Open access • Proceedings Article • DOI:10.1109/INFOCOM.2017.8057148

A game theoretic analysis of selfish mobile computation offloading

Slacdana Josilo, Gyorgy Dan

Institutions: Royal Institute of Technology

Published on: 01 May 2017 - International Conference on Computer Communications

Topics: Computation offloading, Price of anarchy, Strategy, Nash equilibrium and Approximation algorithm

A Game Theoretic Analysis of Selfish Mobile Computation Offloading

Slađana Jošilo and György Dán
ACCESS Linnaeus Center, School of Electrical Engineering
KTH, Royal Institute of Technology, Stockholm, Sweden E-mail: {josilo, gyuri}@kth.se

***Abstract*—Offloading computation to a mobile cloud is a promising approach for enabling the use of computationally intensive applications by mobile devices. In this paper we consider autonomous devices that maximize their own performance by choosing one of many wireless access points for computation offloading. We develop a game theoretic model of the problem, prove the existence of pure strategy Nash equilibria, and provide a polynomial time algorithm for computing an equilibrium. For the case when the cloud computing resources scale with the number of mobile devices we show that all improvement paths are finite. We provide a bound on the price of anarchy of the game, thus our algorithm serves as an approximation algorithm for the global computation offloading cost minimization problem. We use extensive simulations to provide insight into the performance and the convergence time of the algorithms in various scenarios. Our results show that the equilibrium cost may be close to optimal, and the convergence time is almost linear in the number of mobile devices.**

I. Introduction

Computationally intensive applications, including augmented reality, natural language processing, face, gesture and object recognition, and various forms of user profiling for recommendations [1], [2] are increasingly used on mobile devices. Many of these applications consume a significant amount of energy, which can be detrimental to battery life, and together with potentially slow response times due to the limited computational power of the mobile handsets, it may limit user acceptance.

A promising approach to extend the battery lifetime of mobile handsets and to serve the computational needs of computationally intenstive applications is mobile cloud computing [3], [4]. Mobile cloud computing allows to offload the computations through a wireless access point to a cloud infrastructure with significant computational power. The computations can be performed in the cloud and the results sent back to the mobile handset. Commercial cloud infrastructures, such as Amazon EC2, may have plentyful computational resources, but they may not be able to provide sufficiently low response times for many applications. It may thus be better to offload computations to less resourceful mobile edge computing (MEC) infrastructures, which are considered an enabler of 5G mobile networks, as they are located close to the network edge [5].

The proximity of MEC resources to the network edge ensures low propagation times, but when many mobile devices attempt to offload computations simultaneously, the response times could be affected by the contention between the mobile devices for MEC computing resources and for wireless communication resources [6], [7]. The problem is inherently difficult for various reasons. First, the computational tasks of the mobile devices may have different complexities and may need the transmission of different amounts of data. Second, each device could aim at minimizing a combination of its response time and energy consumption for performing the computation. Third, the number of offloading choices for each mobile device increases with the number of access points. Thus, developing scalable algorithms that coordinate the offloading decisions of the mobile devices to ensure the efficient use of MEC resources and to provide predictable performance to the mobile devices is a challenging problem.

In this paper we address this problem by considering the allocation of cloud and wireless resources among mobile devices that can choose either to offload their computation to a cloud through one of many access points or to perform the computation locally. We make three important contributions to solve the problem. First, based on a game theoretical treatment of the problem, we propose an efficient distributed algorithm for coordinating the offloading decisions of the mobile devices, and prove convergence of the algorithm to a pure strategy Nash equilibrium when the computational capability assigned to a mobile device by the cloud is a non-increasing function of the number of mobile users that offload. Second, we show that a simple distributed algorithm can be used for computing equilibria when the cloud computing resources scale directly proportional with the number of mobile users. Finally, by establishing a bound on the price of anarchy of the strategic game, we show that the proposed algorithms have a bounded approximation ratio. We use extensive simulations to illustrate the computational efficiency of the algorithms and to evaluate their approximation ratio for scenarios of practical interest.

The rest of the paper is organized as follows. We present the system model in Section II. We present the algorithms and prove their convergence in Sections III and IV. We provide a bound on the approximation ratio in Section V and present numerical results in Section VI. Section VII discusses related work and Section VIII concludes the paper.

II. System Model and Problem Formulation

We consider a mobile cloud computing system that serves a set $\mathcal{N}=\{1,2,...,N\}$ of mobile users (MU). To facilitate the analysis, we make the assumption that the set of MUs changes slowly, e.g., in the order of seconds or minutes, similar to other works [4], [8], [9], [10]. Each MU has a computationally intensive task to perform, which

The work was partly funded by SSF through the Modane project and by the Swedish Research Council through project 621-2014-6.

978-1-5090-5336-0/17/$31.00 ©2017 IEEE

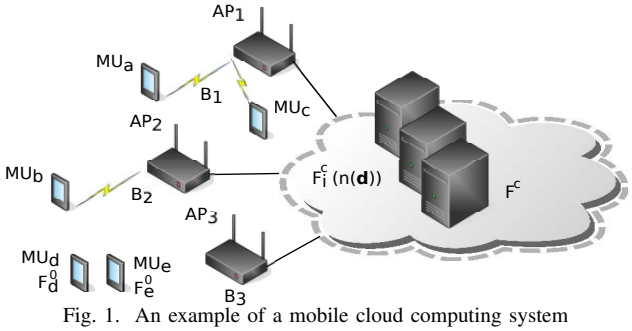

Fig. 1. An example of a mobile cloud computing system

is characterized by the size D_i of the input data (e.g., in bytes), and by the number L_i of CPU cycles required to perform the computation. Each MU can decide whether to perform the task locally or to offload the computation to a cloud server through one of a set of access points (APs) denoted by $\mathcal{A}=\{1,2,...,A\}$.

A. Communication model

If the MU decides to offload the computation to the cloud server, it has to transmit D_i amount of data pertaining to its task to the cloud server through one of the APs. Thus, together with local computing MU i can choose one element of the set $\mathfrak{D}_i=\{0,1,2,...,A\}$, where 0 corresponds to local computing. We denote by $d_i \in \mathfrak{D}_i$ the decision of MU i, and refer to it as her strategy. We refer to the collection $\mathbf{d}=(d_i)_{i\in\mathcal{N}}$ as a strategy profile, and we denote by $\mathfrak{D}=\times_{i\in\mathcal{N}}\mathfrak{D}_i$ the set of all feasible strategy profiles.

For a strategy profile $\mathbf{d}$ we denote by $n_a(\mathbf{d})$ the number of MUs that use AP a for computation offloading, and by $n(\mathbf{d})=\sum_{a\in\mathcal{A}} n_a(\mathbf{d})$ the number of MUs that offload. Similarily, for an AP $a \in \mathcal{A}$ we denote by $O_a(\mathbf{d}) = \{i|d_i = a\}$ the set of MUs that offload using AP a, and we define the set of offloaders as $O(\mathbf{d}) = \cup_{a\in\mathcal{A}} O_a(\mathbf{d})$.

We denote by $R_{i,a}$ the PHY rate of MU i on AP a, which depends on the physical layer signal characteristics and the corresponding channel gain. We denote by $\omega_{i,a}(\mathbf{d})$ the uplink rate that MU i receives when she offloads via AP a. $\omega_{i,a}(\mathbf{d})$ depends on the PHY rate $R_{i,a}$ and on the number $n_a(\mathbf{d})$ of MUs that offload via AP a

$$\omega_{i,a}(\mathbf{d}) = \frac{R_{i,a}}{n_a(\mathbf{d})}. \tag{1}$$

This model can be used to model the bandwidth sharing in TDMA and OFDMA based MAC protocols [11].

