Optimal Pricing for Service Caching and Task Offloading in Edge Computing

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. We model the problem as a multiple-follower Stackelberg game, where the operator is the leader and decides what applications to cache and how much to charge for their use, while the wireless devices (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 the equilibrium price of the operator can be computed in polynomial time for any cache placement, and propose a greedy algorithm for computing the applications to be cached. We use extensive simulations to show that the proposed heuristic performs close to optimal at negligible computational overhead.**

I. Introduction

Battery powered *Wireless Device*s (WDs) are increasingly used for computationally intensive applications including augmented reality, natural language processing, face, gesture and object recognition [1], [2]. Performing such computationally intensive applications on battery powered devices is detrimental to battery life, and may adversely affect user experience.

Edge computing could enable WDs to offload computationally intensive tasks to nearby compute resources in the infrastructure via wireless networks. Through computation offloading, WDs can potentially reduce their energy consumption, while meeting application latency requirements. If many WDs offload simultaneously, however, application performance could suffer due to limited wireless, computational and storage resources in the edge infrastructure. Thus, either pricing or admission control have to be implemented to mitigate contention to an acceptable level.

The contention for communication and computing resources, and their joint management have been explored in the literature [3], [4], [5], [6], [7], [8], [9], but the interaction with the management of storage, i.e., the availability of executable code and data at the edge server, and the impact of pricing, are to a large extent unexplored. Code availability and pricing become particularly important in the case of emerging *Function as a Service* (FaaS) offerings (often called serverless computing), where applications are executed on-demand, by loading container images from storage to memory, and charging is based on execution time. Loading container images is, however, time consuming, and thus for latency sensitive applications caching, i.e., pre-loading the functions is essential in FaaS for avoiding the execution delay that would result from a cold start.

This work was partly funded by the Vinnova Competence Center for Trustworthy Edge Computing Systems and Applications at KTH and by the Swedish Research Council through project 2020-03860.

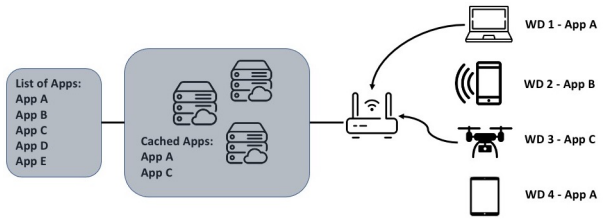

Fig. 1. Edge computing system with $N = 4$ WDs, and $|\mathcal{J}| = 5$ applications. The operator caches $|\mathcal{X}| = 2$ apps due to its memory storage constraint. WDs decide to offload or to perform the apps locally based on their costs and their completion time constraints. WD 1 and WD 3 decide to offload, WD 2 cannot offload because App B is not cached, and WD 4 chooses local computing.

In this work, we explore the interaction between a profit maximizing operator that performs application caching and pricing, and cost minimizing autonomous WDs that can offload their computation, subject to application availability and latency constraints. Our main contributions are as follows.

- We propose a Stackelberg game to model the interaction between the 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 Equilibrium* (NE).
- We propose a polynomial time algorithm for computing the optimal price to be charged by the operator, and a greedy heuristic for application caching.
- We use extensive simulations for showing that the resulting solution is close the Stackelberg equilibium, and significantly outperforms popularity-based caching.

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 NE in Section III, and we propose an algorithm for optimal pricing in Section IV. We show numerical results in Section V, and discuss related work in Section VI. Section VII concludes the paper.

II. System Model and Problem Formulation

We consider a multi-access edge computing system that consists of an edge server with memory storage capacity S, 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. We denote by $\phi_i \in \mathcal{J}$ the type of the task that WD i wants to execute, and we refer to $\mathcal{J}$ as 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 required to perform the task (e.g, in Gcycles) for $j = \phi_i$, and by the completion time requirement $\bar{\tau}_i$.

