for their use, while the WDs are the followers and decide whether or not to offload their computations. We show that the WDs' interaction can be modeled as a player-specific congestion game and show the existence and computability of equilibria. We then show that under perfect and complete information the equilibrium price of the service operator can be computed in polynomial time for any cache placement. For the incomplete information case, we propose a Bayesian Gaussian Process Bandit algorithm for learning an optimal price for a cache placement and provide a bound on its asymptotic regret. We then propose a Gaussian process approximation-based greedy heuristic for computing the cache placement. We use extensive simulations to evaluate the proposed learning scheme, and show that it outperforms state of the art algorithms by up to 50% at little computational overhead.**

I. Introduction

Battery powered *Wireless Device*s (WDs) are increasingly used for computationally intensive applications such as augmented reality, natural language processing, face, gesture and object recognition [1, 2]. Nonetheless, executing these kinds of applications on WDs results in high energy consumption and can adversely affect battery lifetime and the user experience.

Edge computing could become a promising solution for offloading computationally intensive tasks from WDs to nearby compute resources in the infrastructure via wireless networks. By computation offloading, it could become possible for WDs to reduce their energy consumption, while meeting application latency requirements. Nonetheless, if many WDs offload simultaneously, application performance could suffer due to congestion on the limited wireless and computational resources in the edge infrastructure. This realization has spun a great interest in edge resource management, pricing and admission control [3]–[7].

Nonetheless, the management of storage and its interaction with wireless and computing resource management have received much less attention in the literature [8, 9]. Storage is an essential prerequisite for the availability of executable code and data at the edge server, and hence for computation offloading. Code availability and pricing for computation offloading become particularly important in the case of emerging *Function as a Service* (FaaS) offerings (also called serverless computing) where tasks are executed on-demand, by loading container images from storage to memory, and charging is based on execution time. Yet, optimizing code availability and pricing is challenging, as the price and loaded container images affect the decisions of WDs, and the service operator may not have access to information about the WDs and their workloads a priori, e.g., in emerging mobile network architectures [10]–[12].

In this work, we study this important problem. We explore the interaction between a profit-maximizing service operator that performs storage management and pricing, and cost-minimizing autonomous WDs that can offload their computation, subject to application availability and latency constraints. We provide an analysis of the strategic interaction between the service operator and the WDs under complete and perfect information. We then consider the incomplete information case where the service operator has to learn what applications to cache and what price to charge through repeated interaction with the devices, whose decisions whether or not to offload are orchestrated by a network operator. Our main contributions are as follows.

- We propose a Stackelberg game to model the interaction between the service operator and the WDs.
- We show that the interaction of the WDs can be modeled by a player-specific congestion game and we prove the existence of pure strategy *Nash Equilibria*.
- We propose a polynomial time algorithm for computing the optimal price to be charged by the service operator.
- We propose a Bayesian *Gaussian Process* (GP) Bandit based approach to approximate *Subgame Perfect Equilibria* (SPE) of the game under incomplete information.
- We use extensive simulations for showing that the re-

The work was partly funded by the Vinnova Center for Trustworthy Edge Computing Systems and Applications (TECoSA) and the Swedish Research Council through project 2020-03860. The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Linköping University partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

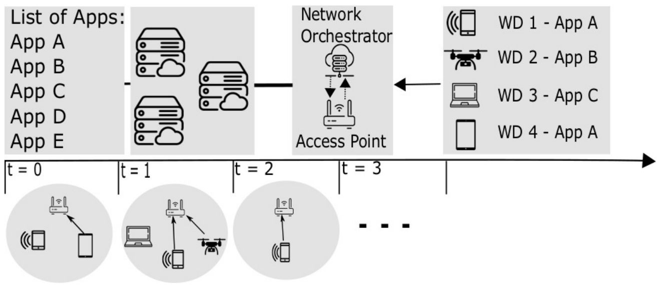

Fig. 1. System with $N = 4$ WDs and $|\mathcal{J}| = 5$ apps. For each time slot the figure shows the active WDs, some of which offload (indicated by an arrow to the access point). The set of offloaders depends on the set $\mathcal{X}(t)$ of cached apps and the price $\pi(t)$, the decision is coordinated by a network orchestrator.

sulting solution outperforms state-of-the-art *Multi-Armed Bandit* (MAB) based algorithms with a small computation overhead.

The rest of the paper is organized as follows. We present the system model and problem formulation in Section II. We show the existence of *Nash Equilibrium* (NE) and we propose an algorithm for computing the optimal pricing under complete information in Section III. We present the proposed *Bayesian Revenue Maximization* (BRM) algorithm for learning a SPE under incomplete information and we provide a regret analysis in Section IV. We show numerical results in Section V, and discuss related work in Section VI. Section VII concludes the paper.

$\mathcal{N}$	Set of WDs
$\mathcal{J}$	Set of applications
t	Time slot index
T	Time horizon
q_i	Activeness probability of WD i
$a_i(t)$	Offloading decision of WD i at time slot t
$\mathcal{N}^a(t)$	Set of active WDs at time slot t
ϕ_i	Type of the application that WD i wants to execute
L_{ϕ_i}	Expected number of cycles per byte for WD i
D_i	Size of the input data of WD i
$\mathcal{X}(t)$	Set of cached apps at time slot t, $\mathcal{X}(t) \subseteq \mathcal{J}$
$\bar{\tau}_i$	Completion time requirement of WD i
s_j	Storage size of app $j \in \mathcal{J}$
S	Storage capacity of the edge servers
$\pi(t)$	Price determined by the service operator at time slot t
τ_i^l	Local execution time of the task of WD i
f_i^l	Local processing frequency of WD i
$m(t)$	Number of offloaders at time slot t
p_i	Transmission power of WD i
$\hat{p}_i$	Maximum transmission power of WD i
R_i^u	Transmission rate of WD i
W	Channel bandwidth
h_i	Channel coefficient from WD i to AP
$\tilde{\sigma}_i^2$	Noise power at the AP for WD i
τ_i^u	Upload time of WD i
f^c	Computing frequency of the edge server
τ_i^c	Execution time of WD i's task at the edge servers
γ_i^l	Power use coefficient of WD i
β_i	Unit energy cost of WD i
$C_i^0(t)$	Local computing cost of WD i at time slot t
$C_i^1(t)$	Offloading cost of WD i at time slot t
$R(\mathcal{X}(t), \pi(t))$	Service operator's instantaneous reward at t
$\bar{r}(\mathcal{X}(t), \pi(t))$	Average reward of the service operator

TABLE I
TABLE OF NOTATIONS.

II. System Model and Problem Formulation

We consider a multi-access edge computing system that consists of an edge server with storage capacity S managed by a service operator, and a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of WDs that can offload their computational task for execution at the edge server via a wireless link. Time is slotted, and we consider that each WD $i \in \mathcal{N}$ is active with probability $q_i > 0$ in time slot t, independent of other WDs and of its activity in previous time slots [13, 14]. We consider that a single time slot is long enough for performing each user's task both in the case of local computing and in the case of computation offloading. This assumption is reasonable in the case of real time applications if the worst-case task completion time is less than the time slot length. We define the random variable $B_i(t)$ to model whether or not WD i is active, i.e., $P(B_i(t) = 1) = q_i$, and define the set $\mathcal{N}^a(t) = \{i \in \mathcal{N} | B_i(t) = 1\}$ of active WDs in time slot t. An inactive WD has no task to execute in time slot t, while if active, WD i wants to execute a task of type $\phi_i \in \mathcal{J}$, where $\mathcal{J}$ is the set of applications (i.e., the set of task types). The applications are the software images required for the execution of the tasks; tasks of different WDs may need the same application image. The computational task of WD i is characterized by the size D_i of the input data (e.g, in bytes), by the expected number L_j of cycles per byte required to perform the task (e.g, in Gcycles/byte) for $j = \phi_i$, and by the completion time requirement $\bar{\tau}_i$.

At the beginning of time slot t, the service operator can decide to cache a subset $\mathcal{X}(t) \subseteq \mathcal{J}$ of applications, subject to its storage capacity constraint

$$\sum_{j \in \mathcal{X}(t)} s_j \leq S, \tag{1}$$

where s_j is the size of the software image for application j. Caching application j in time slot t involves a usage cost $c_j \in \mathbb{R}^+$ to the service operator, e.g., corresponding to the cost of licensing the application from its owner.

If WD i is active in time slot t ($B_i(t) = 1$) and the application it intends to use is cached by the service operator ($\phi_i \in \mathcal{X}(t)$) then WD i can decide to offload the computation to the edge server. We denote by $a_i(t)$ the offloading decision of WD i; $a_i(t) = 1$ corresponds to offloading, and $a_i(t) = 0$ to local computing for time slot t. A WD that offloads in time slot t is charged unit price $\pi(t) \geq 0$ by the service operator. We denote by $\mathcal{P} = [0, \overline{\pi}]$ the domain of prices and assume it is compact. The active WDs use an orchestrator provided by the network operator for coordinating whether or not to offload. We consider that the price $\pi(t)$ is application independent, aligned with pricing in current FaaS offerings. The service operator has to choose and announce the price $\pi(t)$ before the set $\mathcal{N}^a(t)$ of active WDs becomes known, together with the caching decision $\mathcal{X}(t)$.

Next, we present our model of local computing and computation offloading, followed by the problem formulation.

A. Local Computing

If WD i chooses to perform the task locally, the task needs to be executed using local computational resources. We denote by f_i^l the local processing capability (frequency) of WD i, and express the local processing time as

$$\tau_i^l = \frac{L_{\phi_i} D_i}{f_i^l}. \tag{2}$$

We consider that f_i^l can be chosen such that local computing ensures that the task is completed just upon its deadline, i.e., $\tau_i^l = \bar{\tau}_i$. This assumption is reasonable, as dynamic frequency scaling is widely used for reducing the energy consumption of battery powered WDs while meeting performance needs [15].

B. Computation Offloading

If WD i decides to offload, it has to transmit D_i amount of data over the wireless channel to the edge server via an *Access Point* (AP), and then processing is performed at the edge server. We denote by

$$m(t) = \sum_{i=1}^{N} a_i(t), \tag{3}$$

the number of WDs that offload at time slot t, and for simplicity we consider that the available frequency spectrum and the edge processing capacity are equally shared among offloaders. More complex models of resource sharing could be used in practice.

For data transmission, we make the common assumption of a Gaussian channel [8, 16], and we express the data rate achievable by WD i using the Shannon formula [17],

$$R_i^u(p_i, m(t)) = \frac{W}{m(t)} \log_2(1 + \frac{p_i h_i}{\tilde{\sigma}_i^2}), \tag{4}$$

where W is the channel bandwidth, h_i is the channel coefficient from WD i to the AP, p_i is the transmit power of the WD i, $\tilde{\sigma}_i^2$ is the noise power at the AP, and bandwidth is shared equally among the $m(t)$ WDs that decide to offload. The transmission power is bounded by the maximum transmission power $\hat{p}_i$, i.e., $p_i \leq \hat{p}_i$. Given the data rate, we can express the upload time as

$$\tau_i^u(p_i, m(t)) = \frac{D_i}{R_i^u(p_i, m(t))}. \tag{5}$$

We denote by f^c the computing capability of the edge server, and we consider that it is equally shared among the tasks that are offloaded, consequently we can model the the processing time at the edge server as

$$\tau_i^c(m(t)) = \frac{L_{\phi_i} D_i}{f^c / m(t)}. \tag{6}$$

C. WD Cost Model

We model the cost of WD i as a combination of its energy consumption and the price charged by the service operator for computation offloading. In the case of local computing the cost is due to the energy consumed by the local processor to execute the task, i.e.,

$$C_i^0(t) = \tau_i^l (f_i^l)^2 \gamma_i^l \beta_i, \tag{7}$$

where γ_i^l is the power use coefficient, and β_i is the unit energy cost of WD i.