The uplink rate $\omega_{i,a}(\mathbf{d})$ together with the input data size D_i determines the transmission time $T_{i,a}^{c,off}(\mathbf{d})$ of MU i for offloading via AP a,

$$T_{i,a}^{c,off}(\mathbf{d}) = \frac{D_i}{\omega_{i,a}(\mathbf{d})}. \tag{2}$$

To model the energy consumption of the MUs, we assume that MU i uses a constant transmit power of P_i for sending the data, thus the energy consumption of MU i for offloading the input data of size D_i via AP a is

$$E_{i,a}^{c}(\mathbf{d}) = \frac{D_i P_i}{\omega_{i,a}(\mathbf{d})}. \tag{3}$$

B. Computation model

In what follows we introduce our model of the time and energy consumption in the case of local computing and in the case of computation offloading.

1) Local computing: In the case of local computing the task has to be processed using local computing resources. We denote by F_i^0 the computational capability of MU i, and hence the local computing time needed to perform its computation task that requires L_i CPU cycles can be expressed as

$$T_i^0 = \frac{L_i}{F_i^0}. \tag{4}$$

We consider that the energy consumption of local computing is proportional to the computation time, thus denoting by v_i the consumed energy per CPU cycle, we obtain

$$E_i^0 = v_i L_i. \tag{5}$$

2) Cloud computing: In the case of cloud computing, after the data are transmitted via an AP, processing is done at the cloud server. We denote by F^c the computation capability of the cloud, and we consider that the computation capability $F_i^c(n(\mathbf{d}))$, assigned to MU i by the cloud is a non-increasing function of the number $n(\mathbf{d})$ of MUs that offload. Given $F_i^c(n(\mathbf{d}))$ we use a linear model to compute the execution time of a task $<D_i, L_i>$ that is offladed by MU i,

$$T_i^{c,exe} = \frac{L_i}{F_i^c(n(\mathbf{d}))}. \tag{6}$$

Figure 1 shows an example of a mobile cloud computing system in which three of five MUs offload their tasks using one of three APs.

C. Cost Model

We model the cost of a MU as a linear combination of the time it takes to finish the computation and the corresponding energy consumption. We use the weights γ_i^T attributed to the time it takes to finish the computation and γ_i^E attributed to energy consumption, in order to model the delay sensitivity of the application and the consumed energy, respectively, $0 \le \gamma_i^E, \gamma_i^T \le 1$.

Using these notation, for the case of local computing the cost of MU i is determined by the local computing time and the consumed energy per CPU cycle

$$C_i^0 = \gamma_i^T T_i^0 + \gamma_i^E E_i^0 = (\frac{\gamma_i^T}{F_i^0} + \gamma_i^E v_i) L_i. \tag{7}$$

For the case of offloading via AP a, the cost of MU i is determined by the transmission time, the corresponding transmit energy, and the computing time in the cloud

$$\begin{aligned} C_{i,a}^c(\mathbf{d}) &= \gamma_i^T (T_i^{c,exe} + T_{i,a}^{c,off}(\mathbf{d})) + \gamma_i^E E_{i,a}^c(\mathbf{d}) \\ &= (\gamma_i^T + \gamma_i^E P_i)\frac{D_i}{\omega_{i,a}(\mathbf{d})} + \gamma_i^T \frac{L_i}{F_i^c(n(\mathbf{d}))}. \end{aligned} \tag{8}$$

Similar to previous works [7], [12], [13], we do not model the time needed to transmit the results of the computation from the cloud server to the MU, as for typical applications like face and speech recognition, the size of the result of the computation is much smaller than D_i.

To define the cost of MU i in strategy profile $\mathbf{d}$, let us introduce the indicator function $I(d_i, a)$ for MU i as

$$I(d_i, a) = \begin{cases} 1, & \text{if } d_i = a \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

d = *ImprovementPath*(**d**)

1: **while** $\exists i \in \mathcal{N}$ s.t. $\exists d_i', C_i(d_i', d_{-i}) < C_i(d_i, d_{-i})$ **do**
2: $d_i = d_i'$
3: **end while**
4: **return d**

Fig. 2. Pseudo code of the *ImprovementPath* algorithm.

We now express the cost of MU i in strategy profile **d** as

$$C_i(\mathbf{d}) = C_i^0 I(d_i, 0) + \sum_{a \in \mathcal{A}} C_{i,a}^c(\mathbf{d}) I(d_i, a). \quad (10)$$

Finally, we define the total cost $C(\mathbf{d}) = \sum_{i \in \mathcal{N}} C_i(\mathbf{d})$.

D. Computation Offloading Game

We consider that each MU aims at minimizing its cost (10), i.e., it aims at finding a strategy

$$d_i^* \in \arg\min_{d_i \in \mathfrak{D}_i} C_i(d_i, d_{-i}), \quad (11)$$

where we use d_{-i} to denote the strategies of all MUs except MU i. This problem formulation is not only reasonable when MUs are autonomous, but as we show later, our algorithms also serve as polynomial-time approximations for solving the problem of minimizing the total cost $C(\mathbf{d})$.

We thus consider that the MUs play a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$, in which the players are the MUs. We refer to the game as the *multi access point computation offloading game* (MCOG), and we are interested in whether the MUs can compute a strategy profile in which no MU can further decrease her cost by changing her strategy, i.e., a pure Nash equilibrium of the game Γ.

Definition 1. A Nash equilibrium (NE) of the strategic game $<\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$ is a strategy profile d^* such that

$$C_i(d_i^*, d_{-i}^*) \leq C_i(d_i, d_{-i}^*), \quad \forall d_i \in \mathfrak{D}_i.$$

Given a strategy profile (d_i, d_{-i}) we say that strategy d_i' is an improvement step for MU i if $C_i(d_i', d_{-i}) < C_i(d_i, d_{-i})$. We call a sequence of improvement steps in which one MU changes her strategy at a time, according to the *ImprovementPath* Algorithm shown in Figure 2, an *improvement path*. Furthermore, we say that a strategy d_i^* is a best reply to d_{-i} if it solves (11), and we call an improvement path in which all improvement steps are best reply a *best improvement path*. Observe that in a NE all MUs play their best replies to each others' strategies.

III. Equilibria and the JPBR Algorithm

We start the analysis with the definition of the set of congested APs and of the notion of the reluctance to offload.

Definition 2. For a strategy profile **d** we define the set $D_{O \to O}(\mathbf{d})$ of congested APs as the set of APs with at least one MU for which changing to another AP is a better reply,

$$D_{O \to O}(\mathbf{d}) = \{a \in \mathcal{A} | \exists i \in O_a(\mathbf{d}), \exists b \in \mathcal{A} \setminus \{a\}, \\ (n_b(\mathbf{d}) + 1)/R_{i,b} < n_a(\mathbf{d})/R_{i,a}\}.$$

Similarily, for a strategy profile **d** we define the set $D_{O \to L}(\mathbf{d})$ of APs with at least one MU for which local computing is a best reply,

$$D_{O \to L}(\mathbf{d}) = \{a \in \mathcal{A} | \exists i \in O_a(\mathbf{d}), C_{i,a}^c(\mathbf{d}) > C_i^0\}$$

Definition 3. The reluctance to offload via AP a of MU i in a strategy profile **d** is $\rho_i(\mathbf{d}) = \frac{C_{i,a}^c(\mathbf{d})}{C_i^0}$.

To facilitate the analysis, for a strategy profile **d** we rank the MUs that play the same strategy in decreasing order of their reluctance to offload, and we use the tuple (a, l) to index the MU that in the strategy profile **d** occupies position l in the ranking for AP a, i.e., $\rho_{(a,1)}(\mathbf{d}) \geq \rho_{(a,2)}(\mathbf{d}) \geq \ldots \geq \rho_{(a,n_a(\mathbf{d}))}(\mathbf{d})$. Note that for AP a it is MU $(a, 1)$ that can gain most by changing her strategy to local computing among all MUs $i \in O_a(\mathbf{d})$.