The operator can use its storage for caching a subset $\mathcal{X} \subseteq \mathcal{J}$ of applications, subject to capacity constraint

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

where s_j is the size of the software image for application j. Caching application j involves a usage cost r_j to the operator, but allows it to charge unit price π to each offloading WD with $\phi_i = j$. We consider that the price π is application independent, aligned with pricing in current FaaS offerings.

We denote by a_i the offloading decision of WD i; $a_i = 1$ corresponds to offloading, and $a_i = 0$ to local computing. Offloading is only possible for applications that are cached by the operator, i.e., $\phi_i \notin \mathcal{X} \rightarrow a_i = 0, \forall i$. Next, we present our model of local computing and of 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

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

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 [10].

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

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

the number of WDs that offload, 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 [11], [12], and we express the data rate achievable by WD i using the Shannon formula [13],

$$R_i^u(p_i, m) = \frac{W_i}{m} log_2(1 + \frac{p_i h_i}{\sigma_i^2}), \tag{4}$$

where W_i is the channel bandwidth, p_i is the transmit power of the transmitted signal, h_i is the channel coefficient from WD i to the AP, and σ_i^2 is the noise power at the AP. 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

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

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 processing time at the edge server as

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

C. WD Cost Model

We model the cost of WD i as a combination of its energy consumption and the price charged by the 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.,

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

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 operator. We consider that the execution cost is proportional to the task complexity L_{ϕ_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

$$C_i^1 = \tau_i^u p_i \beta_i + L_{\phi_i} \pi, \tag{8}$$

where π is the unit price charged by the operator. The cost of WD i is thus

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

D. Problem Formulation

We consider that the WDs and the 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 operator, i.e., it aims to solve

$$\min_{a_i \in \{0,1\}, p_i \leq \hat{p}_i} C_i(a_i, p_i, a_{-i}) \tag{10}$$

$$\text{s.t.} \quad a_i(\tau_i^u(p_i, m) + \tau_i^c(m)) \leq \tau_i^l, \tag{11}$$

$$a_i = 0 \text{ if } \phi_i \notin \mathcal{X}, \tag{12}$$

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

The operator's profit is the difference of its income from the WDs that offload and the cost of caching the applications,

$$U(a, \mathcal{X}, \pi) = \sum_{i=1}^{N} a_i \mathbb{1}_{\mathcal{X}}(\phi_i) L_{\phi_i} \pi - \sum_{j \in \mathcal{X}} r_j, \tag{13}$$

where r_j is the cost of caching application j, and $\mathbb{1}_{\mathcal{X}}(\phi_i)$ is the indicator function. The operator's objective is to maximize its profit by choosing the applications to cache and the price of executing tasks, i.e.,

$$\max_{\pi \geq 0, \mathcal{X} \subseteq \mathcal{J}} \quad U(a, \mathcal{X}, \pi), \tag{14}$$

$$\text{s.t.} \quad (1). \tag{15}$$

The resulting problem is a multi-follower Stackelberg game, where the operator is the leader and the WDs are the followers. We refer to the problem as the *Time Constrained Computation Offloading* (TCCO) game, and we are interested in the existence of Stackelberg equilibria and the complexity of computing equilibria, under complete information, i.e., the system parameters and utilities are known. While this assumption may be strong, it allows to explore the structure of the game and is fundamental for subsequent analysis under incomplete information.

III. Existence of Equilibria

We first focus on the best response of the WDs, i.e., the followers, for a given caching decision $\mathcal{X}$ and price π set by the operator. The best response is in effect a Nash equilibrium (NE) played by the WDs, if such a NE exists.

To assess whether NE exist, we start with characterizing the optimal offloading decision for WDs.

