Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks

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

***Abstract*—Offloading computation to a mobile cloud is a promising solution to augment the computation capabilities of mobile devices. In this paper we consider selfish mobile devices in a dense wireless network, in which individual mobile devices can offload computations through multiple access points or through the base station to a mobile cloud so as to minimize their computation costs. We provide a game theoretical analysis of the problem, prove the existence of pure strategy Nash equilibria, and provide an efficient decentralized 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. Furthermore, we provide an upper bound on the price of anarchy of the game, which serves as an upper bound on the approximation ratio of the proposed decentralized algorithms. We use simulations to evaluate the time complexity of computing Nash equilibria and to provide insights into the price of anarchy of the game under realistic 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.**

***Index terms*—** computation offloading, mobile edge computing, Nash equilibria, decentralized algorithms

I. Introduction

Mobile handsets are increasingly used for various computationally intensive applications, including augmented reality, natural language processing, face, gesture and object recognition, and various forms of user profiling for recommendations [1], [2]. Executing such computationally intensive applications on mobile handsets may result in slow response times, and can also be detrimental to battery life, which may limit user acceptance.

Mobile cloud computing has emerged as a promising solution to serve the computational needs of these computationally intensive applications, while potentially relieving the battery of the mobile handsets [3], [4]. In the case of mobile cloud computing the mobile devices offload the computations via a wireless network to a cloud infrastructure, where the computations are performed, and the result is sent back to the mobile handset. While computation offloading to general purpose cloud infrastructures, such as Amazon EC2, may not be able to provide sufficiently low response times for many applications, emerging mobile edge computing (MEC) resources may provide sufficient computational power close to the network edge to meet all application requirements [5].

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

Computation offloading to a mobile edge cloud can significantly increase the computational capability of individual mobile handsets, but the response times may suffer when many handsets attempt to offload computations to the cloud simultaneously, on the one hand due to the competition for possibly constrained edge cloud resources, on the other hand due to contention in the wireless access [6], [7]. The problem is even more complex in the case of a dense deployment of access points, e.g., cellular femtocells or WiFi access points, when each mobile user can choose among several access points to connect to. Good system performance in this case requires the coordination of the offloading choices of the indvidual mobile handsets, while respecting their individual performance objectives, both in terms of response time and energy consumption.

In this paper we consider the problem of resource allocation for computation offloading by self-interested mobile users to a mobile cloud. The objective of each mobile user is to minimize its cost, which is a linear combination of its response time and its energy consumption for performing a computational task, by choosing whether or not to offload a task via a wireless network to a mobile cloud. Clearly, the choice of a mobile user affects the cost of other mobile users. If too many mobile users choose offloading, the response times could be affected by the contention between the mobile devices for MEC computing resources and for wireless communication resources. Hence, the fundamental question is whether a self-enforcing resource allocation among mobile users exists, and if it exists, whether it can be computed by mobile users in a decentralized manner.

In order to answer this question, we formulate the computation offloading problem as a non-cooperative game and we make three important contributions. First, based on a game theoretical treatment of the problem, we propose an efficient decentralized 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 decentralized algorithm can be used for computing equilibria when the cloud computing resources scale directly proportional with the number of mobile users. Finally, we provide a bound on the price of anarchy for both models of cloud resources. We provide numerical results based on extensive simulations to illustrate the computational efficiency of the proposed algorithms and to evaluate the price of anarchy 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, respectively. We provide a bound on the price of anarchy 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 (MUs). Each MU has a computationally intensive task to perform, and can decide whether to perform the task locally or to offload the computation to a cloud server. The computational task is characterized by the size D_i of the input data (e.g., in bytes), and by the number L_i of the instructions required to perform the computation. Recent work has shown that a task's work requirement X_i per data bit can be approximated by a Gamma distribution [8], [9]. We can thus model $L_i = D_i X_i$, where X_i is a random variable with mean $\overline{X}_i$. Furthermore, we can express the expected number of instructions required for performing MU i's task as $\overline{L}_i = D_i \overline{X}_i$.

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 $\mathcal{A}=\{1,2,...,A\}$ or through a base station (BS) denoted by B.

A. Decentralized mobile cloud computing architecture

Motivated by the emergence of mobile edge clouds, we consider that the cloud, besides providing computational resources, acts as a centralized entity that stores information about the mobile cloud computing system, e.g., achievable data rates and the number of MUs that offload. Furthermore, we consider that the cloud sends this information to the MUs so that the offloading decisions can be made in a decentralized manner. The motivation for such a decentralized implementation is twofold. First, the cloud can be relieved from complex centralized management. Second, the MUs may be autonomous entities with individual interests, and using a decentralized algorithm they would not need to reveal all their parameters to the cloud, but they only need to report their offloading decisions, which helps in protecting privacy and confidentiality.

Despite using a decentralized algorithm, devices still have to send the data pertaining to their tasks through a shared communication link, and thus to avoid eavesdropping and integrity attacks, cryptographic protection is necessary. Protocols for securing computation offloading have been proposed in [10], [11], and are out of scope for our work.

To enable a meaningful analysis of the resource allocation problem, we make the common assumption that the set of MUs does not change during the computation offloading period, i.e., in the order of seconds [12], [4], [13], [14], [15].

B. Communication model

If an MU i 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 through one of the APs or through the BS. Thus, together with local computing MU i can choose an action from the set $\mathfrak{D}_i=\{0,1,2,...,A,B\}$, where 0 corresponds to local computing, i.e., no offloading. 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 $n_o(\mathbf{d})$ the number of MUs that use $o \in \mathcal{A}\cup\{B\}$ for computation offloading, and by $n(\mathbf{d})=\sum_{o\in\mathcal{A}\cup\{B\}} n_o(\mathbf{d})$ the number of MUs that offload. Similarly, we denote by $O_o(\mathbf{d}) = \{i|d_i = o\}$ the set of MUs that offload through $o \in \mathcal{A}\cup\{B\}$, and we define the set of offloaders as $O(\mathbf{d}) = \cup_{o\in\mathcal{A}\cup\{B\}} O_o(\mathbf{d})$.

We denote by $R_{i,o}$ the PHY rate of MU i on $o \in \mathcal{A}\cup\{B\}$, which depends on the physical layer signal characteristics and the corresponding channel gain. We denote by $\omega_i^o(\mathbf{d})$ the uplink rate that MU i receives when she offloads through $o \in \mathcal{A}\cup\{B\}$. For the case of offloading through an AP a we consider that $\omega_i^a(\mathbf{d})$ depends on the PHY rate $R_{i,a}$ and on the number $n_a(\mathbf{d})$ of MUs that offload via AP a

$$\omega_i^a(\mathbf{d}) = R_{i,a}/n_a(\mathbf{d}). \tag{1}$$

This model of the uplink rate can be used to model the bandwidth sharing in TDMA and OFDMA based MAC protocols [16].