A. The ImproveAP Algorithm

Using these definitions, let us start with investigating whether the simple *ImprovementPath* algorithm can be used for computing a NE. To do so, we analyze the finiteness of improvement paths, and as a first step, we show that improvement paths may be infinite in the MCOG.

Example 1. *Consider a MCOG with $\mathcal{N} = \{a, b, c, d, e\}$ and $\mathcal{A} = \{1, 2, 3\}$ as illustrated in Figure 1. Figure 3 shows a cyclic improvement path starting from the strategy profile $(1, 2, 1, 0, 0)$, in which MUs a and c are connected to AP 1, MU b is connected to AP 2 and MUs d and e perform local computation.*

d_i	d_a	d_b	d_c	d_d	d_e
d(0)	1	2	1	0	0
d(1)	1	2	2	0	0
d(2)	1	0	2	0	0
d(3)	1	0	2	2	0
d(4)	1	0	2	2	2
d(5)	1	0	1	2	2
d(6)	1	3	1	2	2
d(7)	1	3	1	2	0
d(8)	1	3	1	0	0
d(9)	1	2	1	0	0

$R_{c,2} > R_{c,1}$ (1)
$\frac{2}{R_{b,2}}(\gamma_b^T + \gamma_b^E P_b) D_b + 3\gamma_b^T \frac{L_b}{F^c} > C_b^0$ (2)
$C_d^0 > \frac{2}{R_{d,2}}(\gamma_d^T + \gamma_d^E P_d) D_d + 3\gamma_d^T \frac{L_d}{F^c}$ (3)
$C_e^0 > \frac{3}{R_{e,2}}(\gamma_e^T + \gamma_e^E P_e) D_e + 4\gamma_e^T \frac{L_e}{F^c}$ (4)
$R_{c,1} > \frac{2}{3} R_{c,2}$ (5)
$C_b^0 > \frac{1}{R_{b,3}}(\gamma_b^T + \gamma_b^E P_b) D_b + 5\gamma_b^T \frac{L_b}{F^c}$ (6)
$\frac{2}{R_{e,2}}(\gamma_e^T + \gamma_e^E P_e) D_e + 5\gamma_e^T \frac{L_e}{F^c} > C_e^0$ (7)
$\frac{1}{R_{d,2}}(\gamma_d^T + \gamma_d^E P_d) D_d + 4\gamma_d^T \frac{L_d}{F^c} > C_d^0$ (8)
$R_{b,2} > R_{b,3}$ (9)

Fig. 3. A cyclic improvement path in a computation offloading game with 3 APs and 5 MUs. Rows correspond to strategy profiles, columns to MUs. An arrow between adjacent rows indicates the MU that performs the improvement step. The cycle consists of 9 improvement steps and the inequalities on the right show the condition under which the change of strategy is an improvement step.

Starting from the initial strategy profile $(1, 2, 1, 0, 0)$, MU c revises its strategy to AP 2, which is an improvement step if $R_{c,2} > R_{c,1}$, as shown in inequality (1) in the figure. Observe that after 9 improvement steps the MUs reach the initial strategy profile. For each step the inequality on the right provides the condition for being an improvement. It follows from inequalities (1), (5) and (9) that $R_{c,2} > R_{c,1}$, $R_{c,1} > \frac{2}{3} R_{c,2}$ and $R_{b,2} > R_{b,3}$, respectively. Since, $\frac{1}{R_{b,3}}(\gamma_b^T + \gamma_b^E P_b) D_b + 5\gamma_b^T \frac{L_b}{F^c} > \frac{1}{R_{b,3}}(\gamma_b^T + \gamma_b^E P_b) D_b + 3\gamma_b^T \frac{L_b}{F^c}$ holds, from inequalities (2) and (6) it follows that $R_{b,3} > \frac{1}{2} R_{b,2}$. Combining inequalities (3) and (8) we have that $\gamma_d^T \frac{L_d}{F^c} > \frac{1}{R_{d,2}}(\gamma_d^T + \gamma_d^E P_d) D_d$. Similarly, it follows from inequalities (4) and (7) that $\gamma_e^T \frac{L_e}{F^c} > \frac{1}{R_{e,2}}(\gamma_e^T + \gamma_e^E P_e) D_e$. Given these constraints, an instance of the example can be formulated easily, even in the case of homogeneous PHY rates, i.e., $R_{i,a} = R_{i',a}$ for every $i, i' \in \mathcal{N}$, $i \neq i'$. An important consequence of the cycle in the improvement path is that the MCOG does not allow a generalized ordinal potential function, and the *ImprovementPath* algorithm cannot be used for computing NE. Although improvement paths may cycle, as we next show, improvement paths are finite if we only allow the MUs to change between APs but not to start or to stop

$\mathbf{d} = \textit{ImproveAP}(\mathbf{d})$

1: **while** $D_{O\to O}(\mathbf{d}) \neq \emptyset$ **do**
2: $(i', a') \leftarrow \underset{\{i\in O(\mathbf{d}), \exists a\in\mathcal{A}, C_i(a,d_{-i})<C_i(\mathbf{d})\}}{\arg\max} \frac{C_i(\mathbf{d})}{C_i(a,d_{-i})}$
3: Let $\mathbf{d} = (a', d_{-i'})$
4: **end while**
5: **return d**

Fig. 4. Pseudo code of the *ImproveAP* algorithm.

offloading. We refer to this algorithm as the *ImproveAP* algorithm, and show its pseudo code in Figure 4. Our first result shows that all improvement paths generated by the *ImproveAP* algorithm are finite.

Lemma 1. *The ImproveAP algorithm terminates after a finite number of improvement steps.*

Proof. We prove the lemma by showing that the function

$$\Phi(\mathbf{d}) = \sum_{a'=1}^{A}\sum_{n=1}^{n_{a'}(\mathbf{d})} log(n) - \sum_{a'=1}^{A}\sum_{i'=1}^{N} log(R_{i',a'})I(d_{i'}, a').$$

decreases strictly at every improvement step generated by the *ImproveAP* algorithm.

Let us consider an improvement step made by MU i in which she changes from offloading via AP b to offloading via AP a. Observe that after this improvement step the number $n(\mathbf{d})$ of MUs that offload remains unchanged. Hence, according to (8) and (10), the condition $C_i(a, d_{-i}) < C_i(b, d_{-i})$ implies $n_a(a, d_{-i})/n_b(b, d_{-i}) < R_{i,a}/R_{i,b}$. Since $n_a(a, d_{-i}), n_b(b, d_{-i}), R_{i,a}, R_{i,b} > 0$ this is equivalent to

$$\log(n_a(a, d_{-i})) - \log(n_b(b, d_{-i})) < \log(R_{i,a}) - \log(R_{i,b}). \quad (12)$$

Let us rewrite Φ by separating the terms for APs a and b,

$$\Phi(a, d_{-i}) = \sum_{n=1}^{n_a(a,d_{-i})} \log(n) + \sum_{n=1}^{n_b(a,d_{-i})} \log(n) + \sum_{a'\neq a,b}\sum_{n=1}^{n_{a'}(a,d_{-i})} \log(n)$$
$$- \log(R_{i,a}) - \sum_{a'=1}^{A}\sum_{i'\neq i} \log(R_{i',a'})I(d_{i'}, a').$$

Since $n_a(a, d_{-i}) = n_a(b, d_{-i}) + 1$ and $n_b(b, d_{-i}) = n_b(a, d_{-i}) + 1$, we have that

$$\Phi(a, d_{-i}) - \Phi(b, d_{-i}) = \log(n_a(a, d_{-i})) - \log(n_b(b, d_{-i}))$$
$$- (\log(R_{i,a}) - \log(R_{i,b})).$$

It follows from (12) that $\Phi(a, d_{-i}) - \Phi(b, d_{-i}) < 0$. Since the number of strategy profiles is finite, $\Phi(\mathbf{d})$ can not decrease infinitely and the *ImproveAP* algorithm terminates after a finite number of improvement steps. □

Thus, if MUs can only change between APs, they terminate after a finite number of improvement steps.