Lemma 1. *Consider a WD i such that $\phi_i \in \mathcal{X}$, and let $m = \sum_{i' \neq i} a_{i'} + 1$. If $\tau_i^c(m) > \tau_i^l$ then the optimal offloading decision is $a_i^* = 0$. Otherwise, let p_i^* be such that $\tau_i^u(m, p_i^*) + \tau_i^c(m) = \tau_i^l$. Then, if $p_i^* > \hat{p}_i$ then $a_i^* = 0$, otherwise*

$$a_i^* = \begin{cases} 1, & \pi \leq \beta_i (f_i^l \gamma_i^l - p_i^* (\frac{1}{f_i^l} - \frac{m}{f_i^c})) \\ 0, & else, \end{cases} \tag{16}$$

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 $\tau_i^u(m, p_i)$ is a strictly monotonically decreasing function of p_i, and $C(1, p_i, a_{-1})$ is a strictly monotonically increasing function of p_i. Thus, WD i minimizes its cost by choosing a transmit power p_i^* that yields $\tau_i^u(m, p_i^*) + \tau_i^c(m) = \tau_i^l$. Now, if $p_i^* > \hat{p}_i$ then offloading is not feasible. Otherwise, if $p_i^* \leq \hat{p}_i$ then the optimal decision is

$$a_i^* = \begin{cases} 1, & C(1, p_i^*, a_{-i}) \leq C(0, p_i^*, a_{-i}) \\ 0, & else. \end{cases} \tag{17}$$

We can substitute $\tau_i^u(m, p_i^*) = \tau_i^l - \tau_i^c(m)$, (2) and (6) into (17), and obtain (16), which proves the result. □

Using the above best response, a NE of the WDs is a collection of offloading decisions $(a_i^*)_{i \in \mathcal{N}}$ such that $C(a_i^*, p_i^*, a_{-i}^*) \leq C(1 - a_i^*, p_i^*, a_{-i}^*)$, $\forall i \in \mathcal{N}$. Next, we use a topological equivalence argument to show that a NE always exists.

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

Proof. Note that the TCCO 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 $\tilde{C}_i(0, p_i^*, a_{-i}) = C_i(0, p_i^*, a_{-i}) - L_{\phi_i}\pi$, and the cost of offloading is $\tilde{C}_i(1, p_i^*, a_{-i}) = C_i(1, p_i^*, a_{-i}) - L_{\phi_i}\pi$. 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 [14], and so does Γ. This concludes the proof. □

In addition, an equilibrium can be computed through letting WDs update their offloading strategies one at a time, as we show next.

Lemma 2. *The game played among the WDs in the TCCO 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 [15]. □

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, and it can be computed efficiently.

IV. Optimal Pricing and Caching Policy

In this section, we propose a polynomial time algorithm for computing the optimal equilibrium price for the operator caching decision $\mathcal{X}$, and then a greedy policy for computing a caching decision. Throughout the section we consider *Strong Stackelberg Equilibrium* (SSE), i.e., if there are multiple subgame perfect equilibria then one with maximum utility for

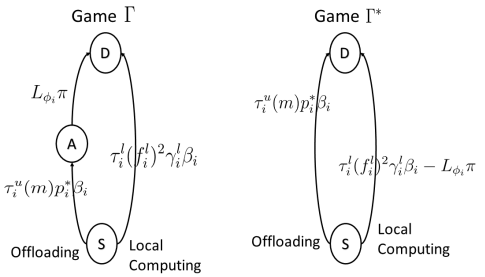

Fig. 2. Topology of the network congestion game Γ and Γ^* used in the proof of Theorem 1.

the operator will be chosen. This is a reasonable assumption if the operator executes the proposed algorithms.

A. Computing the Equilibrium Price

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 the 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 and denote by $\Pi^t_m = \{\pi |\, |\{i | \pi_{i,m} \geq \pi\}| \geq m\}$ the set of threshold prices such that there are at least m WDs with $\pi_{i,m} \geq \pi$. 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 and define $\Pi^t_{\mathcal{X},m} = \{\pi |\, |\{i \in \mathcal{N}_\mathcal{X}, \pi_{i,m} \geq \pi\}| \geq m\}$ as the set of threshold prices such that there are at least m WDs $i \in \mathcal{N}_\mathcal{X}$ with $\pi_{i,m} \geq \pi$. 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 monotonically increasing 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 $C(1, p_i^*, a_{-i}) \leq C(0, p_i^*, a_{-i})$ for π_1 if and only if $C(1, p_i^*, a_{-i}) \leq C(0, p_i^*, a_{-i})$ for π_2. 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 (13) for the equilibrium strategy profile a under homogeneous pricing and obtain