For the case of offloading through the BS we consider that the uplink data rate $\omega_i^B(\mathbf{d})$ of MU i is independent of the number of MUs that offload through the BS, i.e., $\omega_i^B(\mathbf{d}) = R_{i,B}$.

The uplink rate $\omega_i^o(\mathbf{d})$ together with the input data size D_i determines the transmission time $T_{i,o}^c(\mathbf{d})$ of MU i for offloading through $o \in \mathcal{A}\cup\{B\}$,

$$T_{i,o}^c(\mathbf{d}) = D_i/\omega_i^o(\mathbf{d}). \tag{2}$$

To model the energy consumption of the MUs, we consider that every MU i knows the transmit power $P_{i,o}$ that it would use to transmit the data through $o \in \mathcal{A}\cup\{B\}$, e.g., determined using an algorithm as the ones proposed in [17], [18]. Thus, the energy consumption of MU i for offloading the input data of size D_i through $o \in \mathcal{A}\cup\{B\}$ is

$$E_{i,o}^c(\mathbf{d}) = P_{i,o} T_{i,o}^c(\mathbf{d}). \tag{3}$$

C. Computation model

In what follows we introduce our model of the time and energy consumption of performing the computation locally and in the cloud server.

1) Local computing: In the case of local computing data need not be transmitted, but the task has to be processed using local computing power. We consider that the expected time it takes to complete MU i's task locally consists of two parts [19]. The first part is the expected CPU execution time $\overline{T}_i^{0,exe}=\overline{L}_i \cdot CPI_i \cdot CC_i$, where CPI_i and CC_i are the average number of cycles per instruction and the clock cycle time, respectively. The second part is the expected time $\overline{T}_i^{0,ot}$ the processor spends executing

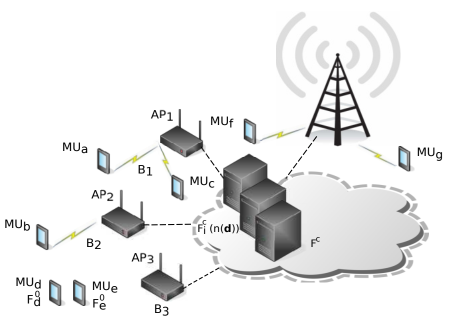

Fig. 1. An example of a mobile cloud computing system

other tasks, including disk and memory management, I/O and operating system activities. Finally, we express the expected time it takes to complete MU i's task locally as

$$\overline{T}_i^0 = \overline{T}_i^{0,exe} + \overline{T}_i^{0,ot}, \tag{4}$$

where $\overline{T}_i^{0,ot}$ may or may not depend on $\overline{L}_i$.

This model essentially implies that the expected execution time of a task is an affine function of $\overline{L}_i$. In practice the MUs can maintain a history of past execution times and can use this history for estimating the parameters of the affine function, which allows to predict the mean execution time as a function of $\overline{L}_i$.

In order to model the expected energy consumption of local computing we denote by v_i the consumed energy per CPU cycle, and thus we obtain

$$\overline{E}_i^0 = v_i \overline{L}_i CPI_i. \tag{5}$$

2) Cloud computing: In the case of cloud computing, after the data are transmitted through one of the APs or through the BS, processing is done at the cloud server. In order to express the expected time it takes to complete MU i's task in the cloud server, we use a similar model as in the case of local computing, but we consider that the expected time $\overline{T}_i^{c,ot}(\mathbf{d})$ the cloud server spends executing other tasks, besides from the system-related tasks, may also depend on the tasks of the other MUs that offload. We make the reasonable assumption that $\overline{T}_i^{c,ot}(\mathbf{d})$ is a non-decreasing function of the number $n(\mathbf{d})$ of MUs that offload, and thus the computation capability assigned to MU i by the cloud is a non-increasing function of $n(\mathbf{d})$. Consequently, we can express the expected time it takes to complete MU i's task in the cloud server as

$$\overline{T}_i^c(\mathbf{d}) = \overline{T}_i^{c,exe} + \overline{T}_i^{c,ot}(\mathbf{d}). \tag{6}$$

where $\overline{T}_i^{c,exe} = \overline{L}_i \cdot CPI_c \cdot CC_c$ and $\overline{T}_i^{c,ot}(\mathbf{d})$ may or may not depend on $\overline{L}_i$.

Figure 1 shows an example of a mobile cloud computing system in which 3 of 7 MUs offload their task through one of 3 APs, 2 MUs offload their task through the BS, and 2 MUs perform local computation.

D. Cost Model

In order to express the expected cost of MU i we denote by γ_i^E the weight attributed to energy consumption and by γ_i^T the weight attributed to the time it takes to finish the computation, $0 \le \gamma_i^E, \gamma_i^T \le 1$.

Using this notation, for the case of local computing the expected cost of MU i can be modeled as a linear combination of the expected time it takes to complete the task locally and the corresponding expected energy consumption,

$$C_i^0 = \gamma_i^T \overline{T}_i^0 + \gamma_i^E \overline{E}_i^0 = \gamma_i^T (\overline{T}_i^{0,exe} + \overline{T}_i^{0,ot}) + \gamma_i^E v_i \overline{L}_i CPI_i. \tag{7}$$

For the case of offloading we consider a subscription-based pricing mechanism in which MUs pay a flat fee F in order to use cloud resources [20]. Consequently, in the case of offloading through $o \in \mathcal{A} \cup \{B\}$ the expected cost of MU i can be modeled as a linear combination of the expected time it takes to complete MU i's task in the cloud server, the transmission time, the corresponding transmit energy, and a fee for using cloud resources,

$$\begin{aligned} C_{i,o}^c(\mathbf{d}) &= \gamma_i^T (\overline{T}_i^c(\mathbf{d}) + T_{i,o}^c(\mathbf{d})) + \gamma_i^E E_{i,o}^c(\mathbf{d}) + F \\ &= (\gamma_i^T + \gamma_i^E P_{i,o}) \frac{D_i}{\omega_i^o(\mathbf{d})} + \gamma_i^T \overline{T}_i^c(\mathbf{d}) + F. \end{aligned} \tag{8}$$

Similar to previous works [7], [21], [22], 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.

For notational convenience let us define the indicator function $I(d_i, o)$ for MU i as

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

We can then express the expected cost of MU i in strategy profile $\mathbf{d}$ as

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

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

E. Computation Offloading Game

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

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

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 $C(\mathbf{d})$.

It may seem natural to model the interaction between the MUs as a Bayesian game, since the number of instructions L_i are random variables. However, it follows from (10) that the solution to (11) is determined by the expectation $\overline{L}_i$ and the number of MUs that offload. We can thus model the interaction between the MUs as a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$, in which the players are the MUs that aim at minimizing their expected cost. 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 expected cost by changing her strategy, i.e., a Nash equilibrium of the game Γ.