In the case of offloading the cost is the sum of the energy consumption of the transmission of the input data and the execution cost that is to be paid to the service operator. We consider that the execution cost is proportional to the task complexity L_{ϕ_i} and the input size D_i, which is reasonable for today's FaaS offerings following the pay-as-you-go model. The cost of WD i in the case of offloading is thus

$$C_i^1(t) = \tau_i^u(p_i, m(t)) p_i \beta_i + L_{\phi_i} D_i \pi(t), \tag{8}$$

where π is the unit price charged by the service operator. The cost of WD i is thus in each time slot

$$\begin{aligned} C_i(a_i(t), p_i, a_{-i}(t)) =& (1 - a_i(t))(\tau_i^l (f_i^l)^2 \gamma_i^l \beta_i) + \\ & a_i(t)(\tau_i^u(p_i, m(t)) p_i \beta_i + L_{\phi_i} D_i \pi(t)). \end{aligned} \tag{9}$$

where $a_{-i}(t)$ denotes the offloading decisions of WDs $\forall i' \in \mathcal{N} \setminus \{i\}$.

D. Problem Formulation

We consider that the WDs and the service operator are rational, strategic entities. The objective of WD i is to minimize its cost subject to its completion time requirement, the constraint on the maximum transmission power, and the caching decision $\mathcal{X}$ of the service operator for each time slot, i.e., at time slot t it aims to solve

$$\begin{aligned} \min_{a_i(t) \in \{0,1\}, p_i \leq \hat{p}_i} \quad & C_i(a_i(t), p_i, a_{-i}(t)) \\ \text{s.t.} \quad & \\ a_i(t)(\tau_i^u(p_i, m(t)) + \tau_i^c(m(t))) \quad & \leq \quad \tau_i^l, \\ a_i(t) = 0 \;\; \text{if } \phi_i \notin \mathcal{X}(t), & \end{aligned} \tag{10}$$

where the first constraint ensures that WD i does not offload if $\tau_i^u(p_i, m(t)) + \tau_i^c(m(t)) > \tau_i^l$, and the second constraint ensures that it offloads only if application ϕ_i is cached by the service operator.

The service operator's reward at time slot t is the difference of its income from the WDs that offload and the cost of caching applications $\mathcal{X}(t)$,

$$\begin{aligned} & R(\mathcal{X}(t), \pi(t)) = \\ & = \sum_{i \in \mathcal{N}^a(t)} a_i(t) L_{\phi_i} D_i \pi(t) \mathbb{1}_{\mathcal{X}(t)}(\phi_i) - \sum_{j \in \mathcal{X}(t)} c_j, \end{aligned} \tag{11}$$

where $\mathbb{1}_{\mathcal{X}(t)}(\phi_i)$ is the indicator function. Observe that $R(\mathcal{X}(t), \pi(t))$ depends on the set $\mathcal{N}^a(t)$ of active WDs, and hence it is a random variable. The expected reward of the service operator in time slot t is the expectation with respect to the distribution of the set of active WDs,

$$\bar{r}(\mathcal{X}(t), \pi(t)) = E_B[R(\mathcal{X}(t), \pi(t))]. \tag{12}$$

The service operator can choose a policy κ for computing $\mathcal{X}(t)$ and $\pi(t)$ based on past actions $\mathcal{X}(\tau)$ and $\pi(\tau)$, and observations, including the set $\{i | a_i(\tau) = 1\}$ of offloading WDs and the obtained rewards $R(\mathcal{X}(\tau), \pi(\tau))$, $\tau \in \{0, \ldots, t-1\}$. Let us denote by $\mathcal{K}$ the set of policies of the service operator.

For a policy $\kappa \in \mathcal{K}$ we define the expected average regret of the service operator up to time T as the loss of reward compared to a static decision $\mathcal{X}^*$, π^* with maximum expected reward,

$$\rho^{\kappa}(T) = \frac{1}{T}\sum_{t=1}^{T} E_B\left[R(\mathcal{X}^*,\pi^*) - R(\mathcal{X}^{\kappa}(t),\pi^{\kappa}(t))\right]. \quad (13)$$

The objective of the service operator is to find a policy κ^* that asymptotically minimizes the expected average regret,

$$\kappa^* \in \arg\min_{\kappa\in\mathcal{K}} \rho^{\kappa} = \arg\min_{\kappa\in\mathcal{K}} \lim_{T\to\infty} \rho^{\kappa}(T), \quad (14)$$

subject to the memory storage constraint (1).

The resulting problem is a stochastic sequential game, in which the service operator and the WDs play a multi-follower Stackelberg game in every time slot. In the Stackelberg game the service operator is the leader and the WDs are the followers. We refer to the problem as the *Dynamic Time Constrained Computation Offloading* (DTCCO) game, and we are interested in learning a policy κ^* that solves (14) under incomplete information, i.e., through interaction with the WDs. Importantly, we assume that the service operator does not know the set of active WDs and their parameters, instead WDs report their parameters to a network orchestrator node owned by the network operator and the orchestrator node coordinates the offloading decisions of the WDs. This assumption is reasonable when the network orchestrator and the service operator are different entities [10]–[12, 18]: the network operator can have access to traffic information and WDs' parameters [12], while the service operator owns or rents computation resources that it makes available to the WDs, but needs to decide caching and pricing before WDs decide whether to use its service.

III. Existence of Equilibria in the Stage Game

We first focus on the stage game played in time slot t among the active WDs, and we characterize their interaction for a given caching decision $\mathcal{X}(t)$ and price $\pi(t)$, chosen by the service operator. We then consider the problem of pricing and caching faced by the service operator under complete information when the service operator knows the set of active WDs and their parameters. This is equivalent to assuming that the network operator and the service operator are the same entity, hence in this section we use the term operator to refer to both entities. As we focus on a single stage, throughout the section we omit the time index to simplify notation.

A. Equilibrium Existence Among WDs

Let us consider a caching decision $\mathcal{X}$ and price π, and investigate the interaction of the active WDs, which is in effect a strategic game. We thus investigate whether the strategic game played by the WDs admits a pure strategy NE. To assess whether NE exist, we start with characterizing the optimal offloading decision for WDs.

Lemma 1. *Consider a WD i such that $\phi_i \in \mathcal{X}$. If $B_i = 0$ then WD i is not active, and thus $a_i = 0$. If $B_i = 1$, i.e., the WD is active, then if $\tau_i^c(m) > \tau_i^l$ then the optimal offloading decision is $a_i^* = 0$. Otherwise, let p_i^* be such that $\tau_i^u(p_i^*, m) + \tau_i^c(m) = \tau_i^l$. Then, if $p_i^* > \hat{p}_i$ then $a_i^* = 0$, otherwise*

$$a_i^* = \begin{cases} 1, & \pi \le \beta_i(f_i^l\gamma_i^l - p_i^*(\frac{1}{f_i^l} - \frac{m}{f^c})) \\ 0, & else, \end{cases} \quad (15)$$

where $\hat{p}_i$ is the maximum transmission power of WD i.

Proof. Observe that if $\tau_i^c(m) > \tau_i^l$ then WD i cannot complete the task on time, thus the optimal offloading decision is $a_i^* = 0$. Otherwise, WD i should choose a transmit power that minimizes its cost while ensuring timely completion. It is easy to see that the upload time $\tau_i^u(p_i, m)$ is a strictly monotonically decreasing function of p_i, and $C_i(1, p_i, a_{-i})$ is a strictly monotonically increasing function of p_i. Thus, WD i minimizes its cost by choosing a transmit power p_i^* that yields $\tau_i^u(p_i^*, m) + \tau_i^c(m) = \tau_i^l$. Now, if $p_i^* > \hat{p}_i$ then offloading is not feasible. Otherwise, if $p_i^* \le \hat{p}_i$ then the optimal decision is

$$a_i^* = \begin{cases} 1, & C_i(1, p_i^*, a_{-i}) \le C_i(0, p_i^*, a_{-i}) \\ 0, & else. \end{cases} \quad (16)$$

We can substitute $\tau_i^u(p_i^*, m) = \tau_i^l - \tau_i^c(m)$, (2) and (6) into (16), and obtain (15), which proves the result. □

The optimal offloading decision of a WDs given other WDs' decisions is called the best reply, and is used in characterizing NE, defined as follows.

Definition 1 (Nash Equilibrium). *A NE is a collection of offloading decisions $(a_i^*)_{i\in\mathcal{N}}$ such that*

$$C_i(a_i^*, p_i^*, a_{-i}^*) \le C_i(1 - a_i^*, p_i^*, a_{-i}^*), \forall i \in \mathcal{N}. \quad (17)$$

Observe that the game played between WDs is a player-specific network congestion game with the topology shown in Fig. 2. Unfortunately, in player-specific congestion games the existence of pure strategy NE is not guaranteed. Next, we use a topological equivalence argument to show that in the considered game a NE always exists.

Theorem 1. *The stage game possesses a pure strategy Nash equilibrium among WDs.*

Proof. Note that the stage game is a player-specific network congestion game with topology shown in Figure 2 (left). The nodes S, A, and D stand for *Source*, *Access Point*, and *Destination*, respectively. In the network topology the path (S, A, D) corresponds to computation offloading, while the direct path (S, D) corresponds to local computing with edge weights $C_i(1, p_i^*, a_{-i})$ and $C_i(0, p_i^*, a_{-i})$, respectively. To show the existence of equilibria, in what follows we show that Γ can be transformed to a network with parallel edges $\tilde{\Gamma}$ such that the games played on the two networks are best response equivalent. We do so by replacing the edge (A, D) and its two end vertices A and D in Γ by a single vertex, and by redefining the costs of incident edges. Thus, we obtain the parallel network topology $\tilde{\Gamma}$ shown in Fig. 2 (right), where the local computing cost is defined as $\tilde{C}_i(0, p_i^*, a_{-i}) = C_i(0, p_i^*, a_{-i}) - L_{\phi_i}D_i\pi$, and the cost of offloading is $\tilde{C}_i(1, p_i^*, a_{-i}) = C_i(1, p_i^*, a_{-i}) - L_{\phi_i}D_i\pi$.

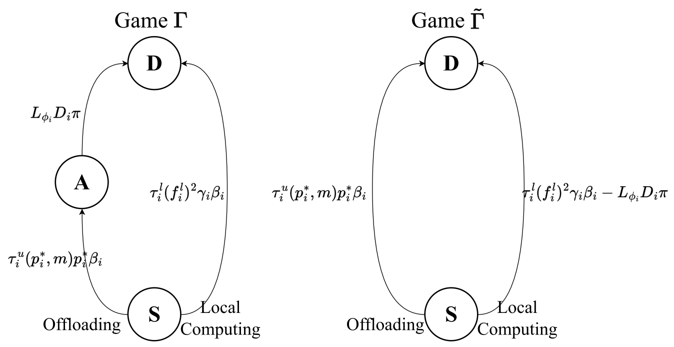
Game Γ
D
$L_{\phi_i} D_i \pi$
A
$\tau_i^l (f_i^l)^2 \gamma_i \beta_i$
$\tau_i^u(p_i^*, m) p_i^* \beta_i$
S
Offloading
Local Computing
Game $\tilde{\Gamma}$
D
$\tau_i^u(p_i^*, m) p_i^* \beta_i$
$\tau_i^l (f_i^l)^2 \gamma_i \beta_i - L_{\phi_i} D_i \pi$
S
Offloading
Local Computing

Fig. 2. Topology of the network congestion game Γ and $\tilde{\Gamma}$ used in the proof of Theorem 1.

Observe that the difference between the cost functions of WD i in Γ and that in $\tilde{\Gamma}$ depends only on the strategy of the operator. This in fact implies that $\tilde{\Gamma}$ and Γ are best-response equivalent, and thus they have identical sets of pure strategy Nash equilibria. Since $\tilde{\Gamma}$ is a singleton player-specific congestion game, it possesses a pure NE [19], and so does Γ. This concludes the proof. $\square$

Given that equilibria do exist, the next question is whether a NE can be computed easily. E.g., one could allow one WD at a time to improve its strategy, i.e., increase the payoff it receives, leading to an improvement path. If every improvement path is finite then the game is said to have the *Finite Improvement Property* (FIP), and a NE can be computed easily. Unfortunately, the FIP is not guaranteed in player-specific congestion games. Next, we show that the considered game does have the FIP.

Lemma 2. *The stage game possesses the finite improvement property, i.e., if WDs update their offloading strategies one at a time, they reach a NE in a finite number of steps.*

Proof. Each WD has two strategies, thus the result follows from Theorem 1 in [20]. $\square$

Hence, an equilibrium can be computed through letting WDs update their offloading strategies one at a time. We can thus conclude that for any caching decision $\mathcal{X}$ and price π set by the operator, there is a NE for the WDs in the stage game, and a NE can be computed efficiently. For a practical implementation, each WD can calculate its threshold price (15) for all $1 \leq m \leq N$, and hence the active WDs can find a NE by only sharing their threshold prices with a network orchestrator entity.