$$U(a, x, \pi) = \sum_{i \in \mathcal{N}: a_i(\mathcal{X}, \pi'') = 1} \mathbb{1}_\mathcal{X}(\phi_i) L_{\phi_i} \pi - \sum_{j \in \mathcal{X}} r_j, \quad (18)$$

which is monotonically increasing in π for any given x on $(\pi', \pi'']$. This concludes the proof. □

Proposition 1. *Let π be a price such that $\max(\Pi^t_{\mathcal{X},m+1}) < \pi \leq \max(\Pi^t_{\mathcal{X},m})$, and 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. First, we prove $\sum_{i \in \mathcal{N}_\mathcal{X}} a'_i \leq m$. We choose $\pi > \max(\Pi^t_{\mathcal{X},m+1})$, by the definition of $\Pi^t_{\mathcal{X},m+1}$ we have $|\mathcal{N}^o_\mathcal{X}(\pi, m)| = m$. Whether or not a WD $i \in \mathcal{N}^o_\mathcal{X}(\pi, m)$ would offload depends also on whether the application ϕ_i is cached by the operator. Thus, if there exists a WD $i \in \mathcal{N}^o_\mathcal{X}(\pi, m)$ such that $\phi_i \notin \mathcal{X}$, then for any equilibrium strategy $a \in \alpha^*(\mathcal{X}, \pi)$ we have $a_i = 0$. Hence, $\sum_{i \in \mathcal{N}_\mathcal{X}} a_i \leq m$. Second, 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. □

Next, we show that for given π and application placement 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 $U(a_i, \{\phi_i\}, \pi) = a_i \mathbb{1}_\mathcal{X}(\phi_i) L_{\phi_i} \pi$, and is independent of what other WDs are offloading.

Lemma 4. *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 $\sum_i a_i = m'$, thus by Proposition 1 we know that $\sum_{i \in \mathcal{N}} a_i = m'$ for any

Algorithm 1: Computing a NE for WDs

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

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

Data: $\mathcal{X}, \Pi^t_\mathcal{X}$ **Result:** π^*, U^*
/* Compute the utility for $\forall \pi_{i,m}$ */
for $k = 1 : |\Pi^t_\mathcal{X}|$ **do**
  $\mathcal{N}^\dagger = Algorithm1(\mathcal{X}, \Pi^t_\mathcal{X}(k))$
  $U(k) = \sum_{i \in \mathcal{N}^\dagger} L_{\phi_i} \Pi^t_\mathcal{X}(k)$ - $\sum_{j \in \mathcal{X}} r_j$
end
$U^* = \max_k U(k)$
$k^* = \min\{k | U(k) = \pi^*\}, \pi^* = \Pi^t_\mathcal{X}(k^*)$

$a \in \alpha^*(\mathcal{X}, \pi)$. 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 the highest payoff for the operator, recall that the income $U_i(a_i, \{\phi_i\}, \pi)$ 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_{ϕ_i} from the set $\mathcal{N}^o_\mathcal{X}(\pi, m') \setminus \mathcal{N}^o_\mathcal{X}(\pi, m'+1)$.

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_{ϕ_i}, both in polynomial time (see Algorithm 1). □

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., $U(a^*, \mathcal{X}, \pi^*) \geq U(a, \mathcal{X}, \pi) \quad \forall a^* \in \alpha^*(\mathcal{X}, \pi^*)$, $\pi, a \in \alpha^*(\mathcal{X}, \pi)$.*

Proof. Consider a price such that $\pi \notin \Pi^t_\mathcal{X}$, and allows a set of equilibria $\alpha^*(\mathcal{X}, \pi)$. If $\pi \in [0, \max(\Pi^t_\mathcal{X}))$, then there is a threshold price $\pi' \in \Pi^t_\mathcal{X}$ such that $\pi' > \pi$. 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 $\pi^* = \arg\max_{\pi' \in \Pi^t_\mathcal{X}} U(a, \mathcal{X}, \pi'(m))$. Hence, Algorithm 2 computes a price that maximizes the utility of the operator. □