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

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

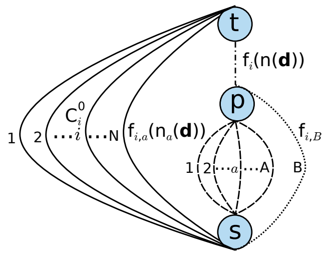

Fig. 2. The computation offloading game modeled as a player-specific congestion game. The source node s represents N MUs and the sink node t corresponds to the execution of a computation task.

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 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 replies a *best improvement path*. Observe that in a NE all MUs play their best replies to each others' strategies.

It is important to note that we do not consider mixed strategy NE. Since the MCOG is a finite game, Nash equilibria in mixed strategies are guaranteed to exist, but they are impractical for two reasons. First, a mixed strategy would make it hard for a cloud scheduler to allocate computational resources. Second, mixed strategy equilibria are computationally hard to find in general. Hence, our focus is on finding pure strategy NE.

F. The MCOG as a player-specific congestion game

In the case of offloading through an AP a the cost $C^c_{i,a}(\mathbf{d})$ depends on the number of MUs that offload through the same AP a and on the total number of MUs that offload, while in the case of offloading through BS the cost $C^c_{i,B}(\mathbf{d})$ only depends on the total number of MUs that offload. Hence, the MCOG can be modeled as a *player-specific congestion* game as illustrated in Figure 2. The source node s represents N MUs, the sink node t corresponds to the execution of the computation task, and the intermediate node p corresponds to offloading. A solid edge i corresponds to local computing by MU i and has a cost of C^0_i, a dashed edge a corresponds to using AP a, a dotted edge B corresponds to using the BS, and the dash-dotted edge that connects node p to the sink node t corresponds to cloud computing. For a strategy profile $\mathbf{d}$ the cost $f_{i,a}(n_a(\mathbf{d}))$ of the dashed edge that corresponds to AP a is a function of the number of MUs that offload via AP a, the cost $f_{i,B}$ of the dotted edge that corresponds to the BS is independent of the other MUs' strategies, and the cost $f_i(n(\mathbf{d}))$ of the dash-dotted edge is a function of the total number of MUs that offload.

Unfortunately, general pure equilibrium existence results are not known for *player-specific congestion* games. Therefore, a natural question is whether the MCOG possesses a pure NE, and if it possesses, whether there is a low complexity decentralized algorithm for computing it. In what follows we answer these questions.

III. Equilibria and the JPBR Algorithm

We start the analysis with the definition of the set of congested communication links and of the notion of the

```
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:     $\mathbf{d} = (d'_i, d_{-i})$
3: end while
4: return d
```

Fig. 3. Pseudo code of the *ImprovementPath* algorithm.

reluctance to offload.

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

$$D_{O\to O}(\mathbf{d}) = \{o \in \mathcal{A}\cup\{B\} | \exists i \in O_o(\mathbf{d}), \exists b \in \mathcal{A}\cup\{B\}\setminus\{o\}, C^c_{i,o}(o, d_{-i}) > C^c_{i,b}(b, d_{-i})\}.$$

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

$$D_{O\to L}(\mathbf{d}) = \{o \in \mathcal{A}\cup\{B\} | \exists i \in O_o(\mathbf{d}), C^c_{i,o}(o, d_{-i}) > C^0_i\}$$

Definition 3. The reluctance to offload via $o \in \mathcal{A}\cup\{B\}$ of MU i in a strategy profile $\mathbf{d}$ is $\rho_i(\mathbf{d}) = C^c_{i,o}(\mathbf{d})/C^0_i$.

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

A. The ImprovementPath Algorithm

Using these definitions, let us start with investigating whether the simple *ImprovementPath* algorithm shown in Figure 3 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, even in the case when the transmit power $P_{i,o}$ of every MU i is the same for all APs.

Example 1. *Consider a MCOG with $\mathcal{N} = \{a, b, c, d, e\}$, $\mathcal{A} = \{1, 2, 3\}$ and the BS. Furthermore, let the expected time it takes to complete MU i's task in the cloud server be linear in $n(\boldsymbol{d})$, $\overline{L}_i$ and CC_c, i.e., $\overline{T}^c_i(\boldsymbol{d}) = n(\boldsymbol{d})\overline{L}_i CC_c$, and assume that the transmit power $P_{i,o}$ of every MU i is the same for all APs, i.e., $P_{i,o} = P_i$. Figure 4 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.*

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 $\frac{1}{R_{b,3}}(\gamma^T_b + \gamma^E_b P_b)D_b + 5\gamma^T_b\overline{L}_b CC_c > \frac{1}{R_{b,3}}(\gamma^T_b + \gamma^E_b P_b)D_b + 3\gamma^T_b\overline{L}_b CC_c$ holds, from inequalities

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

$$R_{c,2}>R_{c,1} \quad (1)$$
$$\frac{2}{R_{b,2}}(\gamma_b^T+\gamma_b^E P_b)D_b+3\gamma_b^T\overline{L}_b CC_c>C_b^0 \quad (2)$$
$$C_d^0>\frac{2}{R_{d,2}}(\gamma_d^T+\gamma_d^E P_d)D_d+3\gamma_d^T\overline{L}_d CC_c \quad (3)$$
$$C_e^0>\frac{3}{R_{e,2}}(\gamma_e^T+\gamma_e^E P_e)D_e+4\gamma_e^T\overline{L}_e CC_c \quad (4)$$
$$R_{c,1}>\frac{2}{3}R_{c,2} \quad (5)$$
$$C_b^0>\frac{1}{R_{b,3}}(\gamma_b^T+\gamma_b^E P_b)D_b+5\gamma_b^T\overline{L}_b CC_c \quad (6)$$
$$\frac{2}{R_{e,2}}(\gamma_e^T+\gamma_e^E P_e)D_e+5\gamma_e^T\overline{L}_e CC_c>C_e^0 \quad (7)$$
$$\frac{1}{R_{d,2}}(\gamma_d^T+\gamma_d^E P_d)D_d+4\gamma_d^T\overline{L}_d CC_c>C_d^0 \quad (8)$$
$$R_{b,2}>R_{b,3} \quad (9)$$

Fig. 4. A cyclic improvement path in a computation offloading game with 5 MUs, 3 APs and 1 BS. 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.

(2) and (6) it follows that $R_{b,3} > \frac{1}{2}R_{b,2}$. Combining inequalities (3) and (8) we have that $\gamma_d^T\overline{L}_d CC_c > \frac{1}{R_{d,2}}(\gamma_d^T + \gamma_d^E P_d)D_d$. Similarly, it follows from inequalities (4) and (7) that $\gamma_e^T\overline{L}_e CC_c > \frac{1}{R_{e,2}}(\gamma_e^T + \gamma_e^E P_e)D_e$. 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 $i, i' \in \mathcal{N}$, $i \neq i'$.

Example 1 illustrates that the improvement paths may be cyclic, and thus the MCOG does not have the finite improvement property, which implies that the MCOG does not allow a generalized ordinal potential function [23]. Consequently, the *ImprovementPath* algorithm in which the MUs perform improvement steps iteratively cannot be used for computing a NE.