B. Optimal Pricing under Complete Information

Next, we consider the problem of the operator in the stage game, and we propose a polynomial-time algorithm for computing the optimal equilibrium price for a given caching decision $\mathcal{X}$. Throughout the subsection, we consider *Strong Stackelberg Equilibrium* (SSE), i.e., if there are multiple subgame perfect equilibria then one with maximum utility for the operator will be chosen. Throughout this subsection, we denote by $U(a, \mathcal{X}, \pi) = R(\mathcal{X}, \pi | \mathcal{N}^a)$ the instantaneous reward.

Let us denote by $\pi_{i,m}$ the maximum price at which WD i would choose to offload for a particular number of offloaders $m \leq N$, and let us call $\pi_{i,m}$ the threshold price of WD i for m. In addition, we define the notation that we will use in this section. Let us define the set $\mathcal{N}^o(\pi, m) = \{i | i \in \mathcal{N}, \pi_{i,m} \geq \pi\}$ of potential offloaders at price π if there were m offloaders, and define the set $\mathcal{N}_{\mathcal{X}} = \{i | i \in \mathcal{N}, \phi_i \in \mathcal{X}\}$ of WDs whose applications are cached by the operator. We then define the set $\Pi^t = \{\pi_{i,m} | i \in \mathcal{N}, 1 \leq m \leq N\}$ of threshold prices. We define corresponding sets for the set $\mathcal{X}$ of cached applications; we define $\Pi^t_{\mathcal{X}} = \{\pi_{i,m} | i \in \mathcal{N}_{\mathcal{X}}, 1 \leq m \leq N_{\mathcal{X}}\}$ as the set of threshold prices. We define the set $\mathcal{N}^o_{\mathcal{X}}(\pi, m) = \{i \in \mathcal{N}_{\mathcal{X}} | \pi_{i,m} \geq \pi\}$ of WDs that would want to offload at price π if a total of m WDs offload for cached application set $\mathcal{X}$. Under the complete information assumption the threshold prices $\pi_{i,m}$, $i \in \mathcal{N}$ can be calculated using Lemma 1. For an application placement $\mathcal{X}$ and price π we denote by $\alpha^*(\mathcal{X}, \pi)$ the set of Nash equilibria among WDs that yield maximum utility to the operator.

We continue with an important result that we will use for proposing a polynomial time algorithm that computes the utility maximizing price.

Lemma 3. *Consider an application placement $\mathcal{X}$ and threshold prices $\pi', \pi'' \in \Pi^t_{\mathcal{X}}$ such that there is no threshold price in the interval (π', π''), i.e., $(\pi', \pi'') \cap \Pi^t_{\mathcal{X}} = \emptyset$. Let $\pi_1, \pi_2 \in (\pi', \pi'']$, $\pi_1 < \pi_2$. Then the set of equilibria $\alpha^*(\mathcal{X}, \pi_1) = \alpha^*(\mathcal{X}, \pi_2)$. Furthermore, for any $a \in \alpha^*(\mathcal{X}, \pi_1)$ the utility of the operator is a monotonically increasing linear function on $(\pi', \pi'']$, i.e., $U(a, \mathcal{X}, \pi_1) < U(a, \mathcal{X}, \pi_2)$.*

Proof. We start with proving the first statement, i.e., $\alpha^*(\mathcal{X}, \pi_1) = \alpha^*(\mathcal{X}, \pi_2)$. Let $a \in \alpha^*(\mathcal{X}, \pi_1)$ be an equilibrium under price π_1. Now, since there is no threshold price on (π', π''), for any $\pi_2 \in (\pi_1, \pi'')$ it holds that $C_i(1, p_i^*, a_{-i}) \leq C_i(0, p_i^*, a_{-i})$ for π_1 if and only if $C_i(1, p_i^*, a_{-i}) \leq C_i(0, p_i^*, a_{-i})$ for π_2, $\forall i \in \mathcal{N}_{\mathcal{X}}$. Hence, $a \in \alpha^*(\mathcal{X}, \pi_2)$.

To prove the second statement, let us consider an equilibrium $a \in \alpha^*(\mathcal{X}, \pi'')$. By the previous statement we know that $a \in \alpha^*(\mathcal{X}, \pi)$ for $\pi \in (\pi', \pi'']$. We can rewrite (11) for the equilibrium strategy profile a under homogeneous pricing and obtain

$$U(a, \mathcal{X}, \pi) = \sum_{i \in \mathcal{N}: a_i = 1} \mathbb{1}_{\mathcal{X}}(\phi_i) L_{\phi_i} D_i \pi - \sum_{j \in \mathcal{J}} c_j, \qquad (18)$$

which is monotonically increasing in π for any given $\mathcal{X}$ on $(\pi', \pi'']$. For all $\pi \in (\pi', \pi'']$, WDs will not change their decisions. Therefore, we can treat $\tilde{C} = \sum_{i \in \mathcal{N}: a_i = 1} \mathbb{1}_{\mathcal{X}}(\phi_i) D_i L_{\phi_i}$ as a constant, and can substitute it in (18) to obtain

$$U(a, \mathcal{X}, \pi) = \tilde{C} \pi - \sum_{j \in \mathcal{J}} c_j. \qquad (19)$$

Since $\frac{\partial U(a, \mathcal{X}, \pi)}{\partial \pi} = \tilde{C}$, the utility of the operator is a linear function on $(\pi', \pi'']$. This concludes the proof. $\square$

The above result allow us to characterize the reward as a function of the price π set by the operator.

Proposition 1. *$U(a, \mathcal{X}, \pi)$ is a left-continuous piecewise-linear function of π.*

Proof. Consider prices $\pi', \pi'' \in \Pi^t_{\mathcal{X}}$ for some $\mathcal{X} \subseteq \mathcal{J}$ such that $(\pi', \pi'') \cap \Pi^t_{\mathcal{X}} = \emptyset$. Then, by Lemma 3, $U(a, \mathcal{X}, \pi)$ is an increasing affine function on the interval $(\pi', \pi'']$. Since the set $\Pi^t_{\mathcal{X}}$ has a finite number of elements, $U(a, \mathcal{X}, \pi)$ is a collection of left-continuous monotonically increasing linear functions, and it is thus piecewise linear. □

Next, we characterize equilibria to allow finding an optimal price efficiently.

Lemma 4. *Let $a', a'' \in \alpha^*(\mathcal{X}, \pi)$ be NE for application placement $\mathcal{X}$ and price π. Then $\sum_{i \in \mathcal{N}_{\mathcal{X}}} a'_i = \sum_{i \in \mathcal{N}_{\mathcal{X}}} a''_i \leq m$, i.e., the number of offloaders is the same in the NE.*

Proof. We prove $\sum_{i \in \mathcal{N}_{\mathcal{X}}} a'_i = \sum_{i \in \mathcal{N}_{\mathcal{X}}} a''_i$ by contradiction. Let $m' = \sum_{i \in \mathcal{N}_{\mathcal{X}}} a'_i$ and $m'' = \sum_{i \in \mathcal{N}_{\mathcal{X}}} a''_i$, and without loss of generality, assume that $m'' < m'$. Then, for strategy profile a', there has to be at least m' WDs with $\pi_{i,m'} \geq \pi$. Similarly, for NE strategy profile a'', there have to be at least m'' WDs with $\pi_{i,m''} \geq \pi$. Observe that $\pi_{i,m'} < \pi_{i,m''}$ since by assumption $m'' < m'$. However, if a' is a NE then we know that there are at least m' WDs for which $\pi_{i,m''} \geq \pi$. Thus in strategy profile a'' there are at least $m' - m''$ WDs that would prefer offloading at price π, and hence a'' cannot be a NE, which contradicts the initial assumption. Thus, $m' = m''$ must hold, which concludes the proof. □

We now use Lemma 4 for designing an algorithm for computing the NE. First, we show that for given π and application placement $\mathcal{X}$, a NE with maximum payoff for the operator can be computed in polynomial time. To show this, observe that for given price π, the operator's income from a WD that offloads is $U(a_i, \{\phi_i\}, \pi) = a_i \mathbb{1}_{\mathcal{X}}(\phi_i) L_{\phi_i} D_i \pi$, and is independent of what other WDs are offloading.

Lemma 5. *Consider a price π and application placement $\mathcal{X}$. Let $m' = \max_m \{|\mathcal{N}^o_{\mathcal{X}}(\pi, m)| \geq m\}$. Consider a set $\mathcal{N}^\dagger \subseteq \mathcal{N}$, $\mathcal{N}^o_{\mathcal{X}}(\pi, m'+1) \subseteq \mathcal{N}^\dagger \subseteq \mathcal{N}^o_{\mathcal{X}}(\pi, m')$ such that $|\mathcal{N}^\dagger| = m'$ and $\sum_{i \in \mathcal{N}^\dagger \setminus \mathcal{N}^o_{\mathcal{X}}(\pi, m'+1)} L_{\phi_i}$ is maximal. Then the strategy profile a in which $a_i = 1 \iff i \in \mathcal{N}^\dagger$ is a NE with maximum payoff for the operator, and can be found in polynomial time.*

Proof. Consider $m' = \max_m \{|\mathcal{N}^o_{\mathcal{X}}(\pi, m)| \geq m\}$, and observe that $\mathcal{N}^o_{\mathcal{X}}(\pi, m'+1) \subset \mathcal{N}^o_{\mathcal{X}}(\pi, m')$. Since $|\mathcal{N}^o_{\mathcal{X}}(\pi, m'+1)| = m'' < m'+1$ in any NE $\sum_{i \in \mathcal{N}_{\mathcal{X}}} a_i < m'+1$, at the same time there is at least one NE in which $\sum_{i \in \mathcal{N}_{\mathcal{X}}} a_i = m'$, thus by Lemma 4 we know that $\sum_{i \in \mathcal{N}_{\mathcal{X}}} a_i = m'$ for any $a \in \alpha^*(\mathcal{X}, \pi)$. Clearly in a NE with m' offloaders $a_i = 1$ for WDs $i \in \mathcal{N}^o_{\mathcal{X}}(\pi, m'+1)$, and hence there are $\binom{|\mathcal{N}^o_{\mathcal{X}}(\pi, m')| - m''}{m' - m''}$ equilibria that can be computed, with potentially different payoffs for the operator. To find an equilibrium with highest payoff for the operator, recall that the income $U(a_i, \{\phi_i\}, \pi)$ from WD i offloading is independent of what other WDs offload. Hence the set $\mathcal{N}^\dagger$ of offloaders that maximizes the income of the operator is such that $\mathcal{N}^o_{\mathcal{X}}(\pi, m'+1) \subseteq \mathcal{N}^\dagger$, and it contains the WDs with highest $L_{\phi_i} D_i$ from the set $\mathcal{N}^o_{\mathcal{X}}(\pi, m') \setminus \mathcal{N}^o_{\mathcal{X}}(\pi, m'+1)$.

Algorithm 1: Computing a NE for WDs

```
Data: $\mathcal{X}, \pi$  Result: $\mathcal{N}^\dagger$
1 $m' = \max_m \{|\mathcal{N}^o_{\mathcal{X}}(\pi, m)| \geq m\}$
2 $\mathcal{N}^\dagger = \emptyset$
3 if $m' < |\mathcal{N}_{\mathcal{X}}|$ then
4     $\mathcal{N}^\dagger = \mathcal{N}^o_{\mathcal{X}}(\pi, m'+1)$
5 end
6 $\mathcal{N}^\dagger$.add($\{|\arg\max \sum_{i \in \mathcal{N}^o_{\mathcal{X}}(\pi, m') \setminus \mathcal{N}^o_{\mathcal{X}}(\pi, m'+1)} L_{\phi_i} D_i| = m' - |\mathcal{N}^\dagger|\}$)
```

Algorithm 2: Calculating optimal price for given $\mathcal{X}$

```
Data: $\mathcal{X}, \Pi^t_{\mathcal{X}}$  Result: $\pi^*, U^*$
/* Calculate the operator's revenue
   for each $\pi_{i,m}$                    */
1 for $k' = 1 : |\Pi^t_{\mathcal{X}}|$ do
2     $\mathcal{N}^\dagger =$ Algorithm1($\mathcal{X}, \Pi^t_{\mathcal{X}}(k')$)
3     $U(k') = \sum_{i \in \mathcal{N}^\dagger} L_{\phi_i} D_i \Pi^t_{\mathcal{X}}(k')$ - $\sum_{j \in \mathcal{X}} c_j$
4 end
5 $U^* = \max_k U(k')$
6 $k^* = \min\{k' | U(k') = \pi^*\}, \pi^* = \Pi^t_{\mathcal{X}}(k'^*)$
```

To see that the solution can be obtained in polynomial time, observe that m' can be found based on $\mathcal{N}^o_{\mathcal{X}}(\pi, m)$, and $\mathcal{N}^\dagger$ can be found by sorting WDs in decreasing order of $L_{\phi_i} D_i$, both in polynomial time (see Algorithm 1). □

Lemma 5 allows us to compute a NE among the WDs efficiently, one that is in accordance with the SSE assumption, i.e., it maximizes the revenue of the service operator. Given a NE for a given price, we are now ready to compute the price that maximizes the operator's revenue for given application placement.