Observe that the computational complexity of Algorithm 1 is $\mathcal{O}(|\mathcal{N}_\mathcal{X}|^2)$, while that of Algorithm 2 is $\mathcal{O}(|\mathcal{N}_\mathcal{X}|^4)$ as it invokes Algorithm 1 up to $|\mathcal{N}_\mathcal{X}|^2$ times.

B. Caching Applications

We have so far shown how to choose an optimal price π^* for given caching decision $\mathcal{X}$. What remains is to compute an optimal caching decision $\mathcal{X}^*$, which together with $\pi^*(\mathcal{X}^*)$ solves (14). However, computing the optimal set of applications is challenging for two reasons. First, the utility of caching an application depends on the set of applications that are already cached, as the number and type of offloaders depend on the set of cached applications. Second, caching an additional application may actually reduce the total utility of the operator, i.e., the operator's utility is not monotone.

Definition 1. *The set function $U : \mathcal{J} \to \mathbb{R}$ is monotone if for any $\mathcal{X} \subset \mathcal{J}$ and $j \in \mathcal{J} \setminus \mathcal{X}$ we have $U(\mathcal{X} \cup \{j\}) \geq U(\mathcal{X})$.*

Monotonicity is a common assumption, e.g., in Knapsack problems with independent item values. In the considered problem the operator's utility need not be monotone.

Proposition 2. *Let $\mathcal{X} \subset \mathcal{J}$ and $j \in \mathcal{J} \setminus \mathcal{X}$, then*

$$U(a, \mathcal{X} \cup \{j\}, \pi^*(\mathcal{X} \cup \{j\})) - U(a, \mathcal{X}, \pi^*(\mathcal{X})) \lesseqgtr 0. \quad (19)$$

Proof. We prove the result through the following example.

Example: Let $\mathcal{N} = \{1, 2\}$, $\mathcal{J} = \{1, 2\}$, $L_1 = 5$ Gcycles, $L_2 = 12$ Gcycles, $\phi_1 = 1, \phi_2 = 2$ $\hat{p}_1 = 200$mW, $\hat{p}_2 = 300$mW, $R_1 = 0.1$, $R_2 = 0.2$, $f^l_1 = f^l_2 = 0.7$ GHz, $f^c = 12$ GHz, $\gamma_i = 10^{-18}$, $\beta_i = 1$, $\forall i \in \mathcal{N}$, $W_1 = W_2 = 0.1$ GHz, $\sigma^2_1 = 0.4, \sigma^2_2 = 1.2$, $h_1 = h_2 = 1$ and $D_1 = 50$ MB, $D_2 = 200$MB.

The resulting threshold price matrix is $\Pi^t_{\{1,2\}} = \begin{bmatrix} 0.6886 & 0.6766 \\ 0.5263 & 0.3349 \end{bmatrix}$ \$/Gcycles. Observe that $\pi^*(\{1\}) = 0.6886$, $U(a, \{1\}, 0.6886) = 3.343$, $\pi^*(\{2\}) = 0.5263$, and $U(a, \{2\}, 0.5263) = 6.11$, while $\pi^*(\{1, 2\}) = 0.3349$ and $U(a, \{1, 2\}, 0.3349) = 5.39$. Clearly, $U(a, \{2\}, 0.5263) > U(a, \{1, 2\}, 0.3349) > U(a, \{1\}, 0.6886)$, which proves the two strict inequalities. Equality holds, e.g., for applications with no WD willing to offload. Hence, U is not monotone. □

Since the operator's utility is not monotone, we propose to use the *Non-negative Greedy Algorithm* (NNG) algorithm to compute the set of cached applications. NNG is a modified version of deterministic local search algorithm proposed in [16] with weight $\epsilon = 0$ and with knapsack constraint; its pseudocode is shown in Algorithm 3. Unfortunately, the utility of the operator is not even weakly submodular,hence it is not possible to provide an approximation ratio bound for the resulting cache placement. Instead we resort to simulations to provide a numerical evaluation of equilibrium performance.