B. The ImproveOffloading Algorithm

Although improvement paths may cycle, as we next show, improvement paths are finite if we only allow the MUs to change between APs or to change between APs and the BS but not to start or to stop offloading. We refer to this algorithm as the *ImproveOffloading* algorithm, and show its pseudo code in Figure 5. Our first result shows that all improvement paths generated by the *ImproveOffloading* algorithm are finite.

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

Proof. Let us define the function

$$\Phi(\mathbf{d}) = \sum_{a'=1}^{A}\sum_{n=1}^{n_{a'}(\mathbf{d})} log(n) - \sum_{o\in\mathcal{A}\cup\{B\}}\sum_{i'=1}^{N} log(S_{i',o})I(d_{i'}, o),$$

where $S_{i,o} \triangleq \frac{R_{i,o}}{D_i(\gamma_i^T+\gamma_i^E P_{i,o})}$.

We prove the lemma by showing that the function $\Phi(\mathbf{d})$ decreases strictly at every improvement step generated by the *ImproveOffloading* algorithm.

First, 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 $n_a(a, d_{-i})/n_b(b, d_{-i}) < S_{i,a}/S_{i,b}$. Since $n_a(a, d_{-i}), n_b(b, d_{-i}) > 0$ and $S_{i,a}$,$S_{i,b}>0$ this is equivalent to

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

```
                 d = ImproveOffloading(d)
1: while $D_{O\to O}(\mathbf{d}) \neq \emptyset$ do
2:     $(i', a') \leftarrow \arg\max_{\{i\in O(\mathbf{d}),\exists o\in\mathcal{A}\cup\{B\},C_i(o,d_{-i})<C_i(\mathbf{d})\}} \frac{C_i(\mathbf{d})}{C_i(o, d_{-i})}$
3:     $\mathbf{d} = (a', d_{-i'})$
4: end while
5: return d
```

Fig. 5. Pseudo code of the *ImproveOffloading* algorithm.

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

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

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

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

It follows from (13) that $\Phi(a, d_{-i}) - \Phi(b, d_{-i}) < 0$.

Second, let us consider an improvement step made by MU i in which she changes from offloading via AP a to offloading via BS B. 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 $C_i(B, d_{-i}){<}C_i(a, d_{-i})$ implies $n_a(a, d_{-i}) > S_{i,a}/S_{i,B}$. Since $n_a(a, d_{-i}), S_{i,a}, S_{i,B}{>}0$ this is equivalent to

$$\log(n_a(a, d_{-i})) > \log(S_{i,a}) - \log(S_{i,B}). \quad (13)$$

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

$$\Phi(B, d_{-i}) - \Phi(a, d_{-i}) = -\log(n_a(a, d_{-i}))+ + \log(S_{i,a}) - \log(S_{i,B}).$$

It follows from (13) that $\Phi(B, d_{-i}) - \Phi(a, d_{-i}) < 0$. Similarly, we can show that $C_i(a, d_{-i}) < C_i(B, d_{-i})$ implies $\Phi(a, d_{-i}) < \Phi(B, d_{-i})$.

Since the number of strategy profiles is finite, $\Phi(\mathbf{d})$ can not decrease infinitely and the *ImproveOffloading* algorithm terminates after a finite number of improvement steps. □

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

C. The JPBR Algorithm

In what follows we use the *ImproveOffloading* algorithm as a building block for proving that a NE always exists in the MCOG even if it does not allow a generalized ordinal potential function.

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

Update phase of JPBR algorithm

```
1:  /* Corresponds to case (i) */
2:  Let $\mathbf{d}'(t) = \text{ImproveOffloading}(\mathbf{d}(t))$
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: /*Link with MU with highest reluctance to offload */
11:     Let $i' \leftarrow (b, 1)$
12:     Let $\mathbf{d}'(t) = (0, d'_{-i'}(t))$/*Best reply by MU $(b, 1)$*/
13:     if $\exists i \in \mathcal{N}\setminus O(\mathbf{d}'(t))$ s.t. $C_i^0 > C_i(b, d'_{-i}(t))$ then
14:       $i' \leftarrow \arg\min_{\{i\in\mathcal{N}\setminus O(\mathbf{d}'(t)) | C_i^0 > C_i(b,d'_{-i}(t))\}} \rho_i(b, d'_{-i}(t))$
15:                 /*MU with lowest reluctance to offload*/
16:       Let $\mathbf{d}'(t) = (b, d'_{-i'}(t))$/*Best reply by MU $i'$*/
17:     else
18:       Let $\mathbf{d}'(t) = \text{ImproveOffloading}(\mathbf{d}'(t))$
19:     end if
20:   end while
21: end if
```

Fig. 6. Pseudo code of the update phase of the JPBR algorithm.

of MUs that offload will not decrease as more MUs are added. Similarly, if $d_i^*(1) = B$, MU i would never have an incentive to offload using one of the APs, because the number of MUs that offload using any of the APs will not decrease as more MUs are added. Otherwise, if MU i decides to offload using one of the APs, she would play her best reply which is given by $d_i^*(1) = \arg\max_{a\in\mathcal{A}} S_{i,a}$. 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 6. 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 $d^*_{N^{(t)}}(t) = o \in \mathcal{A} \cup \{B\}$, 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 $i \in O_o(\mathbf{d}(t))$ offloading using $b \in \mathcal{A} \cup \{B\} \setminus \{o\}$ becomes a best reply, (ii) for some MUs $i \in O(\mathbf{d}(t))$ local computing becomes a best reply. Note that case (i) can happen only if $o \neq B$, as otherwise MU i would be able to gain by changing her strategy to offloading using on of the APs in $\mathbf{d}^*(t-1)$.

Let us first consider case (i) and let MUs execute the *ImproveOffloading* 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 or between APs and the BS. In the resulting strategy profile the number of MUs that offload will be $n(\mathbf{d}^*(t-1)) + 1$. Furthermore, there will be one communication link (denoted by a') for which the number of offloaders is $n_{a'}(\mathbf{d}^*(t-1)) + 1$, while for the other communication links $a \neq a'$ 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 communication link a' is the same communication link o that was chosen by MU $N^{(t)}$ when she 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 offloading through a' to local computation, $n_o(\mathbf{d}'(t)) = n_o(\mathbf{d}^*(t-1))$ holds for every $o \in \mathcal{A} \cup \{B\}$ 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 an MU i that wants to change her strategy from offloading through $b \in \mathcal{A} \cup \{B\} \setminus \{a'\}$ 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 we have that $n(\mathbf{d}'(t)) = n(\mathbf{d}^*(t-1))$. Observe that if $b = B$ there can be no MU that wants to start offloading because $n_a(\mathbf{d}'(t)) = n_a(\mathbf{d}^*(t-1))$ for every AP $a \in \mathcal{A}$, and there is no more MU that would like to stop offloading either because $n(\mathbf{d}'(t)) = n(\mathbf{d}^*(t-1))$, and $\mathbf{d}'(t)$ is a NE. Otherwise, if $b \neq B$ we have that $n_b(\mathbf{d}'(t)) = n_b(\mathbf{d}^*(t-1)) - 1$ and 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, there is no more MU that would like to stop offloading either because $n(\mathbf{d}'(t)) = n(\mathbf{d}^*(t-1))$ and we reached a NE. 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 an MU that did not offload before, then we reached a NE. Otherwise some MUs that offload via $o \in \mathcal{A} \cup \{B\} \setminus \{b\}$ may want to connect to AP b, and we let them execute the *ImproveOffloading* 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 2. *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 (i) 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. 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. Hence, a NE is reached after at most $(N^{(t)}{-}2){+}(N^{(t)}{-}2){+}(N^{(t)}{-}3){+}(N^{(t)}{-}2)(A{-}1)$ updates. □