Theorem 2. *Consider an application placement $\mathcal{X}$. Then the price π^* computed by Algorithm 2 maximizes the operator's revenue, i.e., $U(a^*, \mathcal{X}, \pi^*) \geq U(a, \mathcal{X}, \pi), \forall a^* \in \alpha^*(\mathcal{X}, \pi^*)$, $(\pi, a) \in \alpha^*(\mathcal{X}, \pi)$.*

Proof. Consider a price such that $\pi \notin \Pi^t_{\mathcal{X}}$, and allows a set of equilibria $\alpha^*(\mathcal{X}, \pi)$. Then, for any consecutive threshold prices (i.e. $\pi', \pi'' \in \Pi^t_{\mathcal{X}}$, $(\pi', \pi'') \cap \Pi^t_{\mathcal{X}} = \emptyset$), let π be such that $\pi' > \pi > \pi''$. By Lemma 3 the two prices allow the same set of equilibria, $a \in \alpha^*(\mathcal{X}, \pi') = \alpha^*(\mathcal{X}, \pi)$, and the utilities satisfy $U(a, \mathcal{X}, \pi') > U(a, \mathcal{X}, \pi)$. Thus, to be able to find the profit maximizing price and the corresponding strategy profile, it is sufficient to compute $U(a, \mathcal{X}, \pi'), \forall \pi' \in \Pi^t_{\mathcal{X}}$ and then find the price such that $\pi^* = \arg\max_{\pi' \in \Pi^t_{\mathcal{X}}} U(a, \mathcal{X}, \pi')$. Hence, Algorithm 2 computes a price that maximizes the utility of the operator. □

Theorem 2 implies that the optimal price π^* is a threshold price for the given caching decision $\mathcal{X}$, i.e., $\pi^* \in \Pi^t_{\mathcal{X}}$, and the operator can find the optimal price by calculating and comparing the utility at the threshold prices. Thus, an optimal price can be computed in polynomial time for any given caching decision $\mathcal{X}$ under complete information.

IV. Bayesian optimization for regret minimization

We have so far shown how to choose an optimal price for a caching decision $\mathcal{X}$ under complete information and characterized the reward as a function of the price. In what follows we consider the incomplete information case, i.e., the network operator and the service operator are different entities. We characterize the expected reward of the service operator under incomplete information and we propose an online policy for maximizing it, including the computation of a near-optimal caching decision $\mathcal{X}^*$ together with a corresponding optimal price, which together approximate a solution to (14).

A. Characterization of the per stage expected reward

We first start with characterizing the expected reward as a function of the price $\pi(t)$ chosen by the service operator for a single time slot t. For brevity, we omit the time index t in this subsection.

Proposition 2. *$\bar{r}(\mathcal{X}, \pi)$ is a piecewise linear, left-continuous function of π.*

Proof. Recall that $\bar{r}(\mathcal{X}, \pi)$ is by definition the expectation of the reward $R(\mathcal{X}, \pi|\mathcal{N}^a) = U(a, \mathcal{X}, \pi)$, where the expectation is taken over the set $\mathcal{N}^a$ of active WDs. Thus, by Proposition 1 it is the weighted sum of piecewise linear left-continuous functions, and is thus itself piecewise linear and left-continuous. □

Thus, for any $\mathcal{X} \subseteq \mathcal{J}$ there is a price $\pi^*_{\mathcal{X}} \in \arg\max_{\pi} \bar{r}(\mathcal{X}, \pi)$. We now continue with the analysis of the maximal expected reward, starting with the definition of two properties of reward functions.

Definition 2. *The set function $\bar{r} : 2^{\mathcal{J}} \to \mathbb{R}$ is monotone if for any $\mathcal{X} \subset \mathcal{J}$ and $j \in \mathcal{J} \setminus \mathcal{X}$ we have $\bar{r}(\mathcal{X} \cup \{j\}) \geq \bar{r}(\mathcal{X})$.*

Definition 3. *Given scalar $0 < \nu \leq 1$, we say that the set function $\bar{r} : 2^{\mathcal{J}} \to \mathbb{R}$ is ν-weakly submodular if*

$$\sum_{j \in \mathcal{X}'} \Big(\bar{r}(\mathcal{X} \cup \{j\}) - \bar{r}(\mathcal{X})\Big) \geq \min\{\nu(\bar{r}(\mathcal{X} \cup \mathcal{X}') - \bar{r}(\mathcal{X})), \frac{1}{\nu}(\bar{r}(\mathcal{X} \cup \mathcal{X}') - \bar{r}(\mathcal{X}))\}, \quad (20)$$

where $\mathcal{X}, \mathcal{X}' \subseteq \mathcal{J}$, $\mathcal{X} \cap \mathcal{X}' = \emptyset$.

Monotonicity is a common assumption, e.g., in Knapsack problems with independent item values. As we show next, in the considered problem the service operator's expected reward need not be monotone.

Proposition 3. *Let $\mathcal{X} \subseteq \mathcal{J}$ and $j \in \mathcal{J} \setminus \mathcal{X}$, then*

$$\bar{r}(\mathcal{X} \cup \{j\}, \pi^*_{\mathcal{X} \cup \{j\}}) - \bar{r}(\mathcal{X}, \pi^*_{\mathcal{X}}) \lesseqgtr 0. \quad (21)$$

Proof. We prove the result through the following example.

Example 1. *The WDs' parameters are as shown in Table II. The system-wide and application parameters are $|\mathcal{J}| = 2$, $L_j = (517, 820)$, $c_j = (0.025, 0.019)$, $\gamma_i^l = 10^{-18}$, $\beta_i = 1$, $f^c = 12$ GHz, $W = 240$ MHz. The resulting expected rewards are $\bar{r}(\{1,2\}, 0.24\times10^{-9}) = 5.49$, $\bar{r}(\{1\}, 0.34\times10^{-9}) = 0.15$, $\bar{r}(\{2\}, 0.23\times10^{-9}) = 5.52$, clearly $\bar{r}(\{1\}, 0.34\times10^{-9}) < \bar{r}(\{1,2\}, 0.24\times10^{-9}) < \bar{r}(\{2\}, 0.23\times10^{-9})$.*

Hence, the expected reward is not monotone. □

Together with monotonicity, submodularity is often used for obtaining approximation ratio bounds for NP-hard optimization problems. As we show, in our considered problem the expected reward need not be submodular, as it is not even weakly submodular.

Proposition 4. *Let $\mathcal{X}, \mathcal{X}' \subseteq \mathcal{J}$, $\mathcal{X} \cap \mathcal{X}' = \emptyset$. Then,*

$$\sum_{j \in \mathcal{X}'} \Big(\bar{r}(\mathcal{X} \cup \{j\}) - \bar{r}(\mathcal{X})\Big) < \min\{\nu(\bar{r}(\mathcal{X} \cup \mathcal{X}') - \bar{r}(\mathcal{X})), \frac{1}{\nu}(\bar{r}(\mathcal{X} \cup \mathcal{X}') - \bar{r}(\mathcal{X}))\}, \quad (22)$$

for any $\nu \in (0, 1]$.

Proof. We prove the statement by giving a counterexample for ν-weak submodularity.

Example 2. *The WDs' parameters are as shown in Table III. The system-wide and application parameters are $|\mathcal{J}| = 3$, $L_j = (530, 630, 1039)$, $c_j = (0.0986, 0.09, 0, 0982)$, $\gamma_i^l = 10^{-18}$,$\beta_i = 1$,$f^c = 12$ GHz, $W = 280$ MHz. Assume that $\mathcal{X}' = \{1, 2\}$, and $\mathcal{X} = \{3\}$. The expected reward for this experiment are $\bar{r}(\{3,1\}, \pi^*_{\{3,1\}}) = 7.3689$, $\bar{r}(\{3,2\}, \pi^*_{\{3,2\}}) = 7.3604$, $\bar{r}(\{3\}, \pi^*_{\{3\}}) = 7.459$, $\bar{r}(\{1,2,3\}, \pi^*_{\{1,2,3\}}) = 7.6142$. Using (20), $\bar{r}(\{3,1\}, \pi^*_{\{3,1\}}) + \bar{r}(\{3,2\}, \pi^*_{\{3,2\}}) - 2\bar{r}(\{3\}, \pi^*_{\{3\}}) = -0.1887$, and $\bar{r}(\{1,2,3\}, \pi^*_{\{1,2,3\}}) - \bar{r}(\{3\}, \pi^*_{\{3\}}) = 0.1551$.*

The left hand side of the inequality (20) is negative, whereas the right hand side is positive. The inequality does not hold, hence function is not weakly submodular in general. □

We have thus shown that the expected reward $\bar{r}$ is neither monotone nor weakly submodular in general. Hence, existing results on monotone submodular function maximization do not hold for our problem.

Moreover, our analysis of the expected reward highlights two key challenges in learning an optimal policy κ^* based on past observations, challenges not found in the literature on *Combinatorial Multi-Armed Bandit* (CMAB) optimization [21, 22]. First, the expected reward of caching an application depends on what other applications are cached, i.e., in bandit terminology, the expected reward of a bandit arm is not independent of the set of arms chosen. Second, the rewards of the cached apps $\mathcal{X}$ depend on the chosen price π, i.e., there is an additional continuous decision variable that needs to be optimized.

WD	f_i^l	$\tilde{\sigma}_i^2$	D_i	$\hat{p}_i$	h_i	q_i	ϕ_i
WD1	0.23	0.2078	44	0.1	0.93	0.67	2
WD2	0.34	0.184	0.42	0.176	0.84	0.42	1

TABLE II
WDs' parameters for Example 1

WD	f_i^l	$\tilde{\sigma}_i^2$	D_i	$\hat{p}_i$	h_i	q_i	ϕ_i
WD1	0.595	0.161	13.65	0.32	0.96	0.52	3
WD2	0.288	0.127	47.5	0.26	0.95	0.37	2
WD3	0.506	0.106	24.6	0.29	0.96	0.3	3
WD4	0.452	0.129	3.34	0.23	0.87	0.9	1

TABLE III
WDs' parameters for Example 2

As a consequence, existing approaches for solving CMAB problems, which choose a set of arms (called a super arm) using a computation oracle that is provided with the empirical distribution of the rewards of individual arms, can not be applied directly to our problem. Instead, the choice of what set of applications to cache has to be combined with learning the corresponding optimal price.

B. Combinatorial Bayesian Revenue Maximization

Motivated by the above observations, we propose the BRM algorithm for approximating an optimal policy. BRM combines the exploration of the expected reward of individual applications with the maximization of the reward of a set of applications that are expected to provide the highest reward, computed based on current best estimates. Importantly, the optimization of the price is specific to the set of cached applications, so as to address the issue of potential non-monotonicity. The pseudocode of the algorithm is shown in Algorithm 3.

The key tenet of BRM is that it simultaneously learns to approximate the maximum expected reward of individual applications and sets of applications. With a small, decreasing probability, at time t it caches a single application chosen at random, while otherwise, it selects a set of applications to cache based on their estimated maximum expected rewards, computed using the posterior mean reward of the applications obtained using a GP approximation. For the chosen set $\mathcal{X}$ of applications, it then chooses a price π based on samples of the instantaneous rewards collected in the past, for that set of applications, using a GP approximation of the expected reward function (Line 11).