V. Numerical Results

We used extensive simulations to evaluate the operator's utility in equilibrium, the computation time, and the average number of offloading WDs in equilibria.

For the evaluation we consider a system with up to $N = 200$ WDs, up to $|\mathcal{J}| = 40$ applications and storage capacity up to

Algorithm 3: NNG Algorithm

Data: $\mathcal{J}, S$ **Result:** $\mathcal{X}^*, U^*$
$U^* = 0,\ \mathcal{X} = \emptyset$
for $j \in \mathcal{J}$ **do**
  $U(j) = Algorithm2(\{j\}, \Pi^t_{\{j\}}))$
end
$J' = sort(U)$
$k = 1$
while $k < |\mathcal{J}| \wedge \sum_{j\in\mathcal{X}} s_j < S$ **do**
  $j = J'(k)$
  $U' = Algorithm2(\mathcal{X} \cup \{j\}, \Pi^t_{\mathcal{X}\cup\{j\}})$
  if $U' \geq U^*$ **then**
    $\mathcal{X} = \mathcal{X} \cup \{j\}$
    $\mathrm{U}^* = U'$
  end
  $k = k + 1$
end
while $\exists j \in \mathcal{X} \quad | \quad U(\mathcal{X} \setminus \{j\}) > U(\mathcal{X})$ **do**
  $\mathcal{X} = \mathcal{X} \setminus \{j\}$
  $k = 1$
  /* Return to first while loop */
end

$S = 8$. The computational complexity L_j follows a continuous uniform distribution on $[1, 10]$ Gcycles, and the cost r_j of application j follows a continuous uniform distribution on $[1, 10]$\$. 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 the WDs, the maximum transmission power $\hat{p}_i$ is uniformly distributed on $[50, 300]$ mW, and the channel bandwidth W_i is uniformly distributed on $[200, 300]$ MHz. f^l_i is uniformly distributed on $[0.5, 1.5]$ GHz, and D_i is uniformly distributed on $[1, 50]$ MB. For the channel parameters, σ^2_i is uniformly distributed on $[0.1, 0.3]$, and so is the channel gain on $[0.8, 1]$. Lastly, we set $\gamma_i = 10^{-18}$, $\beta_i = 1, \forall i \in \mathcal{N}$. These choices of parameters are similar to those used in previous work [17], [18]. The results shown are the averages of 250 simulations, together with 95% confidence intervals. We use two baselines for comparison. First, we use exhaustive search to find the optimal set of cached applications, i.e., a SSE of the TCCO game. Second, we consider caching the most popular applications in terms of $|\{i : \phi_i = j\}|$ subject to the storage constraint, and compute the optimal price for this caching decision, referred to as popularity-based caching (PBC).

A. Operator's Profit

Figure 3 shows the profit of the operator (U) as a function of the N number of WDs, for cache sizes $S \in \{2, 4, 8\}$ and application set cardinalities, and $|\mathcal{J}| \in \{8, 16, 40\}$. The figure shows that the profit of the operator has a decreasing marginal gain in the number of WDs, as an increasing contention for edge server resources makes that WDs are less likely to decide to offload. At the same time the difference between the equilibrium profit, and the profit computed using the NNG heuristic increases with the number of WDs until a point where there is contention for the resources (see curves $(J = 8, S = 2)$, $(J = 8, S = 4)$, $(J = 16, S = 2)$), indicating that the interaction between the WDs becomes more intricate causing the greedy heuristic to fail to perform well. Overall, it can be seen, however, that the NNG algorithm achieves a profit very close to the actual equilibrium, the relative difference between the profits is only up to 4%. PBC performs poorly in comparison to NNG, showing the importance of considering the strategic interaction between the WDs.