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 3. *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 or using the BS, we can consider MU i as an 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 an 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. □

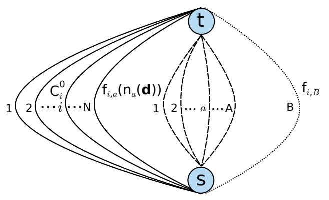

Fig. 7. The computation offloading game in the case of an elastic cloud modeled as a player-specific singleton congestion game. The source node s represents N MUs, and the sink node t corresponds to the execution of a computation task.

Observe that Theorem 1 and Theorem 3 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 the MCOG with polynomial complexity. In what follows we show that a much simpler algorithm can be used for computing an equilibrium if we assume that the expected time needed for the cloud to complete MU i's task is independent of the other MUs' strategies, and thus of the number of MUs that offload. This is a reasonable assumption for large cloud computing infrastructures, in which the cloud computing resources scale with the number of MUs. We refer to the resulting model as the *elastic* cloud model, and we express the expected time needed for the cloud to complete MU i's task as $\overline{T}_i^c = \overline{T}_i^{c,exe} + \overline{T}_i^{c,ot}$. Consequently, the cost function of MU i in the case of offloading is simpler because the part of MU i's cost concerning the time needed for performing its task in the cloud does not depend on the strategy profile $\mathbf{d}$. Therefore, in the case of the *elastic* cloud model the cost function of MU i when it offloads its task through AP a can be expressed as

$$C^c_{i,a}(\mathbf{d}) = (\gamma_i^T + \gamma_i^E P_{i,a}) D_i \frac{n_a(\mathbf{d})}{R_{i,a}} + \gamma_i^T \overline{T}_i^c + F, \quad (14)$$

which only depends on the number of MUs that offload through the same AP a. Furthermore, in the case of offloading through BS B the cost function is independent of the other MUs' strategies and can be expressed as

$$C^c_{i,B} = (\gamma_i^T + \gamma_i^E P_{i,B}) \frac{D_i}{R_{i,B}} + \gamma_i^T \overline{T}_i^c + F, \qquad (15)$$

Let us recall that when the expected time needed for the cloud to complete MU i's task depends on the other MUs' strategies, the MUs have to share both communication and computing resources (c.f., Figure 2). On the contrary, in the case of the *elastic* cloud model, the MUs only have to share communication resources when they offload their tasks through one of the APs, and thus the MCOG can be modeled as a *player-specific singleton congestion* game as illustrated in Figure 7. Compared to the graph shown in Figure 2, the graph in Figure 7 has only the source node s and the sink node t. A solid edge i corresponds to local computing by MU i and has a cost of C_i^0, a dashed edge a corresponds to using AP a, and the dotted edge B corresponds to using the BS. Every MU can either choose the solid edge or the dotted edge that corresponds to itself or one of the dashed edges, thus one of $A+2$ edges. For a strategy profile $\mathbf{d}$ the cost $f_{i,a}(n_a(\mathbf{d}))$ of the dashed edge that corresponds to AP a is a function of the number of MUs that offload through AP a, while the cost $f_{i,B}$ of the dotted edge that corresponds to the BS is independent of the other MUs' strategies.

For *player-specific singleton congestion* games it is known that a pure strategy Nash equilibrium always exsists, even if potential function may not exist [24]. Before we formulate the theorem, let us recall the definition of a generalized ordinal potential from [23].

Definition 4. A function $\Phi : \times\mathfrak{D}_i \rightarrow \mathbb{R}$ is a generalized ordinal potential function for the strategic game $\Gamma^s =< \mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i >$ 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 4. *The MCOG with elastic cloud admits the generalized ordinal potential function*

$$\Phi(\boldsymbol{d}) = \sum_{a'=1}^{A} \sum_{n=1}^{n_{a'}(\boldsymbol{d})} log(n) - \sum_{o\in\mathcal{A}\cup\{B\}} \sum_{i'=1}^{N} log(M_{i'} S_{i',o}) I(d_{i'}, o), \quad (16)$$

and hence it possesses a pure strategy Nash equilibrium, if $F < \gamma_i^E v_i \overline{L}_i CPI_i + \gamma_i^T(\overline{T}_i^0 - \overline{T}_i^c)$.

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 (14), the condition $C_i(a, d_{-i}) < C_i(0, d_{-i})$ implies that

$$(\gamma_i^T + \gamma_i^E P_{i,a}) D_i \frac{n_a(a, d_{-i})}{R_{i,a}} + \gamma_i^T \overline{T}_i^c + F \quad (17) \\ < \gamma_i^T \overline{T}_i^0 + \gamma_i^E v_i \overline{L}_i CPI_i.$$

After algebraic manipulations we obtain

$$n_a(a, d_{-i}) < M_i S_{i,a}, \quad (18)$$

where $S_{i,a} \triangleq \frac{R_{i,a}}{D_i(\gamma_i^T + \gamma_i^E P_{i,a})}$ and $M_i \triangleq \gamma_i^T(\overline{T}_i^0 - \overline{T}_i^c) + \gamma_i^E v_i \overline{L}_i CPI_i - F$.