For a given application placement $\mathcal{X}$, the function that we want to maximize is one dimensional, i.e., $\bar{r}(\mathcal{X},\pi):\mathcal{P}\rightarrow\mathbb{R}$, and we propose to approximate it by a GP using *Bayesian Optimization* (BO). Let us denote by $\mathcal{D}_t=\{(\mathcal{X}(l),\pi(l),R(\mathcal{X}(l),\pi(l))\}_{l=1}^t$ the set of reward samples collected up to time t, and by $\mathcal{D}_t^{\mathcal{X}}=\{(\mathcal{X},\pi(l),R(\mathcal{X},\pi(l))|l=1,\dots,t;\mathcal{X}(l)=\mathcal{X}\}$ the reward samples collected for a set $\mathcal{X}$ of applications up to time t. Let $n_t^{\mathcal{X}}=|\mathcal{D}_t^{\mathcal{X}}|$ be the number of reward samples collected for set $\mathcal{X}$, and let $\pi_{\mathcal{X}}(l)$ be the price used when the target function of set $\mathcal{X}$ was sampled the l^{th} time, $1\leq l\leq n_t^{\mathcal{X}}$, and denote by $P_t^{\mathcal{X}}=[\pi_{\mathcal{X}}(1),\dots,\pi_{\mathcal{X}}(n_t^{\mathcal{X}})]$ the vector of prices used when the target function of set $\mathcal{X}$ is sampled.

For a set of applications $\mathcal{X}$ at time t the GP approximation of the expected reward function $\bar{r}(\mathcal{X},\pi(t))$ as a function of the price models the expected reward as a collection of random variables $\{\bar{r}(\mathcal{X},\pi)\}_{\pi\in\mathcal{P}}$, such that the finite collection of random variables $\{\bar{r}(\mathcal{X},\pi_{\mathcal{X}}(l))\}_{l\leq n_t^{\mathcal{X}}}$ are jointly Gaussian with mean

$$E[\bar{r}(\mathcal{X},\pi_{\mathcal{X}}(l))]=\mu^{\mathcal{X}}(\pi_{\mathcal{X}}(l)), \tag{23}$$

and covariance

$$\begin{aligned}&cov(\bar{r}(\mathcal{X},\pi_{\mathcal{X}}(l)),\bar{r}(\mathcal{X},\pi_{\mathcal{X}}(l')))= \qquad (24)\\ &E\Big[\Big(\bar{r}(\mathcal{X},\pi_{\mathcal{X}}(l))-\mu^{\mathcal{X}}(\pi_{\mathcal{X}}(l))\Big)\Big(\bar{r}(\mathcal{X},\pi_{\mathcal{X}}(l'))-\mu^{\mathcal{X}}(\pi_{\mathcal{X}}(l'))\Big)\Big]\\ &=k^{\mathcal{X}}(\pi_{\mathcal{X}}(l),\pi_{\mathcal{X}}(l'))\leq 1,\end{aligned}$$

for all $l,l'\leq n_t^{\mathcal{X}}$, where $k^{\mathcal{X}}$ is called the kernel function. An example of commonly used kernel functions is the squared exponential kernel $k(\pi,\pi')=e^{-\frac{||\pi-\pi'||^2}{2\theta^2}}$, where θ is called the length scale parameter. Let us denote by $\boldsymbol{y}_t^{\mathcal{X}}=[R(\mathcal{X},\pi_{\mathcal{X}}(1)),\dots,R(\mathcal{X},\pi_{\mathcal{X}}(n_t^{\mathcal{X}}))]^T$ the vector of revenue samples collected until time t. Then, the posterior distribution of the GPs approximation of the expected reward with zero mean prior (i.e., $GP(0,k^{\mathcal{X}}(.,.))$ will have mean $\mu_t^{\mathcal{X}}(\pi)$, covariance $k_t^{\mathcal{X}}(\pi,\pi')$ and variance $\sigma_t^{\mathcal{X}}(\pi)$ that can be computed as [23]

$$\mu_t^{\mathcal{X}}(\pi)=\boldsymbol{k}_t^{\mathcal{X}}(\pi)^T(\boldsymbol{K}_t^{\mathcal{X}}+\bar{\sigma}^2\boldsymbol{I})^{-1}\boldsymbol{y}_t^{\mathcal{X}} \tag{25}$$

$$k_t^{\mathcal{X}}(\pi,\pi')=k^{\mathcal{X}}(\pi,\pi')-\boldsymbol{k}_t^{\mathcal{X}}(\pi)^T(\boldsymbol{K}_t^{\mathcal{X}}+\bar{\sigma}^2\boldsymbol{I})^{-1}\boldsymbol{k}_t^{\mathcal{X}}(\pi') \tag{26}$$

$$\sigma_t^{\mathcal{X}}(\pi)=\sqrt{k_t^{\mathcal{X}}(\pi,\pi)}, \tag{27}$$

where $\boldsymbol{k}_t^{\mathcal{X}}(\pi)=[k^{\mathcal{X}}(\pi,\pi_{\mathcal{X}}(1)),\dots,k^{\mathcal{X}}(\pi,\pi_{\mathcal{X}}(n_t^{\mathcal{X}}))]^T\in\mathbb{R}^{n_t^{\mathcal{X}}}$, $\boldsymbol{K}_t^{\mathcal{X}}=\big(k^{\mathcal{X}}(\pi_{\mathcal{X}}(l),\pi_{\mathcal{X}}(l'))\big)_{l,l'\leq n_t^{\mathcal{X}}}\in\mathbb{R}^{n_t^{\mathcal{X}}\times n_t^{\mathcal{X}}}$ is the positive semi-definite kernel matrix, $\boldsymbol{I}$ is the $n_t^{\mathcal{X}}\times n_t^{\mathcal{X}}$ identity matrix, and $\bar{\sigma}^2$ is the prior of the noise variance.

Given the posterior distribution $GP(\mu_t^{\mathcal{X}},k_t^{\mathcal{X}})$, the algorithm chooses the next price $\pi_{\mathcal{X}}(n_t^{\mathcal{X}}+1)$ to be explored so as to maximize the upper-confidence bound of the expected reward, which is computed based on $\mu_t^{\mathcal{X}}$ and $\sigma_t^{\mathcal{X}}$ (Line 11). Intuitively, maximization of the upper confidence bound aims at finding a tradeoff between maximizing the instantaneous reward based on past samples and between exploring prices for which the estimated reward has high variance. BRM is inspired by the recently proposed *Enlarged Confidence Gaussian Process Upper Confidence Bound* (EC-GP-UCB) algorithm [24], but compared to EC-GP-UCB it uses a novel acquisition function that results in a deterministic regret bound. In addition, it extends the GP approximation to the selection of cached objects, performed using the GP-*Non-negative Greedy* (NNG) algorithm, which also makes use of the estimated posterior means.

C. Regret Analysis

In this section we provide a bound on the regret achieved by the proposed algorithm. We consider a particular set $\mathcal{X}$ of cached apps throughout the subsection, i.e., we provide a bound on the regret

$$\rho_{\mathcal{X}}(T)=\sum_{t=1}^{T}\Big(\max_{\pi\in\mathcal{P}}\bar{r}(\mathcal{X},\pi)-\bar{r}(\mathcal{X},\pi(t))\Big) \tag{28}$$

and hence we omit $\mathcal{X}$, and we use t instead of $n_t^{\mathcal{X}}$ for simplicity. We start with introducing *Reproducing Kernel Hilbert*

Algorithm 3: BRM

```
Result: $\hat{r}$
1  $t = 1, \underline{r} = 0, \mathcal{D}_0 = \emptyset$
2  for $t = 1, 2, \dots$ do
3      $\theta(t) \sim Bernoulli(t^{-\xi})$
4      if $\theta(t) = 1$ then
5          $\hat{\mathcal{X}} = \{\text{Unif}(\mathcal{J})\}$ /* Choose $j \in \mathcal{J}$
               uniform at random */
6      else
7          $\hat{\mathcal{X}} = \text{GP-NNG}(\{\mu_{t-1}^{\{j\}}\}_{j \in \mathcal{J}})$
8      end
       /* GP approximation for $\hat{\mathcal{X}}$ between
          Line 10-12 */
9      $\mathcal{D}_{t-1}^{\hat{\mathcal{X}}} = \{(\mathcal{X}, \pi, R) \in \mathcal{D}_{t-1} | \mathcal{X} = \hat{\mathcal{X}}\}$
10     $\psi_{t-1}^{\hat{\mathcal{X}}} = \frac{\epsilon^{max}\sqrt{n_{t-1}^{\hat{\mathcal{X}}}}}{\bar{\sigma}} + V$
11     $\hat{\pi}_{\hat{\mathcal{X}}} \in$
         $\arg\max_{\pi \in \mathcal{P}} \mu_{t-1}^{\hat{\mathcal{X}}}(\pi) + \left(\psi_{t-1}^{\hat{\mathcal{X}}} + \frac{\Delta\sqrt{n_{t-1}^{\hat{\mathcal{X}}}}}{\bar{\sigma}}\right)\sigma_{t-1}^{\hat{\mathcal{X}}}(\pi)$
       /* Observe $R(\hat{\mathcal{X}}, \hat{\pi}_{\hat{\mathcal{X}}})$ */
12     $\mathcal{D}_t^{\hat{\mathcal{X}}} = \mathcal{D}_{t-1}^{\hat{\mathcal{X}}} \cup \{(\hat{\mathcal{X}}, \hat{\pi}_{\hat{\mathcal{X}}}, R(\hat{\mathcal{X}}, \hat{\pi}_{\hat{\mathcal{X}}}))\}$
       /* Compute $\mu_t^{\hat{\mathcal{X}}}, \sigma_t^{\hat{\mathcal{X}}}$ using (25)–(27) */
13     $\underline{r} = (\underline{r} + R(\hat{\mathcal{X}}, \hat{\pi}_{\hat{\mathcal{X}}}))$
14     $\hat{r} = \underline{r}/t$
15 end
```

Algorithm 4: GP-NNG Algorithm

```
Data: $\{\mu^{\{j\}}\}_{j \in \mathcal{J}}$ Result: $\mathcal{X}$
1  $\mathcal{X} = \emptyset$
2  $M_j = \max_\pi \mu^{\{j\}}(\pi)$, $M := \{M_j\}_{j \in \mathcal{J}}$
3  while $M \neq \emptyset \wedge \sum_{j \in \mathcal{X}} s_j \leq S$ do
4      $j^* = \arg\max_j M_j$ /* Ties are broken
           arbitrarily */
5      if $\max_\pi \sum_{j \in \mathcal{X} \cup \{j^*\}} \mu^{\{j\}}(\pi) \geq$
         $\max_\pi \sum_{j \in \mathcal{X}} \mu^{\{j\}}(\pi)$ then
6          $\mathcal{X} = \mathcal{X} \cup \{j^*\}$
7      end
8      $M = M \setminus M_{j^*}$
9  end
```

*Space*s (RKHSs), which is a central concept in the study of BO using GPs.

Definition 4 (**Hilbert Space**). *A Hilbert Space is an inner product space that is complete with respect to the norm induced by the inner product (Ex. $\mathbb{R}^d, d \in \mathbb{N}^+$).*

Definition 5 (**Reproducing Kernel Hilbert Space**). *Let $\mathcal{L} \neq \emptyset$ and $\mathcal{H}_k$ be a Hilbert function space over $\mathcal{L}$ that consists of functions $f : \mathcal{L} \to \mathbb{R}$.*

- *A function $k : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ is called a reproducing kernel of $\mathcal{H}_k$ if we have $k(.,l) \in \mathcal{H}_k, \forall l \in \mathcal{L}$ and the reproducing property*

$$f(l) = \langle f, k(.,l) \rangle \tag{29}$$

holds $\forall f \in \mathcal{H}_k$ and $\forall l \in \mathcal{L}$

- *The space $\mathcal{H}_k$ is called a RKHS over $\mathcal{L}$ if $\forall l \in \mathcal{L}$ the Dirac functional $\delta_l : \mathcal{H}_k \to \mathbb{R}$ defined by*

$$\delta_l(f) := f(l), f \in \mathcal{H}_k \tag{30}$$

is continuous.