B. The JPBR Algorithm

In what follows we use the *ImproveAP* algorithm as a building block for proving that a NE always exists in the MCOG even if it cannot be computed using the *ImprovementPath* algorithm.

Theorem 1. *The MCOG possesses a pure strategy Nash equilibrium.*

Proof. We use induction in the number N of MUs in order to prove the theorem. We denote by $N^{(t)} = t$ the number of MUs that are involved in the game in induction step t.

For $N^{(1)}=1$ the only participating MU plays her best reply $d_i^*(1)$. Since there are no other MUs, $\mathbf{d}^*(1)$ is a NE. Observe that if $d_i^*(1)=0$, MU i would never have an incentive to deviate from this decision, because the number of MUs that offload will not decrease as more MUs are added. Otherwise, if MU i decides to offload, she would play her best reply which is given by $d_i^*(1) = \arg\max_{a\in\mathcal{A}} R_{i,a}$.

Assume now that for $t-1>0$ there is a NE $\mathbf{d}^*(t-1)$. Upon induction step t one MU enters the game; we refer to this MU as MU $N^{(t)}$. Let MU $N^{(t)}$ play her best reply $d^*_{N^{(t)}}(t)$ with respect to the NE strategy profile of the MUs that already participated in induction step $t-1$, i.e., with respect to $d_{-N^{(t)}}(t) = \mathbf{d}^*(t-1)$. After that, MUs can perform best improvement steps one at a time starting from the strategy profile $\mathbf{d}(t) = (d^*_{N^{(t)}}(t), d_{-N^{(t)}}(t))$, following the algorithm shown in Figure 5. We refer to this as the update phase. In order to prove that there is a NE in induction step t, in the following we show that the MUs will perform a finite number of best improvement steps in the update phase.

Observe that if $d^*_{N^{(t)}}(t) = 0$, then $n_a(\mathbf{d}(t)) = n_a(\mathbf{d}^*(t-1))$ for every $a \in \mathcal{A}$ and thus $\mathbf{d}(t)$ is a NE. If $d^*_{N^{(t)}}(t) = a \in \mathcal{A}$, but none of the MUs want to deviate from their strategy in $\mathbf{d}^*(t-1)$ then $\mathbf{d}(t)$ is a NE. Otherwise, we can have one or both of the following cases: (i) for some MUs $i \in O_a(\mathbf{d}(t))$ offloading using AP $b \in \mathcal{A} \setminus \{a\}$ becomes a best reply, (ii) for some MUs $i \in O(\mathbf{d}(t))$ local computing becomes a best reply.

Let us first consider case (i) and let MUs execute the *ImproveAP* algorithm. Recall that by Lemma 1 the MUs will reach a strategy profile in which there is no MU that can further decrease her cost by changing her strategy between APs. In the resulting strategy profile the number of MUs that offload will be $n(\mathbf{d}^*(t-1))+1$. Furthermore, there will be one AP (denoted by a') for which the number of offloaders is $n_{a'}(\mathbf{d}^*(t-1))+1$, while for the other APs $a \neq a'$ it is $n_a(\mathbf{d}^*(t-1))$. As a consequence, there can be no MU that wants to start offloading in the resulting strategy profile if she did not want to do so in $\mathbf{d}^*(t-1)$.

If in this strategy profile no MU wants to stop offloading either, i.e., $|D_{O\to L}(\mathbf{d}(t))| = 0$, then we reached a NE. Otherwise $|D_{O\to L}(\mathbf{d}(t))| > 0$, which is the same as case (ii) above. Note that if case (i) did not happen, i.e. $|D_{O\to O}\mathbf{d}(t)| = 0$, then AP a' is the same AP a that was chosen by MU $N^{(t)}$ when it was added. Now if $a' \in D_{O\to L}(\mathbf{d}(t))$, let MU $(a', 1)$ perform an improvement step and let $\mathbf{d}'(t)$ be the resulting strategy profile. Since MU $(a', 1)$ changed her strategy from AP a' to local computation, $n_a(\mathbf{d}'(t)) = n_a(\mathbf{d}^*(t-1))$ holds for every AP $a \in \mathcal{A}$ and $\mathbf{d}'(t)$ is a NE.

Otherwise, if $a' \notin D_{O\to L}$ and $|D_{O\to L}| > 0$, we have that there is MU i that wants to change her strategy from offloading through AP $b \in \mathcal{A}\setminus\{a'\}$ to local computing. Note that the only reason why MU i would want to change to local computing is that the number of MUs that offload was incremented by one, i.e., $n(\mathbf{d}(t)) = n(\mathbf{d}^*(t-1))+1$. Among all MUs that would like to change to local computing, let

us allow the MU i with highest reluctance to perform the improvement step (note that this is MU $(b,1)$, $b \neq a'$). We denote the resulting strategy profile by $\mathbf{d}'(t)$. Due to this improvement step $n_b(\mathbf{d}'(t)) = n_b(\mathbf{d}^*(t-1)) - 1$, and thus some MUs that perform local computation may be able to decrease their cost by connecting to AP b. If there is no MU $i \in \mathcal{N} \setminus O(\mathbf{d}'(t))$ that would like to start offloading, then there is no more MU that would like to stop offloading either because $n(\mathbf{d}'(t)) = n(\mathbf{d}^*(t-1))$. Otherwise, among all MUs $i \in \mathcal{N} \setminus O(\mathbf{d}'(t))$ that would like to start offloading, let MU i' with lowest reluctance to offload, i.e., $\rho_{i'}(b, d'_{-i'}(t))$, connect to AP b. We now repeat these steps starting from Line 8 until no more MUs want to stop offloading. Note that when one MU is replaced by another MU, the number of MUs that offload through any of the APs does not change. Therefore, offloading cost of the MU that starts to offload will not increase in the following update steps and she will not want to stop to offload. Since the MU that starts to offload will not have an incentive to stop to offload and the number of MUs is finite, the sequence of stopping to offload and starting to offload is finite too.

Let b be the AP that the last MU that stopped offloading was connected to. If the last MU that stopped offloading was replaced by a MU that did not offload before, then we reached a NE. Otherwise some MUs that offload via AP $a \in \mathcal{A} \setminus \{b\}$ may want to connect to AP b, and we let them execute the *ImproveAP* algorithm, which by Lemma 1 terminates in a finite number of improvement steps. Now, no MU wants to stop offloading, and if there is no MU that wants to start offloading either then we reached a NE. Otherwise if there is a MU that wants to start to offload, we repeat the steps starting from Line 8. Let us recall that the MU that starts to offload would not want to stop to offload and as a consequence the size of the set $D_{O \to L}$ will decrease every time when a MU stops to offload. Therefore, after a finite number of steps, the MUs will reach either an equilibrium in which the number of offloaders is the same as in the strategy profile $\mathbf{d}^*(t-1)$ or an equilibrium in which the number of offloaders is incremented by 1, which proves the inductive step. □

Consider now that we add one MU at a time and for every new MU we compute a NE following the proof of Theorem 1. We refer to the resulting algorithm as the *Join and Play Best Replies (JPBR)* algorithm. In what follows we provide a bound on the complexity of this algorithm.