Since $n_a(a, d_{-i}) > 0$ and $M_i S_{i,a} > 0$, (17) implies that

$$\log(n_a(a, d_{-i})) < \log(M_i S_{i,a}). \quad (19)$$

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

$$\Phi(a, d_{-i}) = \sum_{n=1}^{n_a(a,d_{-i})} \log(n) + \sum_{a'\neq a} \sum_{n=1}^{n_{a'}(a,d_{-i})} \log(n) \\ - \log(M_i S_{i,a}) - \sum_{o\in\mathcal{A}\cup\{B\}} \sum_{i'\neq i} \log(M_{i'} S_{i',o}) I(d_{i'}, o),$$

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

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

Since $n_a(a, d_{-i}) = n_a(0, d_{-i}) + 1$, we obtain $\Phi(a, d_{-i}) - \Phi(0, d_{-i}) = \log(n_a(a, d_{-i})) - \log(M_i S_{i,a})$. It follows from (19) 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 (14), the condition $C_i(a, d_i) < C_i(b, d_i)$ implies that

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

which is equivalent to

$$\frac{n_a(a, d_{-i})}{n_b(b, d_{-i})} < \frac{S_{i,a}}{S_{i,b}}. \quad (20)$$

Since $n_a(a, d_{-i}), n_b(b, d_{-i}) > 0$ and $S_{i,a}, S_{i,b} > 0$, (20) implies

$$\log(n_a(a, d_{-i})) - \log(n_b(b, d_{-i})) < \log(S_{i,a}) - \log(S_{i,b}). \quad (21)$$

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

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

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 $\Phi(a, d_{-i}) - \Phi(b, d_{-i}) = \log(n_a(a, d_{-i})) - \log(n_b(b, d_{-i})) - (\log(S_{i,a}) - \log(S_{i,b}))$. It follows from (21) that $\Phi(a, d_{-i}) - \Phi(b, d_{-i}) < 0$.

Third, we show that $C_i(B, d_i) < C_i(0, d_i)$ implies $\Phi(B, d_i) < \Phi(0, d_i)$. According to (7), (10) and (15), the condition $C_i(B, d_i) < C_i(0, d_i)$ implies that

$$(\gamma_i^T + \gamma_i^E P_{i,B}) \frac{D_i}{R_{i,B}} + \gamma_i^T \overline{T}_i^c + F < \gamma_i^T \overline{T}_i^0 + \gamma_i^E v_i \overline{L}_i CPI_i, \quad (22)$$

which is equivalent to

$$1 < M_i S_{i,B}. \quad (23)$$

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

$$\Phi(B, d_{-i}) = \sum_{a'=1}^{A} \sum_{n=1}^{n_{a'}(B,d_{-i})} \log(n) - \log(M_i S_{i,B}) \\ - \sum_{o\in\mathcal{A}\cup\{B\}} \sum_{i'\neq i} \log(M_{i'} S_{i',o}) I(d_{i'}, o).$$

Since $n_a(B, d_{-i}) = n_a(0, d_{-i})$ for every AP a, we have that $\Phi(B, d_{-i}) - \Phi(0, d_{-i}) = -\log(M_i S_{i,B})$, and since $M_i S_{i,B} > 0$, it follows from (23) that $\Phi(B, d_{-i}) - \Phi(0, d_{-i}) < 0$. Similarly, we can show that $C_i(0, d_{-i}) < C_i(B, d_{-i})$ implies $\Phi(0, d_{-i}) < \Phi(B, d_{-i})$.

Finally, we show that $C_i(B, d_i) < C_i(a, d_i)$ implies $\Phi(B, d_i) < \Phi(a, d_i)$. According to (10), (14) and (15), the condition $C_i(B,d_i) < C_i(a,d_i)$ implies that

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

which is equivalent to

$$n_a(a,d_{-i}) > \frac{S_{i,a}}{S_{i,B}}. \tag{24}$$

Since $n_a(a,d_{-i}) > 0$ and $S_{i,a}, S_{i,B} > 0$, (24) implies

$$\log(n_a(a,d_{-i})) > \log(S_{i,a}) - \log(S_{i,B}). \tag{25}$$

Since $n_a(a,d_{-i}) = n_a(B,d_{-i}) + 1$, we have that $\Phi(B,d_{-i}) - \Phi(a,d_{-i}) = -\log(n_a(a,d_{-i})) + \log(S_{i,a}) - \log(S_{i,B})$. It follows from (25) that $\Phi(B,d_{-i}) - \Phi(a,d_{-i}) < 0$. Similarly, we can show that $C_i(a,d_{-i}) < C_i(B,d_{-i})$ implies $\Phi(a,d_{-i}) < \Phi(B,d_{-i})$, 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 [25]. Therefore, the existence of a generalized ordinal potential function allows us to use the *ImprovementPath* Algorithm (c.f., Figure 3) 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.*

In what follows, we prove that if MUs aim at minimizing only their energy consumption, the *ImprovementPath* algorithm can be used for computing a NE not only for the MCOG with elastic cloud, but also in the general case, i.e., if the time needed for performing the task of an MU in the cloud depends on the strategy profile $\mathbf{d}$.

Proposition 5. *The* ImprovementPath *algorithm terminates in a NE after a finite number of improvement steps for the MCOG with $\gamma_i^T = 0$, i.e., if each MU aims at minimizing its energy consumption.*

Proof. Observe that if $\gamma_i^T = 0$, MU i spends the energy only to transmit the data through one of the APs in the case of offloading, and thus the cost function of MU i only depends on the number of MUs that offload through the same AP, and it is independent of the total number of MUs that offload. Hence, the MCOG in which $\gamma_i^T=0$ for all MUs can be modeled by a congestion game on parallel links as shown in Figure 7. Furthermore, (16) is a generalized ordinal potential function of the game with $\gamma_i^T=0$, which proves the proposition. □

V. Price of Anarchy

We have so far shown that a NE exists and provided low complexity algortihms for computing it. 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

$$PoA = \frac{\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d}\in\mathfrak{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})}. \tag{26}$$

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

Theorem 6. *The price of anarchy for the computation offloading game is upper bounded by*

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

both in the case of elastic cloud and in the case of non-elastic cloud.

Proof. First we show that if there is a NE in which all players 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 $\sum_{i\in\mathcal{N}} C_i(d_i^*, d_{-i}^*) \leq \sum_{i\in\mathcal{N}} C_i^0$ holds, and if all players 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 $d_i = o$ for some $o \in \mathcal{A} \cup \{B\}$, 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, $\omega_i^o(d_i, d_{-i}) \leq R_{i,o}$ and $\overline{T}_i^c(n(d_i, d_{-i})) \leq \overline{T}_i^c$, which implies

$$\begin{aligned}(\gamma_i^T + \gamma_i^E P_{i,o})&\frac{D_i}{\omega_i^o(d_i, d_{-i})} + \gamma_i^T \overline{T}_i^c(n(d_i, d_{-i})) \\ &\geq (\gamma_i^T + \gamma_i^E P_{i,o})\frac{D_i}{R_{i,o}} + \gamma_i^T \overline{T}_i^c = C_{i,o}^{\bar{c}}.\end{aligned}$$

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

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

which proves the theorem. □

In the following we provide an upper bound on the PoA in the case of homogeneous MUs, that is, when all MUs have the same parameters.

Proposition 7. *In the case of homogeneous MUs and when the expected time $\overline{T}_i^{c,ot}(\mathbf{d})$ the cloud server spends executing other tasks is an affine function of the number of MUs that offload, the price of anarchy for the MCOG is at most $\frac{5}{2}$.*

Proof. When the MUs are homogeneous and $\overline{T}_i^{c,ot}(\mathbf{d})$ is an affine function of the number of MUs that offload, the MCOG falls into the category of congestion games with affine cost functions. It follows from [26] that congestion games with affine cost functions are smooth games with a PoA at most $\frac{5}{2}$, which proves the proposition. □

Observe that the PoA is in fact a bound on the approximation ratio of the decentralized algorithms used for computing a NE.