Many recent works in BO obtained regret bounds for functions that belong to some RKHS with a continuous kernel [25], [24], [26], [27]. By using a RKHS with a *universal* continuous kernel one can uniformly approximate any continuous bounded function on a compact domain [28]. There are well known zero-regret optimization algorithms for objective functions that are inside of RKHS $\mathcal{H}_k$ spanned by a given continuous kernel k [25, 26]. By Lemma 4.28 in [29], we know that if $\mathcal{H}_k$ is an RKHS generated by kernel k, then k is bounded and separately continuous if and only if $\forall f \in \mathcal{H}_k$ is bounded and continuous. Nonetheless, by Proposition 2 we know that $\bar{r}$ is not continuous, and hence $\bar{r} \notin \mathcal{H}_k$ for any continuous kernel k. Consequently, achieving zero-regret is infeasible.

Hence, our regret bound is based on first bounding the regret that we could achieve if the target function was in a suitably chosen RKHS, and, one that is known to contain functions that are not too far from our target function $\bar{r}$. To make this precise, for a kernel k let us denote by $\mathcal{H}_k(\mathcal{P})$ the RKHS of well-behaved functions over the domain $\mathcal{P}$ formed by k. Let us define the class of functions with bounded RKHS norm

$$\mathcal{F}_k(\mathcal{P}, V) = \{\bar{r}^h \in \mathcal{H}_k(\mathcal{P}) : ||\bar{r}^h||_k \leq V\}, \tag{31}$$

i.e., the RKHS norm $||\bar{r}^h||_k = \sqrt{\langle \bar{r}^h, \bar{r}^h \rangle_k}$ is bounded by $V > 0$, where $\langle ., . \rangle$ is the inner product forming $\mathcal{H}_k$. Furthermore let us choose a class of functions $\mathcal{F}_k(\mathcal{P}, V)$ so that there exists a $\Delta > 0$ such that

$$\min_{\bar{r}^h \in \mathcal{F}_k(\mathcal{P}, V)} ||\bar{r}^h - \bar{r}||_\infty \leq \Delta, \tag{32}$$

and there is a function

$$\tilde{r} \in \operatorname*{arg\,min}_{\bar{r}^h \in \mathcal{F}_k(\mathcal{P}, V)} ||\bar{r}^h - \bar{r}||_\infty, \tag{33}$$

such that $||\tilde{r} - \bar{r}||_\infty \leq \Delta$, where the norm is the maximum pointwise difference over $\mathcal{P}$. We refer to $\mathcal{F}_k(\mathcal{P}, V)$ as the hypothesis class and to $\tilde{r}$ as the hypothesis function, and we note that by definition $|\tilde{r}(\pi) - \bar{r}(\pi)| \leq \Delta, \forall \pi \in \mathcal{P}$.

We will start with bounding the regret in the hypothetical scenario that our observations are taken from the hypothesis function, defined as

$$\tilde{\rho}(T) = \sum_{t=1}^{T} \left(\max_{\pi \in \mathcal{P}} \tilde{r}(\pi) - \tilde{r}(\pi(t)) \right). \tag{34}$$

To continue the analysis, let us first recall that we can only take noisy observations of the target function $\bar{r}$, i.e., $y_{t'} = R(\pi(t')) = \bar{r}(\pi(t')) + \epsilon(\pi(t')), 1 \leq t' \leq t$. By (32) and (33) we know that there is a function $v(\pi)$ such that

$$\bar{r}(\pi) = \tilde{r}(\pi) + v(\pi), \; v(\pi) \in [-\Delta, \Delta], \tag{35}$$

and hence we can express the observations as

$$y_{t'} = \tilde{r}(\pi(t')) + v(\pi(t')) + \epsilon(\pi(t')), 1 \leq t' \leq t \tag{36}$$

$$y_{t'}^h = y_{t'} - v(\pi(t')), \tag{37}$$

where $y_{t'}^h$ is the hypothetical noisy observations at time slot t' if we had sampled from the hypothesis function, and $\epsilon(\pi(t))$ is the observation noise.

As a first step we show that the observation noise $\epsilon(\pi(t))$ is σ-sub-Gaussian, defined as follows.

Definition 6 (Sub-Gaussian Distribution). *Let $\sigma > 0$. We say that the random variable X is σ-sub-Gaussian if*

$$E\Big[\exp\Big(\mu(X - E[X])\Big)\Big] \leq \exp\Big(\frac{\sigma^2\mu^2}{2}\Big) \tag{38}$$

for any $\mu \in \mathbb{R}$. σ^2 is called the variance proxy.

Lemma 6. *The observation noise $\epsilon(\pi(t))$ is σ-sub-Gaussian with proxy $\sigma^2 = (\epsilon^{min} - \epsilon^{max}))^2/4$, where $\epsilon^{min} = \min_{\pi(t)\in\mathcal{P}} \epsilon(\pi(t))$, $\epsilon^{max} = \max_{\pi(t)\in\mathcal{P}} \epsilon(\pi(t))$.*

Proof. Observe that $E[\epsilon(\pi(t))] = E[R(\pi(t)) - \bar{r}(\pi(t))] = 0$ for any $\pi(t) \in \mathcal{P}$, and $\epsilon^{min} \leq \epsilon(\pi(t)) \leq \epsilon^{max}$. By using Hoeffding's Lemma we obtain

$$E[e^{\mu\epsilon(\pi(t))}] \leq \exp\Big(\mu^2(\epsilon^{max} - \epsilon^{min})^2/8\Big), \tag{39}$$

and hence for $\sigma^2 = (\epsilon^{max} - \epsilon^{min})^2/4$ the observation noise $\epsilon(\pi(t))$ is σ-sub-Gaussian by Definition 6. □

Let us first assume that we have a target function $\bar{r}^h$ that is inside the hypothesis class (we will remove this assumption later), i.e., $\bar{r}^h \in \mathcal{F}_k(\mathcal{P}, V)$, we can then bound the pointwise error between the posterior mean μ_t^h computed using (25) at time slot t and the target function $\bar{r}^h$ as a function of the variance σ_t.

Proposition 5. *Let $\bar{r}^h \in \mathcal{F}_k(\mathcal{P}, V)$ and let $\boldsymbol{y}_t$ be the vector of noisy observations that corresponds to sampled prices P_t. Then for $t \geq 1$ and $\pi \in \mathcal{P}$ we have*

$$|\bar{r}^h(\pi) - \mu_t^h(\pi)| \leq \psi_t\sigma_t(\pi), \tag{40}$$

where $\psi_t = \frac{\epsilon^{max}\sqrt{t}}{\bar{\sigma}} + V$.

Proof. The proof can be found in the Appendix. □

In reality, our target function is not an element of the hypothesis class (i.e. $\bar{r} \notin \mathcal{F}_k(\mathcal{P}, V)$). Nonetheless, the only difference between μ_t and μ_t^h comes from the observation vectors $\boldsymbol{y}_t$ and $\boldsymbol{y}_t^h$ respectively. We next provide a bound on the absolute difference between the posterior means obtained from querying the true target function and the best-in-class hypothesis function. Observe that the posterior standard deviation (26) does not depend on the observations.

Lemma 7. *For any $\pi \in \mathcal{P}$, $t \geq 1$ and $\bar{\sigma} > 0$, we have*

$$|\mu_t^h(\pi) - \mu_t(\pi)| \leq \Big(\frac{\Delta\sqrt{t}}{\bar{\sigma}}\Big)\sigma_t(\pi). \tag{41}$$

Proof. The proof can be found in the Appendix. □

Combining Proposition 5 and Lemma 7 we can bound the error of the posterior mean with respect to the hypothesis function.

Corollary 1. *For all $\pi \in \mathcal{P}$,*

$$|\tilde{r}(\pi) - \mu_t(\pi)| \leq \Big(\psi_t + \frac{\Delta\sqrt{t}}{\bar{\sigma}}\Big)\sigma_t(\pi), \tag{42}$$

where $\psi_t = \frac{\epsilon^{max}\sqrt{t}}{\bar{\sigma}} + V$.

Proof. The proof follows from Proposition 5 and Lemma 7 by substitution,

$$\begin{aligned} |\tilde{r}(\pi) - \mu_t(\pi)| &\leq |\tilde{r}(\pi) - \mu_t^h(\pi)| + |\mu_t^h(\pi) - \mu_t(\pi)| \\ &\leq \Big(\frac{\epsilon^{max}\sqrt{t}}{\bar{\sigma}} + V\Big)\sigma_t(\pi) + \Big(\frac{\Delta\sqrt{t}}{\bar{\sigma}}\Big)\sigma_t(\pi) \\ &\leq \Big(\frac{(\epsilon^{max} + \Delta)\sqrt{t}}{\bar{\sigma}} + V\Big)\sigma_t(\pi). \end{aligned} \tag{43}$$

This proves the statement. □

Based on the above results, we can provide a deterministic asymptotic regret bound for our algorithm with respect to the hypothesis function.

Theorem 3. *Consider the hypothesis class $\mathcal{F}_k(\mathcal{P}, V)$ of functions on the domain $\mathcal{P} \subset \mathbb{R}$ for some $V > 0$. For any $\bar{r}$ defined on $\mathcal{P}$ and $\Delta \geq 0$ such that $\min_{\bar{r}^h\in\mathcal{F}_k(\mathcal{P},V)} ||\bar{r}^h - \bar{r}||_\infty \leq \Delta$, the BRM algorithm achieves asymptotic regret*

$$\tilde{\rho}(T) = \mathcal{O}\Big(T(\Delta + \epsilon^{max})\sqrt{\Gamma_T} + V\sqrt{\Gamma_T T}\Big), \tag{44}$$

where Γ_T is kernel specific sublinear maximum information gain.

Proof. The proof can be found in the Appendix. □

Combining the above with (32), we can bound the regret of the proposed algorithm with respect to the target function $\bar{r}$.

Theorem 4. *Consider the hypothesis class $\mathcal{F}_k(\mathcal{P}, V)$ of functions on the domain $\mathcal{P} \subset \mathbb{R}$ for some $V > 0$. Let $\Delta \geq 0$ such that $\min_{\bar{r}^h\in\mathcal{F}_k(\mathcal{P},V)} ||\bar{r}^h - \bar{r}||_\infty \leq \Delta$. Then the asymptotic regret, in the sense of (28), of the BRM algorithm is*

$$\begin{aligned} \rho_{\mathcal{X}}(T) &= \mathcal{O}\Big(T\Big((\Delta + \epsilon^{max})\sqrt{\Gamma_T} + 2\Delta\Big) + V\sqrt{\Gamma_T T}\Big) \\ &= \mathcal{O}\Big(T(\Delta + \epsilon^{max})\sqrt{\Gamma_T} + V\sqrt{\Gamma_T T}\Big). \end{aligned}$$

Proof. Observe that by (32) , $(\bar{r}(\pi(t)) - \tilde{r}(\pi(t))) \in [-\Delta, \Delta]$ for any $\pi(t)$, and thus $|\rho_{\mathcal{X}}(T) - \tilde{\rho}(T)| \leq 2\Delta T$. The result then follows from (44). □

The bound given in Theorem 4 shows that the choice of the kernel function plays a prominent role in minimizing the regret bound. Observe that Δ, V, Γ_T depend on the choice of the kernel by the service operator. We know by Proposition 2 that the average revenue is a piece-wise linear function, hence it is rough (the opposite of smooth). Thus, choosing a smooth (rough) kernel function will result in a high (low) pointwise error Δ and low (high) V since the norm $||\bar{r}^h||_k \leq V$ is a measure of the roughness of the target function, and low (high) Γ_T since the information gain obtained from smooth (rough) functions is low (high) because of the high (low) correlation between nearby observations. The service operator can influence these parameters through the choice of the kernel, and thus it can influence the worst case accuracy of the algorithm.

V. Numerical Results

We used extensive simulations to evaluate the performance of the proposed algorithm in terms of service operator utility, exploration vs. exploitation, and the effect of the WDs' probability of being active on the average reward.

For the evaluation we consider a system with up to $N = 100$ WDs, up to $|\mathcal{J}| = 60$ applications and storage capacity up to $S = 8$. The computational complexity L_j is drawn from a uniform distribution on $[100, 1100]$ cycles/B, and the cost c_j of application j is drawn from a uniform distribution on $[0.01, 0.1]$\$. The computational capability of the edge server is $f^c = 12$ GHz. The task types of the WDs are chosen uniform at random from $\mathcal{J}$.