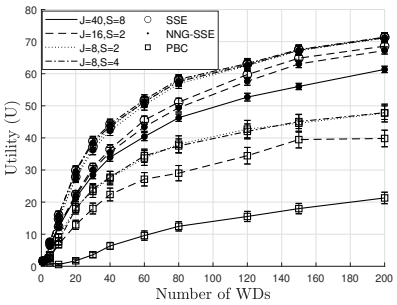

Fig. 3. Utility vs. number of WDs with $(S = 4, J = 8)$, $(S = 2, J = 8)$, and $(S = 2, J = 16)$, $(S = 8, J = 40)$.

Comparing the results for different scenarios in Figure 3 we can observe that the operator's profit is highest for $S = 4, J = 8$, and it is lowest for $S = 8, J = 40$. The reason is that for $S = 4, J = 8$ there is a higher chance of having more WDs requiring the application compared to other cases. At the same time, the profit increase from $S = 2, J = 8$ to $S = 4, J = 8$ is rather marginal, showing that the marginal gain of increased cache storage is rather low.

One more interesting observation is that the utility for the scenario $J = 16, S = 2$ is higher than that for $J = 40, S = 8$ even though the relative cache size $\frac{S}{J}$ is lower for $J = 16, S = 2$. This is because for $J = 40$ the number of WDs per application is lower, and since caching an application comes with cost r_j, the marginal utility of caching an additional application is lower for $J = 40$ than for $J = 16$.

B. Computation Time

We now evaluate the computation time of finding equilibria using exhaustive search and using the NNG heuristic. Figure 4 shows the computational time as a function of the number N of WDs for various system configurations, measured on a Matlab implementation running on an Intel i9 10900K processor. The figure highlights the computational advantage of the greedy NNG heuristic over exhaustive search as the number of WDs increases.

The computational complexity of NNG heuristic is a function of S, J and N. Increase in any input results in an increase in the computation time. Among the considered scenarios, the computation time of the NNG heuristic is the highest for $S = 8, J = 40$, although it is close to $S = 4, J = 8$. This is because in latter scenario the number of WDs per application is highest among the scenarios. Thus, both Algorithm 2 and the greedy NNG heuristic require more iterations. This is

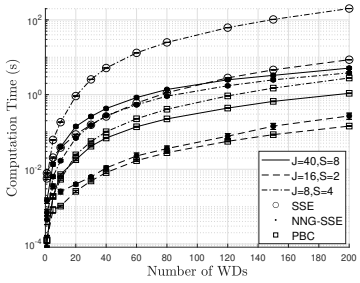

Fig. 4. Computation time vs. number of WDs for $(S = 4, J = 8), (S = 2, J = 16), (S = 8, J = 40)$.

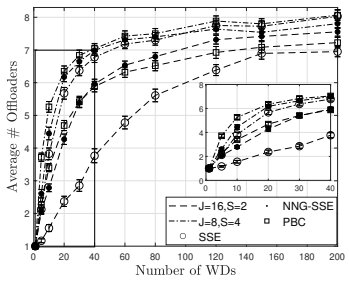

Fig. 5. Average number of offloaders vs. number of WDs for various storage capacity constraints.

confirmed by the opposite case, $S = 2, J = 16$ where the number of WDs per application is lowest, and so is the computation time. We can conclude that the number of WDs per application is an important factor that affects the computational time of finding an approximate equilibrium. Comparing NNG and PBC, we can see that the computation time of the two algorithms is similar, although finding the most popular applications is simpler than Algorithm 3. This is confirmed by the results of scenarios $J = 16, S = 2$ and $J = 40, S = 8$ for PBC: the computation time is lower for $J = 40, S = 8$ than for $J = 16, S = 2$ since PBC does not use Algorithm 3 but only runs Algorithm 2.