Proposition 1. *When MU $N^{(t)}$ enters the game in an equilibrium $\mathbf{d}^*(t-1)$, a new Nash equilibrium can be computed in $O((A+2)N^{(t)} - 2A)$ time.*

Proof. In the worst case scenario $|O(\mathbf{d}^*(t-1))| = N^{(t)} - 2$, $d^*_{N^{(t)}}(t) = a \in \mathcal{A}$ and case (ii) happens such that in the next $N^{(t)} - 2$ update steps all MUs $i \in O(\mathbf{d}^*(t-1))$, i.e., $N^{(t)} - 2$ MUs change between APs before they reach the strategy profile in which there is no MU that can decrease her offloading cost by choosing another AP. Furthermore, in the worst case scenario, this is followed by a sequence of update steps in which $N^{(t)} - 2$ MUs stop to offload and $N^{(t)} - 3$ MUs start to offload and between every stop to offload and start to offload update step, MUs change between the APs. When a MU stops to offload, the sequence in which MUs change between APs consists of at most $A-1$ update steps. Therefore, a NE is reached after at most $(N^{(t)} - 2) + (N^{(t)} - 2) + (N^{(t)} - 3) + (N^{(t)} - 2)(A-1)$ update steps. □

Update phase of JPBR algorithm

1: /* Corresponds to case (i) */
2: Let $\mathbf{d}'(t) = \text{ImproveAP}(\mathbf{d}(t))$
3: /* Corresponds to case (ii) */
4: **if** $a' \in D_{O \to L}(\mathbf{d}'(t)), n_{a'}(\mathbf{d}'(t)) = n_{a'}(\mathbf{d}^*(t-1)) + 1$ **then**
5: Let $i' \leftarrow (a', 1)$
6: Let $\mathbf{d}'(t) = (0, d'_{-i'}(t))$/* Best reply by MU i' */
7: **else**
8: **while** $D_{O \to L}(\mathbf{d}'(t)) \neq \emptyset$ **do**
9: $b \leftarrow \arg\max_{a \in D_{O \to L}} \rho_{(a,1)}(\mathbf{d}'(t))$
10: /* AP with MU with highest reluctance to offload */
11: Let $i' \leftarrow (b, 1)$
12: Let $\mathbf{d}'(t) = (0, d'_{-i'}(t))$
13: /* Best reply by MU $(b,1)$ */
14: **if** $\exists i \in \mathcal{N} \setminus O(\mathbf{d}'(t))$ s.t. $C_i^0 > C_i(b, d'_{-i}(t))$ **then**
15: $i' \leftarrow \arg\min_{\{i \in \mathcal{N} \setminus O(\mathbf{d}'(t)) | C_i^0 > C_i(b, d'_{-i}(t))\}} \rho_i(b, d'_{-i}(t))$
16: /*MU with lowest reluctance to offload*/
17: Let $\mathbf{d}'(t) = (b, d'_{-i'}(t))$ /* Best reply by MU i' */
18: **else**
19: Let $\mathbf{d}'(t) = \text{ImproveAP}(\mathbf{d}'(t))$
20: **end if**
21: **end while**
22: **end if**

Fig. 5. Pseudo code of the update phase of the JPBR algorithm.

Since we add one MU at a time, we can formulate the following result.

Corollary 1. *The* JPBR *algorithm terminates in an equilibrium allocation in $O((A+2)N^2/2 - (A-1)N)$ time.*

So far we have shown that starting from a NE and adding a new MU, a new NE can be computed. We now show a similar result for the case when a MU leaves.

Theorem 2. *Consider the MCOG and assume that the system is in a NE. If a MU leaves the game and the remaining MUs play their best replies one at a time, they converge to a NE after a finite number of updates.*

Proof. Let us consider that MU i leaves the game when the system is in a NE. If the strategy of MU i was to perform local computation, none of the remaining MUs would have an incentive to change their strategies. If the strategy of MU i was to offload using one of the APs, we can consider MU i as a MU that after changing its strategy from offloading to local computing would have no incentive to offload again. Recall from the proof of Theorem 1 that when a MU changes her strategy from offloading to local computing the game converges to a NE after a finite number of updates. This proves the theorem. □

Observe that Theorem 1 and Theorem 2 allow for the efficient computation of Nash equilibria even if the number of MUs changes, if the time between MU arrivals and departures is sufficient to compute a new equilibrium. Furthermore, the computation can be done in a decentralized

manner, by letting MUs perform best improvements one at a time. The advantage of such a decentralized implementation compared to a centralized solution could be that MUs do not have to reveal their parameters.

IV. The Case of an Elastic Cloud

The JPBR algorithm can be used for computing an equilibrium for MCOG with polynomial complexity. In what follows we show that a much simpler algorithm can be used for computing an equilibrium if the cloud computation capability assigned to MU i is independent of the other players' strategies, $F_i^c(n(\mathbf{d})) = F^c$, and thus of the number of MUs that offload. This model can be relevant for large cloud computing infrastructures, in which the cloud computing resources scale with the number of MUs, and we refer to this model as the *elastic* cloud model. In the case of the *elastic* cloud model the cost function in the case of offloading can be expressed as

$$C_{i,a}^c(\mathbf{d}) = (\gamma_i^T + \gamma_i^E P_i) D_i \frac{n_a(\mathbf{d})}{R_{i,a}} + \gamma_i^T \frac{L_i}{F^c}. \quad (13)$$

Before we formulate the theorem, let us recall the definition of a generalized ordinal potential from [14].

Definition 4. A function $\Phi : \times\mathfrak{D}_i \rightarrow \mathbb{R}$ is a generalized ordinal potential function for the strategic game $< \mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i >$ if for an arbitrary strategy profile (d_i, d_{-i}) and for any corresponding improvement step d_i' it holds that

$$C_i(d_i', d_{-i}) - C_i(d_i, d_{-i}) < 0 \Rightarrow \Phi(d_i', d_{-i}) - \Phi(d_i, d_{-i}) < 0.$$

Theorem 3. *The MCOG with elastic cloud admits the generalized ordinal potential function*

$$\Phi(\boldsymbol{d}) = \sum_{a'=1}^{A} \sum_{n=1}^{n_{a'}(\boldsymbol{d})} log(n) - \sum_{a'=1}^{A} \sum_{i'=1}^{N} log(M_{i'} R_{i',a'}) I(d_{i'}, a'), \quad (14)$$

and hence it possesses a pure strategy Nash equilibrium.

Proof. To prove that $\Phi(\mathbf{d})$ is a generalized ordinal potential function, we first show that $C_i(a, d_{-i}) < C_i(0, d_{-i})$ implies $\Phi(a, d_{-i}) < \Phi(0, d_{-i})$.