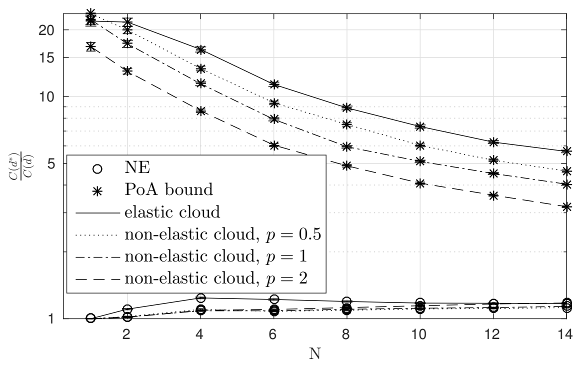

Fig. 8. The *cost ratio* and the upper bound on the PoA for the *elastic* and *non-elastic* cloud ($p = 0.5, 1, 2$), $A = 3$ APs.

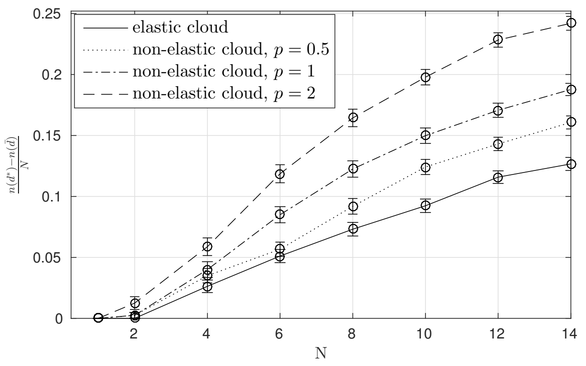

Fig. 9. *Offloading difference ratio* vs. number of MUs N for the *elastic* and *non-elastic* cloud ($p = 0.5, 1, 2$), $A = 3$ APs.

VI. Numerical Results

We use extensive simulations to evaluate the cost performance and the computational time of the *JPBR* algorithm. We placed the MUs and the APs at random on a regular grid with 10^4 points defined over a square area of $1km \times 1km$. We chose the channel gain of MU i to AP a to be 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 [27]. The channel bandwidth B_a of every AP a was set to 5 *MHz*, while the data transmit power $P_{i,a}$ of every MU i and for every AP a was set to to $0.4W$ according to [28]. Given the noise power P_n we calculate the PHY rate $R_{i,a}$ as $R_{i,a} = B_a \log(1 + P_{i,a}d_{i,a}^{-\alpha}/P_n)$.

The clock rate CR_i of MU i was drawn from a continuous uniform distribution with parameters $[0.5, 1]$ *GHz* based on the specification of NVIDIA Tegra 2, which is the reference platform for Android OS [29], while the clock rate CR_c of the cloud was set to 100 *GHz* [30]. Unless otherwise noted, the input data size D_i was uniformly distributed on $[0.42, 2]$ *Mb*. We drew the total number of CPU cycles required to perform the computation ($L_i \cdot CPI$) from a continuous uniform distribution with parameters $[0.1, 0.8]$ *Gcycles*. The consumed energy per CPU cycle v_i was set to $10^{-11}(CR_i)^2$ according to measurements reported in [4], [31]. 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 computed the system cost that can be achieved if all MUs execute their computation tasks locally, which coincides with the bound on the PoA. Unless otherwise noted, the MUs are added at random in the induction steps of the JPBR algorithm. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

A. Optimal vs. Equilibrium Cost

Figure 8 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 expected time it takes to complete MU i's task is linear in the number of MUs that offload, i.e., $\overline{T}_i^{c,exe}(\mathbf{d}) = pn(\mathbf{d})\overline{L}_i CPI_c CC_c$. We refer to this latter case as a *non-elastic* cloud and to the coefficient p as the *cloud provisioning coefficient*. A coefficient of $p = 1$ corresponds to a cloud with fixed amount of resources, $p < 1$ to resources that scale slower than the demand, while $p > 1$ 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* ($p = 0.5, 1$ and 2).

The results in Fig. 8 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 p increases, and thus the problem becomes computationally easier for larger values of p.

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 9 for the same set of parameters as in Figure 8. The results show that the *offloading difference ratio* increases with the number of MUs, which explains the increased *cost ratio* observed in Figure 8, 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 [32]. The results also show that the *offloading difference ratio* is slightly lower in the case of the *elastic* cloud, which is due that a higher proportion of MUs offload in the optimal solution for the *elastic* cloud.

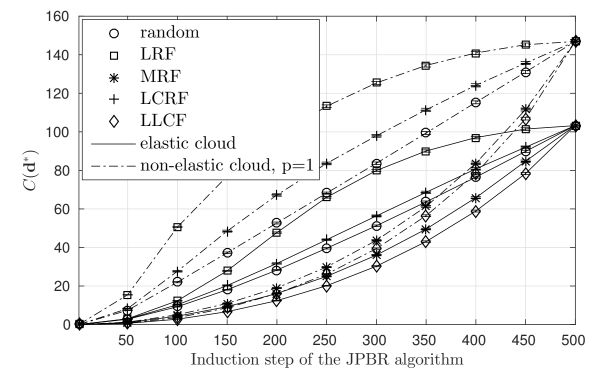

Fig. 10. The system cost vs. the induction step for the *elastic* and *non-elastic* cloud ($p = 1$), $A = 10$ APs, $N = 500$ MUs.

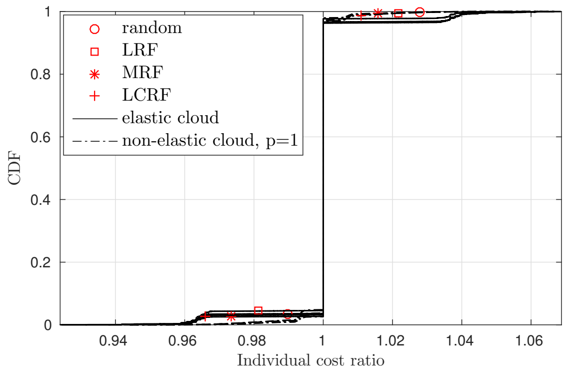

Fig. 11. Distribution of the *individual cost ratio* for the *elastic* and *non-elastic* cloud ($p = 1$), $A = 10$ APs, $N = 500$ MUs.