For each WD, the maximum transmission power $\hat{p}$ is drawn from a uniform distribution on $[150, 350]$ mW, and f_i^l is drawn from a uniform distribution on $[0.1, 0.8]$ GHz, and D_i is drawn from a uniform distribution on $[1, 50]$ MB. The channel noise variance $\tilde{\sigma}_i^2$ and the channel gain h_i are uniformly distributed on $[0.1, 0.3]$ and $[0.8, 1]$, respectively. We set $\gamma_i = 10^{-18}, \beta_i = 1, \forall i \in \mathcal{N}$. The probability q_i that WD i is active is drawn from a uniform distribution on $[0, 1]$. Lastly, the channel bandwidth W is chosen uniform at random on $[200, 300]$ MHz for each simulation. These choices of parameters are similar to those used in previous work [30, 31]. The results shown are the averages of at least 150 simulations, together with 95% confidence intervals.

We use three baselines for comparison. The first baseline knows the realizations of $B_i(t)$ and the parameters of the active WDs in every time slot $t \leq T$. It uses Algorithm 3 given in [32] to compute the cached set $\mathcal{X}^*(t)$ at every time slot t and the corresponding optimal price. We refer to this as the *Oracle*. The second baseline is *Static Expected Reward Maximization* (SERM), which knows the parameters of the WDs, and uses this knowledge for computing the expected reward $\bar{r}(\{j\}, \pi^*)$ for all $j \in \mathcal{J}$. SERM then uses Algorithm 3 in [32] with $\bar{r}(\{j\}, \pi^*)$ as input to estimate the optimal service caching (instead of the instantaneous reward $R(\{j\}, \pi^*)$). This baseline is expected to serve as an upper bound for the performance upper of BRM since it has access to more information about the system. The third baseline is the *Combinatorial Upper Confidence Bound* (CUCB) algorithm proposed in [21] for combinatorial multi-armed bandit problems. CUCB knows the WDs' parameters, and at the end of every time slot it can calculate the price $\pi^*_{\hat{\mathcal{X}}}$ that would have been optimal given the active WDs by using Algorithm 2, and the corresponding reward of each application. It maintains the average of the computed optimal prices and rewards for each application j, which it uses for choosing the set of applications to be cached using the CUCB algorithm, together with the average of the prices of the chosen applications.

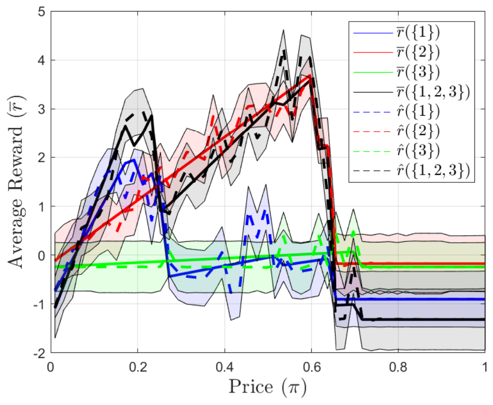

Fig. 3. Average reward vs. price for caching various applications, GP approximation vs. actual.

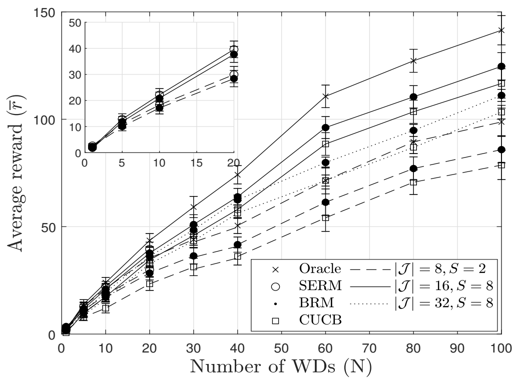

Fig. 4. Average reward vs. number of WDs (N).

A. Approximation of Average Reward

Fig. 3 shows the average reward as a function of the price (\$/Gcycle) for $|\mathcal{J}| = 3$ and $S \leq 3$. Solid lines represent the actual expected rewards and the dashed lines show the estimates obtained using the BRM algorithm. BRM approximates well the rewards around their maxima, and as such it manages to find prices that are close to optimal. It is interesting to note that $\bar{r}(\{2\}, \pi^*_{\{2\}})$ is slightly higher than $\bar{r}(\{1,2,3\}, \pi^*_{\{1,2,3\}})$, i.e., caching more applications is detrimental to the average reward.

B. Service Operator's Profit

Fig. 4 shows the average reward of the service operator as a function of the number of WDs for three scenarios. The figure shows results for SERM for $N \leq 20$, as the time required for calculating the expected reward in (12) increases exponentially with the number of WDs. Nonetheless, for $N \leq 20$ we can observe that BRM performs close to SERM, and as such BRM approximates the expected reward of the individual applications sufficiently well. In addition, BRM outperforms CUCB for all scenarios, especially as the number of WDs increases. This is because the interaction between WDs become more intricate as the number of WDs increases, and hence taking the mean of the optimal prices from all

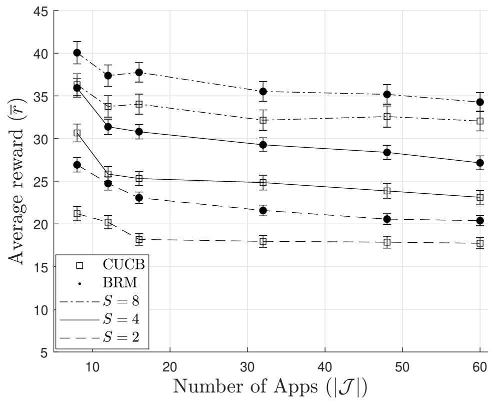

Fig. 5. Average reward vs. number of applications ($|\mathcal{J}|$).

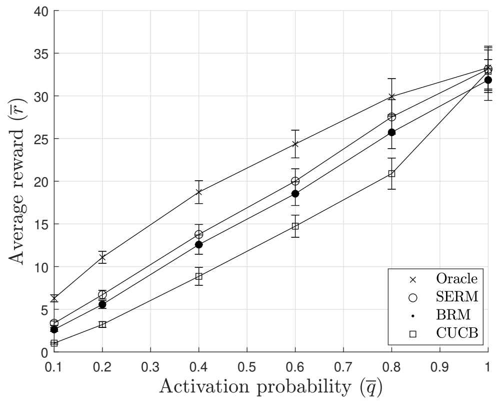

Fig. 7. Average reward vs. average activation probability of the WDs ($\overline{q}$) for $N = 10$, $|\mathcal{J}| = 8$, $S = 2$.

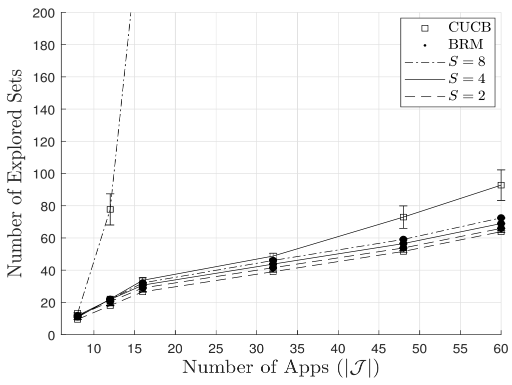

Fig. 6. Number of explored set of apps. vs. the number of apps. ($|\mathcal{J}|$).

realizations of $\mathcal{N}^a(t)$ for a given application placement in each time slot fails to perform well. It also is interesting to note that as the number of WDs increases, the gap between the *Oracle* and the rest of the curves increases since the entropy of the active WDs increases (c.f. Section V-D).

Comparing the results for the three scenarios, we observe that scenario ($|\mathcal{J}| = 16, S = 8$) has the highest average reward, which is due to that this scenario has the highest average number of WDs per cached application ($\frac{S}{|\mathcal{J}|}$), which allows more reward per application. Scenarios ($|\mathcal{J}| = 32, S = 8$) and ($|\mathcal{J}| = 8, S = 2$) have the same ratio, yet the former allows higher reward because there are more applications that can offer better options to service operator to choose from.

Fig. 5 shows the average reward as a function of the number of applications ($|\mathcal{J}|$) for $N = 20$, up to $S = 8$. The figure shows that the algorithms are fairly insensitive to the increase in the number of applications. The figure also shows that BRM can explore well and can approximate the optimal application placement despite high number of applications and consistently outperforms CUCB.

C. Exploration vs. Exploitation

Fig. 6 shows the number of distinct sets of applications explored by the algorithms as a function of the number of applications. The figure shows that the proposed BRM scales well as it does not explore as many sets as CUCB does, and yet achieves superior reward, i.e., it performs better in exploitation. The difference is most significant for higher values of storage capacity S.

D. Effect of the Activation Probability

Fig. 7 shows the average reward as a function of the activation probability of the WDs. For simplicity, we used the same activation probability for all WDs, i.e., $q_i = \overline{q}$. The figure shows that the gap between the *Oracle* and the rest of the algorithms is highest for $\overline{q} = 0.5$, when the randomness of activations is highest. On the contrary, for $q_i = 1$, *Oracle* and SERM achieve the same reward because $\mathcal{N}^a(t) = \mathcal{N}$. BRM performs very close to SERM for all values of the activation probability, despite it not having access to WDs parameters. Overall, we can conclude that the proposed BRM algorithm achieves high utility at low computational complexity.

VI. Related Work

A number of recent works deal with energy efficient computation offloading for a single mobile user [33]–[37]. [33] proposes a system that enables energy-aware offloading to the infrastructure. Also the proposed algorithm maximizes energy savings with minimal computational burden. [34] proposed CPU frequency scaling and transmission power adaptation to optimize energy consumption of the computation of a task. [35] investigated the cloud computing in terms of use of bandwidth and energy consumption, and provided the results obtained from an experimental platform (Amazon EC2). The results show that cloud offloading is sustainable considering the energy consumption. [36] presents a dynamic offloading algorithm in order to achieve energy savings under time constraints. In [37], experimental results are used to show that battery power savings can be achieved using computation offloading. Inspired by these works that show the potential energy savings through offloading, we consider a system level optimization problem with an emphasis on the interaction between the WDs and the service operator, and provide a game theoretic analysis combined with online learning.

Going beyond offloading by a single device, a number of recent works proposed optimization approaches to minimize the cost of task execution for multiple mobile devices [38]–[41]. Authors in [38] model the cost of the users as a combination of the energy consumption and the completion time, formulate the problem as a Markov decision process, and provide a near-optimal offloading policy. Authors in [39] study task partitioning to maximize throughput in processing streaming data. A two-tiered edge/cloud model with user mobility in a location-time workflow framework was considered in [40], and a heuristic was proposed to minimize the sum cost of mobile users. Authors in [41] consider the joint allocation of wireless and cloud resources and proposed an iterative algorithm to minimize users' energy consumption. Unlike these works that focus on the WDs costs only, our model and problem formulation account for the financial incentives of the service operator as well, and provides a joining treatment of the problem faced by WDs and by the operator.

Another line of works provide a game theoretic and optimization treatment of the computation offloading problem [42]–[47]. [42] allows WDs to choose what share of their task to offload in order to minimize the energy consumption and at the same time to meet its delay constraint, while the cloud allocates resources accordingly. [43] considers a model in which tasks arrive simultaneously to the cloud through a single wireless link and proposes a non-cooperative game among users that minimize their own energy use. The users are subject to execution deadlines, and have user specific channel bit rates. [45] considers a hierarchical MEC network, where mobile users can make offloading decisions, and decide the uplink transmission power, perform cloud selection, and route the tasks. A distributed offloading approach is developed based on the game theory, in which UEs collaborate with each other to minimize the network cost in terms of energy consumption and latency. [47] models the load-balancing problem as a stochastic congestion game in which each users aims to minimize its task execution time. The experiments show that the proposed algorithm can improve the load balancing of the cloud system, and enhance the quality of service. Different from these works, our model considers service caching and pricing together with the optimization problem faced by WDs, resulting in a Stackelberg game formulation.

Most related to ours are recent works that consider application caching and offloading [8, 9]. [8] formulates a Bayesian Stackelberg game, where the leader is the operator and followers are WDs. The operator's aim is to maximize the total revenue by choosing a price and applications to cache, while WDs aim to minimize their cost in terms of the charged price and delay. [9] considers the joint optimization of computation, caching, and communication to an edge cloud and uses simulations to show that the proposed method achieves shorter completion times compared to the other schemes. Our work is different from both of these works in terms of the modelling assumptions and the problem formulation. In [8], authors do not consider slotted time, dynamic population and resource management. In [9], authors do not consider slotted time and resource management, but they consider dynamic task requests. Unlike our work, they do not analyze the interaction between WDs and the operator, and thus they formulate a cost minimization problem to be solved by the operator.