According to (7), (10) and (13), the condition $C_i(a, d_{-i}) < C_i(0, d_{-i})$ implies that

$$(\gamma_i^T + \gamma_i^E P_i) D_i \frac{n_a(a, d_{-i})}{R_{i,a}} + \gamma_i^T \frac{L_i}{F^c} < (\frac{\gamma_i^T}{F_i^0} + \gamma_i^E v_i) L_i.$$

After algebraic manipulations we obtain

$$\frac{n_a(a, d_{-i})}{R_{i,a}} < M_i \triangleq \frac{\gamma_i^E v_i + \gamma_i^T (\frac{1}{F_i^0} - \frac{1}{F^c})}{\gamma_i^T + \gamma_i^E P_i} \cdot \frac{L_i}{D_i}. \quad (15)$$

Since $n_a(a, d_{-i}) > 0$ and $M_i R_{i,a} > 0$, (15) implies that

$$\log(n_a(a, d_{-i})) < \log(M_i R_{i,a}). \quad (16)$$

For the strategy profile (a, d_{-i}) it holds that

$$\Phi(a, d_{-i}) = \sum_{n=1}^{n_a(a,d_{-i})} \log(n) + \sum_{a' \neq a} \sum_{n=1}^{n_{a'}(a,d_{-i})} \log(n) - \log(M_i R_{i,a}) - \sum_{a'=1}^{A} \sum_{i' \neq i} \log(M_{i'} R_{i',a'}) I(d_{i'}, a'),$$

and for the strategy profile $(0, d_{-i})$

$$\Phi(0, d_{-i}) = \sum_{n=1}^{n_a(0,d_{-i})} \log(n) + \sum_{a' \neq a} \sum_{n=1}^{n_{a'}(0,d_{-i})} \log(n) - \sum_{a'=1}^{A} \sum_{i' \neq i} \log(M_{i'} R_{i',a'}) I(d_{i'}, a').$$

Since $n_a(a, d_{-i}) = n_a(0, d_{-i}) + 1$, we obtain $\Phi(a, d_{-i}) - \Phi(0, d_{-i}) = \log(n_a(a, d_{-i})) - \log(M_i R_{i,a})$. It follows from (16) that $\Phi(a, d_{-i}) - \Phi(0, d_{-i}) < 0$. Similarly, we can show that $C_i(0, d_{-i}) < C_i(a, d_{-i})$ implies $\Phi(0, d_{-i}) < \Phi(a, d_{-i})$.

Second, we show that $C_i(a, d_i) < C_i(b, d_i)$ implies $\Phi(a, d_i) < \Phi(b, d_i)$. According to (10) and (13), the condition $C_i(a, d_i) < C_i(b, d_i)$ implies that

$$(\gamma_i^T + \gamma_i^E P_i) D_i \frac{n_a(a, d_{-i})}{R_{i,a}} < (\gamma_i^T + \gamma_i^E P_i) D_i \frac{n_b(b, d_{-i})}{R_{i,b}}$$

which is equivalent to

$$n_a(a, d_{-i}) / n_b(b, d_{-i}) < R_{i,a} / R_{i,b}. \quad (17)$$

Since $n_a(a, d_{-i}), n_b(b, d_{-i}), R_{i,a}, R_{i,b} > 0$ (17) implies

$$\log(n_a(a, d_{-i})) - \log(n_b(b, d_{-i})) < \log(R_{i,a}) - \log(R_{i,b}). \quad (18)$$

Let us rewrite Φ by separating the terms for APs a and b,

$$\Phi(a, d_{-i}) = \sum_{n=1}^{n_a(a,d_{-i})} \log(n) + \sum_{n=1}^{n_b(a,d_{-i})} \log(n) + \sum_{a' \neq a,b} \sum_{n=1}^{n_{a'}(a,d_{-i})} \log(n) - \log(M_i R_{i,a}) - \sum_{a'=1}^{A} \sum_{i' \neq i} \log(M_{i'} R_{i',a'}) I(d_{i'}, a').$$

Since $n_a(a, d_{-i}) = n_a(b, d_{-i}) + 1$ and $n_b(b, d_{-i}) = n_b(a, d_{-i}) + 1$, we have that $\Phi(a, d_{-i}) - \Phi(b, d_{-i}) = \log(n_a(a, d_{-i})) - \log(n_b(b, d_{-i})) - (\log(R_{i,a}) - \log(R_{i,b}))$. It follows from (18) that $\Phi(a, d_{-i}) - \Phi(b, d_{-i}) < 0$, which proves the theorem. □

It is well known that in a finite strategic game that admits a generalized ordinal potential all improvement paths are finite [14]. Therefore, the existence of a generalized ordinal potential function allows us to use the *ImprovementPath* Algorithm (c.f., Figure 2) for computing a NE.

Corollary 2. *The* ImprovementPath *algorithm terminates in a NE after a finite number of improvement steps for the MCOG with elastic cloud.*

V. Price of Anarchy

We have so far shown that NE exists and provided low complexity algorithms for computing an NE. We now address the important question how far the system performance would be from optimal in a NE. To quantify the difference from the optimal performance we use the price of anarchy (PoA), defined as the ratio of the worst case NE cost and the minimal cost

$$PoA = \frac{\max_{\mathbf{d}^*} \sum_{i \in \mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d} \in \mathfrak{D}} \sum_{i \in \mathcal{N}} C_i(\mathbf{d})}. \quad (19)$$

In what follows we give an upper bound on the PoA.

Theorem 4. *The price of anarchy for the computation offloading game is upper bounded by*

$$\frac{\sum_{i\in\mathcal{N}} C_i^0}{\sum_{i\in\mathcal{N}} \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}\}}.$$

Proof. First we show that if there is a NE in which all MUs perform local computation then it is the worst case NE. To show this let $\mathbf{d}^*$ be an arbitrary NE. Observe that $C_i(d_i^*, d_{-i}^*) \leq C_i^0$ holds for every player $i \in \mathcal{N}$. Otherwise, if $\exists i \in \mathcal{N}$ such that $C_i(d_i^*, d_{-i}^*) > C_i^0$, player i would have an incentive to deviate from decision d_i^*, which contradicts our initial assumption that $\mathbf{d}^*$ is a NE. Thus in any NE $\sum_{i\in\mathcal{N}} C_i(d_i^*, d_{-i}^*) \leq \sum_{i\in\mathcal{N}} C_i^0$, and if all MUs performing local computation is a NE then it is the worst case NE.

Now we derive a lower bound for the optimal solution of the computation offloading game. Let us consider an arbitrary decision profile $(d_i, d_{-i}) \in \mathfrak{D}$. If $d_i = 0$, then $C_i(d_i, d_{-i}) = C_i^0$. Otherwise, if $d_i = a$ for some $a \in \mathcal{A}$, we have that in the best case $d_{i'} = 0$ for every $i' \in \mathcal{N} \setminus \{i\}$, and thus $n(\mathbf{d}) = 1$. Therefore, $\omega_{i,a}(d_i, d_{-i}) \leq R_{i,a}$ and $F_i^c(n(d_i, d_{-i})) \leq F^c$, which implies that

$$C_{i,a}^c(d_i, d_{-i}) = (\gamma_i^T + \gamma_i^E P_i)\frac{D_i}{\omega_{i,a}(d_i, d_{-i})} + \gamma_i^T \frac{L_i}{F_i^c(n(d_i, d_{-i}))}$$
$$\geq (\gamma_i^T + \gamma_i^E P_i)\frac{D_i}{R_{i,a}} + \gamma_i^T \frac{L_i}{F^c} = C_{i,a}^{\bar{c}}.$$

Hence, we have $C_i(d_i, d_{-i}) \geq \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}\}$ and $\sum_{i\in\mathcal{N}} C_i(d_i, d_{-i}) \geq \sum_{i\in\mathcal{N}} \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}\}$. Using these we can establish the following bound

$$PoA = \frac{\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d}\in\mathfrak{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})} \leq \frac{\sum_{i\in\mathcal{N}} C_i^0}{\sum_{i\in\mathcal{N}} \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}\}},$$

which proves the theorem. □

VI. Numerical Results

We use extensive simulations to evaluate the cost performance and the computational time of the *JPBR* algorithm. We consider that the APs and MUs are placed over a $1km \times 1km$ region. The APs are located at grid points in the region, while the MUs are placed uniformly at random. We consider that the channel gain of MU i to AP a is proportional to $d_{i,a}^{-\alpha}$, where $d_{i,a}$ is the distance between MU i and AP a, and α is the path loss exponent, which we set to 4 according to the path loss model in urban and subrurban areas [15]. The channel bandwidth B_a of every AP a was set to $\mu = 5$ *MHz*, while the data transmit power P_i of every MU i was set to to 0.4*W* according to [16]. The computational capability F_i^0 of MU i was drawn from a continuous uniform distribution on $[0.5, 1]$ *Gcycles*, while the computation capability of the cloud F^c was set to 100 *Gcycles* [17]. Unless otherwise noted, the input data size D_i and the number L_i of CPU cycles required to perform the computation are uniformly distributed on $[0.42, 2]$ *Mb* and $[0.1, 0.8]$ *Gcycles*, respectively. The consumed energy per CPU cycle v_i was set to $10^{-11}(F_i^0)^2$ according to measurements reported in [4], [18]. The weights attributed to energy consumption γ_i^E and the response time γ_i^T were drawn from a continuous uniform distribution on $[0, 1]$.