C. Number of Offloaders at Equilibrium

In this subsection, we analyze the number of offloaders at equilibrium. Figure 5 shows the average number of offloaders in equilibrium as a function of the number of WDs. We can observe that for each scenario, the number of offloaders is lowest at the Stackelberg equilibrium (SSE), and is highest when using PBC for the region $N < 40$. For $N > 40$ all curves converge to a similar average number of offloaders. At the same time, more offloaders does not imply higher profit for a given scenario: even though the number of offloaders is lowest in the Stackelberg equilibrium (SSE), the corresponding profit is highest (c.f. Fig. 3). These results confirm that caching, pricing and the strategic interactions of the WDs need to be jointly considered by the operator for utility maximization.

VI. Related Work

A number of recent works deal with energy efficient computation offloading for a single mobile user [19], [20], [21], [22], [23]. [19] proposes a system that enables energy-aware offloading to the infrastructure. Also the proposed algorithm maximizes energy savings with minimal computational burden. [20] proposed CPU frequency scaling and transmission power adaptation to optimize energy consumption of the computation of a task. [21] 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. [22] presents a dynamic offloading algorithm in order to achieve energy savings under time constraints. In [23], experimental results are used to show that battery power savings can be achieved using computation offloading.

A number of recent works proposed optimization approaches to minimizing the cost of task execution for multiple mobile devices [24], [25], [26], [27]. Authors in [24] 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 [25] 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 [26], and a heuristic was proposed to minimize the sum cost of mobile users. Authors in [27] consider the joint allocation of wireless and cloud resources and proposed an iterative algorithm to minimize users' energy consumption.

Another line of works provide a game theoretic treatment of the computation offloading problem [28], [29], [30], [31]. [28] 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. [29] 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. [30] 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. [31] 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.

Most related to ours are recent works that consider application caching and offloading [12], [32]. [12] 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. [32] considers the joint opti-

mization 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. Contrary to [12], [32], we treat the execution time as a constraint, and consider that the WDs strategically minimize their computation cost, leading to a new game formulation. We model the interactions between the WDs as a player-specific congestion game, we analyze the existence of equilibria, and we propose an algorithm for calculating the profit maximizing price for the operator in the resulting single leader multi-follower Stackelberg game.

VII. Conclusion

We have provided a game theoretic analysis pricing, application caching and computation offloading for edge computing. We showed that an equilibrium of offloading decisions and the 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. Our numerical results show that a simple greedy heuristic can be used for computing approximate Stackelberg equilibria with performance close to the actual equilibrium performance, at low computational complexity. Our analytical results about equilibrium existence open for a variety of avenues for future research, including the consideration of equilibria under incomplete information and strategies for minimizing regret in dynamic settings, which arguably are more accurate models of actual systems.

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, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 85–97, 2019.

[4] ——, "Joint wireless and edge computing resource management with dynamic network slice selection,," *IEEE/ACM Trans. on Networking*, to appear.

[5] ——, "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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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*, pp. 1–1, 2019.

[11] 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.

[12] 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.

[13] "Chapter 4 - an overview of digital communication and transmission," in *Wireless Communications Networking*, ser. The Morgan Kaufmann Series in Networking, V. K. Garg, Ed. Burlington: Morgan Kaufmann, 2007, pp. 85–122.

[14] I. Milchtaich, "The equilibrium existence problem in finite network congestion games," in *Internet and Network Economics*, P. Spirakis, M. Mavronicolas, and S. Kontogiannis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 87–98.

[15] "Congestion games with player-specific payoff functions," *Games and Economic Behavior*, vol. 13, no. 1, pp. 111–124, 1996.

[16] U. Feige, V. S. Mirrokni, and J. Vondrak, "Maximizing non-monotone submodular functions," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, 2007, pp. 461–471.

[17] 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.

[18] 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.

[19] 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, 10 2010, pp. 49–62.

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

[21] 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 *2013 Proceedings IEEE INFOCOM*, 2013, pp. 1285–1293.

[22] 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.

[23] 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.

[24] 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 (WoWMoM)*, 2015, pp. 1–9.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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 Communications (ICC)*, 2015, pp. 3192–3197.

[30] 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.

[31] 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.

[32] 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.