Contrary to the works that formulate a game theoretic model, our model considers the interactions between WDs as a player-specific congestion game, and we model the interaction between WDs and the operator as a Stackelberg game. We then analyze the existence of equilibria, and we propose an algorithm for calculating the optimal pricing for given application placement under perfect and complete information. In addition, we consider the incomplete information case with a dynamic population of WDs, and we propose a novel Bayesian Gaussian Process Bandit optimization approach for joint pricing and caching.

Related in terms of methodology are recent recent works that propose to use BO in edge computing. In [48] authors use BO for finding a trade-off between performance and energy consumption in *virtual Base Stations* (vBS), based on a GP model combined with contextual bandit optimization. Authors in [49] propose BO for learning the relationship between the cost and the run-time of serverless functions and the function instance configuration, and they aim at minimizing the cost of using a serverless system from a single WD's perspective by choosing the memory allocated to the function. Different from [49], where authors use the *expected improvement* as acquisition function, which may fail to find a good balance between exploration and exploitation for very rough target functions, we propose to use an acquisition function based on an upper confidence bound, so as to provide robustness despite a discontinuous target function. Different from these works, our proposed solution BRM employs BO with GP by introducing a new acquisition function and combines this with a novel heuristic that approximates the optimal service caching.

VII. Conclusion

In this work we have provided a game theoretic analysis of pricing, application caching and computation offloading for edge computing. For the case of complete and perfect information, we showed that an equilibrium of offloading decisions and an optimal price for a particular caching decision can be computed in polynomial time, but the efficient computation of a strong Stackelberg equilibrium is infeasible due to the intricate interactions between caching decisions for different applications. We then analyzed the incomplete information case with a dynamic population of users, and proposed a novel Bayesian Gaussian Process Bandit optimization approach for joint pricing and caching. Our numerical results show that the proposed algorithm is computationally efficient, and it outperforms state-of-the-art combinatorial multi-armed bandit algorithms. Future directions of research include considering pricing for the use of wireless resources, heterogeneous pricing for computing resources, and WDs whose activity may be correlated over time or may be non-stationary.

References

[1] M. Hakkarainen, C. Woodward, and M. Billinghurst, "Augmented assembly using a mobile phone," in *IEEE/ACM Intl. Symp. on Mixed and Augmented Reality*, 2008, pp. 167–168.

[2] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, “uWave: Accelerometer-based personalized gesture recognition and its applications,” in *IEEE Intl. Conf. on Pervasive Computing and Communications*, 2009, pp. 1–9.

[3] S. Jošilo and G. Dán, “Joint management of wireless and computing resources for computation offloading in mobile edge clouds,” *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1507–1520, 2021.

[4] C. You, K. Huang, and H. Chae, “Energy efficient mobile cloud computing powered by wireless energy transfer,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1757–1771, 2016.

[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 Transactions on Wireless Communications*, vol. 19, no. 1, pp. 235–250, 2020.

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

[7] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, “Joint service placement and request routing in multi-cell mobile edge computing networks,” in *IEEE INFOCOM*, 2019, pp. 10–18.

[8] J. Yan, S. Bi, L. Duan, and Y.-J. A. Zhang, “Pricing-driven service caching and task offloading in mobile edge computing,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 7, pp. 4495–4512, 2021.

[9] M. Chen, Y. Hao, L. Hu, M. S. Hossain, and A. Ghoneim, “Edge-cocaco: Toward joint optimization of computation, caching, and communication on edge cloud,” *IEEE Wireless Communications*, vol. 25, no. 3, pp. 21–27, 2018.

[10] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, “Wireless caching: technical misconceptions and business barriers,” *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, 2016.

[11] P. Cruz, N. Achir, and A. C. Viana, “On the edge of the deployment: A survey on multi-access edge computing,” *ACM Comput. Surv.*, mar 2022, just Accepted.

[12] Z. Xiong, J. Zhao, Y. Zhang, D. Niyato, and J. Zhang, “Contract design in hierarchical game for sponsored content service market,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 9, pp. 2763–2778, 2021.

[13] G. Poghosyan, I. Pefkianakis, P. Le Guyadec, and V. Christophides, “Extracting usage patterns of home IoT devices,” in *IEEE Symposium on Computers and Communications*, 2017, pp. 1318–1324.

[14] L. Xu, G. Shao, Y. Cao, H. Yang, C. Sun, T. Zhang, B. Wen, X. Cheng, C. Song, and X. He, “Research on telecom big data platform of LTE/5G mobile networks,” in *IEEE International Conferences on Ubiquitous Computing Communications and Data Science and Computational Intelligence and Smart Computing, Networking and Services*, 2019, pp. 756–761.

[15] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, “Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing,” *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1634–1644, 2021.

[16] W. Chen and L. Han, “Time-efficient task caching strategy for multi-server mobile edge cloud computing,” in *IEEE HPCC/SmartCity/DSS*, 2019, pp. 1429–1436.

[17] V. K. Garg, “An overview of digital communication and transmission,” in *Wireless Communications Networking*. Morgan Kaufmann, 2007, pp. 85–122.

[18] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, “Online collaborative data caching in edge computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 281–294, 2021.

[19] P. Spirakis, M. Mavronicolas, and S. Kontogiannis, *Proc. of Intl. Workshop on Internet and Network Economics*, 2006, vol. 4286.

[20] I. Milchtaich, “Congestion games with player-specific payoff functions,” *Games and Economic Behavior*, vol. 13, no. 1, pp. 111–124, 1996.

[21] W. Chen, Y. Wang, and Y. Yuan, “Combinatorial multi-armed bandit: General framework and applications,” in *Proc. of Intl. Conf. on Machine Learning*, vol. 28, no. 1, 2013, pp. 151–159.

[22] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, and P. Lu, “Combinatorial multi-armed bandit with general reward functions,” in *Conf. on Neural Information Processing Systems*, 2016, p. 1659–1667.

[23] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.* MIT Press, 2006.

[24] I. Bogunovic and A. Krause, “Misspecified Gaussian process bandit optimization,” in *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[25] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, “Gaussian process bandits without regret: An experimental design approach,” *Intl. Conf. on Machine Learning*, 2009.

[26] S. Vakili, N. Bouziani, S. Jalali, A. Bernacchia, and D. Shiu, “Optimal order simple regret for Gaussian process bandits,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 21 202–21 215.

[27] F. Berkenkamp, A. P. Schoellig, and A. Krause, “No-regret Bayesian optimization with unknown hyperparameters,” *J. Mach. Learn. Res.*, vol. 20, no. 1, p. 1868–1891, 2019.

[28] C. A. Micchelli, Y. Xu, and H. Zhang, “Universal kernels,” *J. Mach. Learn. Res.*, vol. 7, p. 2651–2667, 2006.

[29] I. Steinwart and A. Christmann, *Support Vector Machines*, 1st ed. Springer, 2008.

[30] Y. Huo, X. Dong, and W. Xu, “5G cellular user equipment: From theory to practical hardware design,” *IEEE Access*, vol. 5, pp. 13 992–14 010, 2017.

[31] P. Joshi, F. Ghasemifard, D. Colombi, and C. Törnevik, “Actual output power levels of user equipment in 5G commercial networks and implications on realistic RF EMF exposure assessment,” *IEEE Access*, vol. 8, pp. 204 068–204 075, 2020.

[32] F. Tütüncüoğlu and G. Dán, “Optimal pricing for service caching and task offloading in edge computing,” in *2022 17th Wireless On-Demand Network Systems and Services Conference (WONS)*, 2022, pp. 1–8.

[33] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “MAUI: Making smartphones last longer with code offload,” vol. 2010, Oct. 2010, pp. 49–62.

[34] Y. Wen, W. Zhang, and H. Luo, “Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones,” in *Proc. IEEE INFOCOM*, 2012, pp. 2716–2720.

[35] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, “To offload or not to offload? The bandwidth and energy costs of mobile cloud computing,” in *Proc. IEEE INFOCOM*, 2013, pp. 1285–1293.

[36] D. Huang, P. Wang, and D. Niyato, “A dynamic offloading algorithm for mobile computing,” *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

[37] A. Rudenko, P. Reiher, G. Popek, and G. Kuenning, “Saving portable computer battery power through remote process execution,” *Mobile Computing and Communications Review*, vol. 2, 1998.

[38] E. Hyytiä, T. Spyropoulos, and J. Ott, “Offload (only) the right jobs: Robust offloading using the markov decision processes,” in *IEEE Intl. Symp. on a World of Wireless, Mobile and Multimedia Networks*, 2015, pp. 1–9.

[39] L. Yang, J. Cao, S. Tang, T. Li, and A. T. Chan, “A framework for partitioning and execution of data stream applications in mobile cloud computing,” in *IEEE Intl. Conf. on Cloud Computing*, 2012, pp. 794–802.

[40] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, “MuSIC: Mobility-aware optimal service allocation in mobile cloud computing,” in *IEEE Intl. Conf. on Cloud Computing*, 2013, pp. 75–82.

[41] S. Sardellitti, G. Scutari, and S. Barbarossa, “Joint optimization of radio and computational resources for multicell mobile-edge computing,” *IEEE Trans. on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[42] Y. Wang, X. Lin, and M. Pedram, “A nested two stage game-based optimization framework in mobile cloud computing system,” in *IEEE Intl. Symp. on Service-Oriented System Engineering*, 2013, pp. 494–502.

[43] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, “Energy efficient offloading for competing users on a shared communication channel,” in *IEEE Intl. Conf. on Comm.*, 2015, pp. 3192–3197.

[44] C. Lin, X. Xiang, and C. Chen, “Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing,” in *Proc. of Intl. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2015, pp. 271–278.

[45] B. Wu, J. Zeng, L. Ge, X. Su, and Y. Tang, “Energy-latency aware offloading for hierarchical mobile edge computing,” *IEEE Access*, vol. 7, pp. 121 982–121 997, 2019.

[46] B. Zhou and H. Yang, “Rendering scheduling framework in edge computing: A congestion game-based approach,” in *IEEE Intl. Performance Computing and Communications Conf.*, 2019, pp. 1–6.

[47] F. Zhang and M. M. Wang, “Stochastic congestion game for load balancing in mobile-edge computing,” *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 778–790, 2021.

[48] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, “Bayesian online learning for energy-aware resource orchestration in virtualized rans,” in *IEEE INFOCOM 2021*, 2021, pp. 1–10.

[49] N. Akhtar, A. Raza, V. Ishakian, and I. Matta, “COSE: Configuring serverless functions using statistical learning,” in *IEEE INFOCOM 2020*, 2020, pp. 129–138.

[50] S. R. Chowdhury and A. Gopalan, "On kernelized multi-armed bandits," in *Proc. of Intl. Conf. on Machine Learning - Volume 70*, 2017, p. 844–853.

[51] I. Bogunovic, A. Krause, and S. Jonathan, "Corruption-tolerant Gaussian process bandit optimization," in *Conf. on Artificial Intelligence and Statistics*, 2020.

Feridun Tütüncüoğlu is a Ph.D. student at the Division of Network and Systems Engineering in KTH Royal Institute of Technology, Stockholm, Sweden. He received M.Sc in Electrical & Electronics Engineering from Bilkent University, Turkey in 2019. He worked as a research engineer at the department of Electrical & Electronics Engineering, Bilkent University from 2017 to 2019. His research interests include design and analysis of decentralized algorithms, online learning algorithms and game theoretical models of edge computing resource management and allocation.

György Dán (M'07, SM'17) is a professor at KTH Royal Institute of Technology, Stockholm, Sweden. He received the M.Sc. in computer engineering from the Budapest University of Technology and Economics, Hungary in 1999, the M.Sc. in business administration from the Corvinus University of Budapest, Hungary in 2003, and the Ph.D. in Telecommunications from KTH in 2006. He worked as a consultant in the field of access networks, streaming media and videoconferencing 1999-2001. He was a visiting researcher at the Swedish Institute of Computer Science in 2008, a Fulbright research scholar at University of Illinois at Urbana-Champaign in 2012-2013, and an invited professor at EPFL in 2014-2015. He served as area editor of Computer Communications 2014-2021, and has been editor of IEEE Transactions on Mobile Computing since 2019. His research interests include the design and analysis of content management and computing systems, game theoretical models of networked systems, and cyber-physical system security and resilience.