In order to evaluate the cost performance of the equilibrium strategy profile $\mathbf{d}^*$ computed by the *JPBR* algorithm, we computed the optimal strategy profile $\bar{\mathbf{d}}$ that minimizes the total cost, i.e., $\bar{\mathbf{d}} = \arg\min_{\mathbf{d}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})$. Furthermore, as a baseline for comparison we use the system cost that can be achieved if all MUs execute their computation tasks locally, which coincides with the bound on the PoA.

A. Optimal vs. Equilibrium Cost

Figure 6 shows the *cost ratio* $C(\mathbf{d}^*)/C(\bar{\mathbf{d}})$ vs. the number of MUs. The results are shown for the case of the *elastic* cloud as well as for the case when the cloud computational capability assigned to a MU that offloads is a reciprocal function of the number of MUs that offload, i.e. $F_i^c(\mathbf{d}) = \frac{F^c}{an(\mathbf{d})}$. We refer to this latter case as a *non-elastic* cloud and to the coefficient a as the *cloud provisioning coefficient*. A coefficient of $a = 1$ corresponds to a cloud with fixed amount of resources, $a < 1$ to resources that scale slower than the demand, while $a > 1$ corresponds to a cloud with backup resources that scale with the demand. To make the computation of the optimal strategy profile $\bar{\mathbf{d}}$ feasible, unless otherwise noted, we considered a scenario with $A = 3$ APs and we show the *cost ratio* $C(\mathbf{d}^*)/C(\bar{\mathbf{d}})$ as a function of the number of MUs. We consider the *non-elastic* cloud model that does not implement redundancy mechanisms for three values of the *cloud provisioning coefficient* ($a = 0.5, 1$ and 2).

The results in Fig. 6 show that the performance of *JPBR* is close to optimal (*cost ratio* is close to 1) in all cases, and the *cost ratio* is fairly insenstive to the number of MUs, which is due to the number of MUs that choose to offload, as we will see later. The results for the bound on the PoA additionally confirm that the *JPBR* algorithm performs good in terms of the *cost ratio*. It is interesting to note that the gap between the PoA bound and the actual *cost ratio* decreases with increasing number of MUs. This is due to the benefit of offloading decreases as the number of MUs increases, and as a result the optimal solution and the *JPBR* algorithm will converge to a strategy profile in which most of the MUs perform local computation. We can also observe that the upper bound on the PoA decreases as a increases, and thus the problem becomes computationally easier for larger values of a.

In order to gain insight in the structure of the equilibrium strategy profiles $\mathbf{d}^*$, it is interesting to compare the number of MUs that offload in equilibrium $\mathbf{d}^*$ and the number of MUs that offload in the optimal solution $\bar{\mathbf{d}}$. We define the *offloading difference ratio* $(n(\mathbf{d}^*) - n(\bar{\mathbf{d}}))/N$, and show it in Figure 7 for the same set of parameters as in Figure 6. The results show that the *offloading difference ratio* increases with the number of MUs, which explains the increased *cost ratio* observed in Figure 6, as more offloaders reduce the achievable rate, which in turn leads to increased costs. The observation that the number of MUs that offload is higher in equilibrium than in the optimal solution is consistent with the theory of the tragedy of the commons in the economic literature [19]. The results also show that the *offloading difference ratio* is slightly lower in the case of the *elastic*

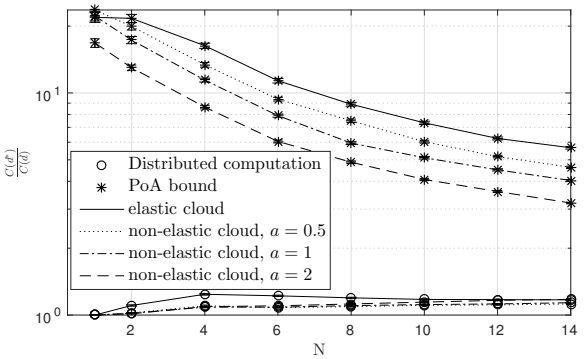

Fig. 6. The *cost ratio* and the upper bound on the PoA for the *elastic* and *non-elastic* cloud ($a = 0.5, 1, 2$), $A = 3$ APs. The results shown are the averages of 600 simulations, together with 95% confidence intervals.

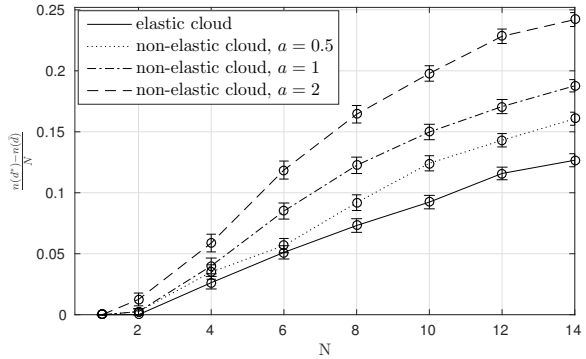

Fig. 7. *Offloading difference ratio* vs. number of MUs N for the *elastic* and *non-elastic* cloud ($a = 0.5, 1, 2$), $A = 3$ APs. The results shown are the averages of 600 simulations, together with 95% confidence intervals.

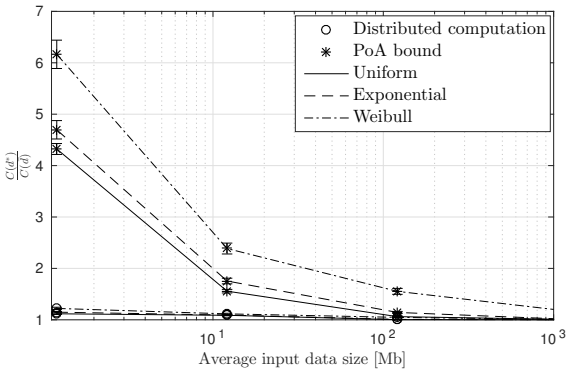

Fig. 8. The *cost ratio* and the upper bound on the PoA for the *non-elastic* cloud ($a = 1$), uniform, exponential and Weibull distributions of the input data sizes, $A = 3$ APs, $N = 12$ MUs. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

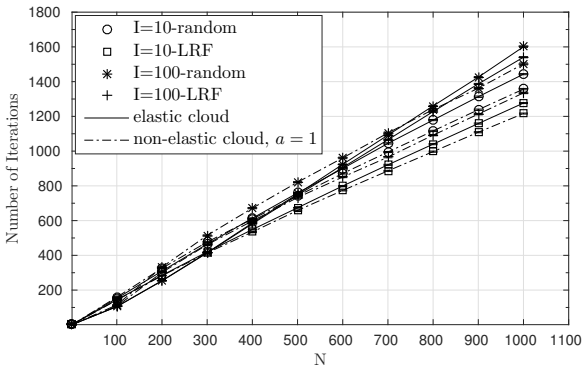

Fig. 9. Number of iterations vs. number of MUs N for the *elastic* and *non-elastic* cloud ($a = 1$), A =10 and 100 APs. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

cloud, which is due that a higher proportion of MUs offload in the optimal solution for the *elastic* cloud.

B. Impact of the input data size

In order to analyse the impact of the input data size we considered three distributions with the same mean for the input data size, uniform (lower limit fixed to 0.42 and upper limit scales with the mean), exponential, and Weibull (shape parameter 0.5), and considered that all MUs have to offload a task that requires a computation of $L_i = 0.45$ *Gcycles*. Figure 8 shows the *cost ratio* $C(\mathbf{d}^*)/C(\bar{\mathbf{d}})$ and the upper bound on the PoA as a function of the mean input data size. The results are shown for the *non-elastic* cloud (a=1), $N = 12$ MUs and $A = 3$ APs, and show that while the *cost ratio* does not change, the upper bound on the PoA decreases with the mean input data size and for large data sizes it reaches the *cost ratio*. This is due to the transmission time increases with the input data size and if the MUs have to offload a large amount of data, it becomes optimal for most of them to perform local computation, which coincides with the worst case equilibrium. Note that the upper bound on the PoA decreases slower in the case of the Weibull distribution because for the same mean it has a median that is smaller than that of the uniform and exponential distributions.

C. Computational Complexity

In order to evaluate the computational complexity of the *JPBR* algorithm, we consider the number of iterations, the total number of update steps over all induction steps plus the number of induction steps, to compute the strategy profile $\mathbf{d}^*$ for the *elastic* cloud and for the *non-elastic* cloud ($a = 1$), A=10 and A=100 APs. Figure 9 shows the number of iterations as a function of the number of MUs for two orderings of adding MUs: in the first case the MUs are added in random order, while in the second case the MUs are added in increasing order of their ratio $\frac{D_i}{C_i^0 L_i}$. We refer to the latter as the *least reluctance first* (LRF) order. Intuitively, one would expect that the LRF order results in a smaller number of iterations, since the MUs with lower $\frac{D_i}{C_i^0 L_i}$ ratio have lower computational capability to execute computationally more demanding tasks with smaller offloading data size than the MUs with higher $\frac{D_i}{C_i^0 L_i}$. However, the simulation results show that the number of iterations is fairly insensitive to the order of adding the MUs and mostly depends on the number of MUs. This insensitivity allows for a very low-overhead decentralized solution, as the coordinator need not care about the order in which the MUs are added for computing the equilibrium allocation. The results also show that the number of iterations scales approximately linearly with the number of MUs, and indicates that the worst case scenario described in Corollary1 is unlikely to happen. Thus *JPBR* is an efficient decentralized algorithm for coordinating computation offloading among autonomous MUs.

VII. Related Work

Most previous works considered the problem of energy efficient computation offloading for a single mobile user [3], [4], [6], [20], [21], and thus they do not consider the allocation of resources between mobile users.

Some recent works considered the problem of energy efficient computation offloading for multiple mobile users [8], [22], [9]. [8] studied the partitioning problem for mobile data stream applications, and proposed a heuristic for solv-

ing the optimization problem that maximizes throughput. [22] considered a two-tiered cloud infrastructure under user mobility in a location-time workflow framework, and proposed a heuristic for minimizing the users' cost. [9] provided an iterative algorithm for solving the optimization problem that minimizes the mobile users' energy consumption by joint allocation of wireless and cloud resources.

A few recent works provided a game theoretic treatment of the computation offloading problem [23], [24], [7], [25], [26], [27]. [23] considered a two-stage problem, where first each mobile user decides which parts of a task to offload so as to minimize its energy consumption and to meet its service response deadline, and then the cloud allocates computational resources to the offloaded tasks. [24] considered a three-tier cloud architecture and stochastic task arrivals, and provided a distributed algorithm for the computation of an equilibrium. [26] considered tasks that arrive at the same time, a single wireless link, and elastic cloud, and showed the existence of equilibria when all mobile users have the same delay budget. Our work differs from [23] in that we consider that the allocation of cloud resources is known to the mobile users, from [24] in that we consider contention in the wireless access, and from [26] in that we consider multiple wireless links and a non-elastic cloud.

Most related to our work are [7], [25], [27]. [7] considered a single wireless link and an elastic cloud, assumed upload rates to be determined by the Shannon capacity of an interference channel, and showed that the game is a potential game. [25] extended the model to multiple wireless links and showed that the game is still a potential game under the assumption that a mobile user experiences the same channel gain for all links. Unlike these works, we consider time-fair bandwidth sharing and the case of a non-elastic cloud. [27] considered multiple wireless links, fair bandwidth sharing and a non-elastic cloud, and claims the game to have an exact potential.

The importance of our contribution from a game theoretical perspective is that the computation offloading game with non-elastic cloud is a player-specific congestion game for which the existence of equilibria is not known in general [28], thus the JPBR algorithm and our proof of equilibrium existence advance the state of the art in the study of equilibria in general congestion games.

VIII. Conclusion

We have provided a game theoretic analysis of selfish mobile computation offloading. We proposed a polynomial complexity algorithm for computing equilibrium allocations of the wireless and cloud resources, and provided a bound on the price of anarchy, which serves as an approximation ratio bound for the optimization problem. Our numerical results show that the proposed algorithms and the obtained equilibria provide good system performance irrespective of the number of mobile users and access points, for various distributions of the input data size and task complexity, and confirm the low complexity of the proposed algorithms.

References

[1] M. Hakkarainen, C. Woodward, and M. Billinghurst, "Augmented assembly using a mobile phone," in *Proc. of IEEE/ACM ISMAR*, Sept 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 *Proc. of IEEE PerCom*, March 2009, pp. 1–9.

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. of ACM MobiSys*, 2010, pp. 49–62.

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

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," Sep. 2015.

[6] 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. of IEEE INFOCOM*, April 2013, pp. 1285–1293.

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, 2015.

[8] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Apr. 2013.

[9] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE T-SIPN*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[10] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, "An iterative double auction for mobile data offloading," in *Proc. of WiOpt*, May 2013, pp. 154–161.

[11] T. Joshi, A. Mukherjee, Y. Yoo, and D. P. Agrawal, "Airtime fairness for ieee 802.11 multirate networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 4, pp. 513–527, 2008.

[12] 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, Jun. 2012.

[13] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer Mag.*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[14] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124 – 143, 1996.

[15] A. Aragon-Zavala, *Antennas and propagation for wireless communication systems*. John Wiley & Sons, 2008.

[16] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proc. of ACM*. IMC, 2009, pp. 280–293.

[17] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *ISCC*, 2012, pp. 59–66.

[18] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of the 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 4–4.

[19] G. Hardin, "The tragedy of the commons," *Science*, vol. 162, no. 3859, pp. 1243–1248, 1968.

[20] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mob. Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb 2013.

[21] E. Hyytiä, T. Spyropoulos, and J. Ott, "Offload (only) the right jobs: Robust offloading using the Markov decision processes," in *Proc. of IEEE WoWMoM*, Jun. 2015, pp. 1–9.

[22] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "MuSIC: Mobility-aware optimal service allocation in mobile cloud computing," in *Proc. of IEEE CLOUD*, Jun. 2013, pp. 75–82.

[23] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *SOSE, 2013 IEEE 7th Int. Symp. on*, Mar. 2013, pp. 494–502.

[24] V. Cardellini, V. De Nitto Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, 2015.

[25] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, to appear.

[26] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy efficient offloading for competing users on a shared communication channel," in *Proc. of IEEE ICC*, Jun. 2015, pp. 3192–3197.

[27] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. of ACM MSWiM*, 2015, pp. 271–278.

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