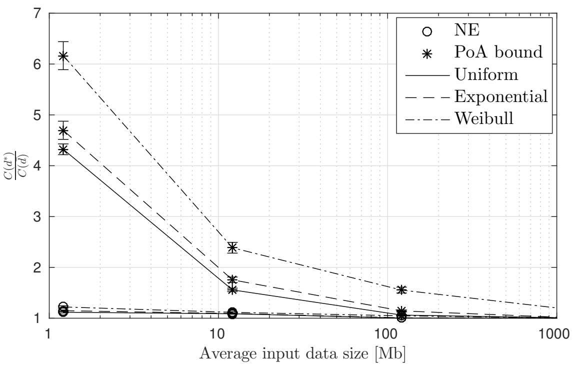

Fig. 12. The *cost ratio* and the upper bound on the PoA for the *elastic* and *non-elastic* cloud ($p = 1$), uniform, exponential and Weibull distributions of the input data sizes, $A = 3$ APs, $N = 12$ MUs.

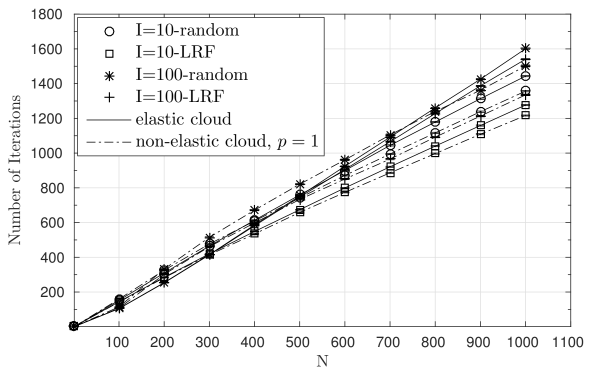

Fig. 13. Number of iterations vs. number of MUs N for the *elastic* and *non-elastic* cloud ($p = 1$), A =10 and 100 APs.

B. Impact of the order of adding MUs

Since the order in which the MUs are added in induction steps of the JPBR algorithm can be arbitrarily chosen, in the following we investigate in which order the controller should add the MUs, so that their costs are minimized. To do so, we consider five orderings of adding MUs: *random* where MUs are added at random, *least reluctance first* (LRF) where MUs are added in increasing order of their ratio $\frac{D_i}{C_i^0 L_i CPI_i}$, *most reluctance first* (MRF) where MUs are added in decreasing order of their ratio $\frac{D_i}{C_i^0 L_i CPI_i}$, *least clock rate first* ($LCRF$) where MUs are added in increasing order of their clock rate CR_i, and *least local cost first* ($LLCF$) where MUs are added in increasing order of their local cost C_i^0. Figure 10 shows the system cost as a function of the number of induction steps performed (i.e., MUs added) by the JPBR algorithm, for the *elastic* and the *non-elastic* cloud (p=1), $N = 500$ MUs, and $A = 10$ APs. The results show that the order of adding MUs affects the system cost during the induction steps, but the system cost $C(\mathbf{d}^*)$ in the equilibrium $\mathbf{d}^*$ computed upon the last induction step is almost the same for all considered orderings, and thus the order of adding MUs does not affect the system performance significantly.

To gain insight in the impact of the order of adding MUs on the cost of the MUs, we define the *individual cost ratio* for a particular order of entry, compared to the LLCF order of entry. Figure 11 shows the CDF of the *individual cost ratio* in the equilibrium $\mathbf{d}^*$ computed upon the last induction step for the same set of parameters as in Figure 10. The results show that the individual costs slightly differ for different orderings only for a few MUs, and thus changing the order of adding MUs does not affect the performance of individual MUs either.

C. 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 $L_i \cdot CPI = 0.45$ *Gcycles*. Figure 12 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 (p=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.

D. 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 ($p = 1$), A=10 and A=100 APs. Figure 13 shows the number of iterations as a function of the number of MUs for the *random* and the LRF order of adding MUs.

Intuitively, one would expect that the LRF order results in a smaller number of iterations, since the MUs with lower $\frac{D_i}{C_i^0 L_i CPI_i}$ ratio have lower computational capability to execute computationally more demanding tasks with smaller offloading data size than the MUs with higher $\frac{D_i}{C_i^0 L_i CPI_i}$. 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

There is a significant body of works that deals with the design of energy efficient computation offloading for a single mobile user [3], [4], [6], [33], [34], [21], [35]. The experimental results in [34] showed that significant battery power savings can be achieved by computation offloading. [6] studied the commmunication overhead of computation offloading and the impact of bandwidth availability on an experimental platform. [3] proposed a code partitioning solution for fine-grained energy-aware computation offloading. [21] proposed an algorithm for offloading partitioned code under bandwidth and delay constraints. [4] proposed CPU frequency and transmission power adaptation for energy-optimal computation offloading under delay constraints. [35] modeled the offloading problem under stochastic task arrivals as a Markov decision process and provided a near-optimal offloading policy.

A number of recent works considered the problem of joint energy minimization for multiple mobile users [13], [36], [14]. [13] studies computation partitioning for streaming data processing with the aim of maximizing throughput, considering sharing of computation instances among multiple mobile users, and proposes a genetic algorithm as a heuristic for solving the resulting optimization problem. [36] models computation offloading to a tiered cloud infrastructure under user mobility in a location-time workflow framework, and proposes a heuristic for minimizing the users' cost. [14] aims at minimizing the mobile users' energy consumption by joint allocation of radio resources and cloud computing power, and provides an iterative algorithm to find a local minimum of the optimization problem.

A few recent works provided a game theoretic treatment of computation offloading in a game theoretical setting [37], [38], [7], [39], [40], [41]. [37] considers a two-stage problem, where first each mobile user decides what share of its task to offload so as to minimize its energy consumption and to meet its delay deadline, and then the cloud allocates computational resources to the offloaded tasks. [38] considers a two-tier cloud infrastructure and stochastic task arrivals and proves the existence of equilibria and provides an algorithm for computing and equilibrium. [40] considers tasks that arrive simultaneously, a single wireless link, and elastic cloud, and show the existence of equilibria when all mobile users have the same delay budget. Our work differs from [37] in that we consider that the allocation of cloud resources is known to the mobile users, from [38] in that we take into account contention in the wireless access, and from [40] in that we consider multiple wireless links and a non-elastic cloud.

Most related to our work are the problems considered in [12], [7], [39], [41]. [12] considers a system where multiple devices can offload their tasks to a non-elastic cloud through one of multiple shared heterogeneous wireless links. Different from [12], we consider that devices besides offloading through shared heterogeneous wireless links can offload their tasks through a non-shared link, and we consider that devices use different transmit powers for different wireless links when they offload their tasks to the cloud. [7] considers contention on a single wireless link and an elastic cloud, assumes upload rates to be determined by the Shannon capacity of an interference channel, and shows that the game is a potential game. [39] extends the model to multiple identical wireless links, shows that the game is still a potential game, and that the same algorithm as in [7] can be used for computing an equilibrium allocation. Unlike these works, we consider heterogeneous wireless links, fair bandwidth sharing and a non-elastic cloud. [41] considers multiple wireless links, fair bandwidth sharing and a non-elastic cloud, and claims the game to have an exact potential. In our work we on the one hand extend the model to an elastic cloud, on the other hand we show that an exact potential cannot exist in case of a non-elastic cloud, but at the same time we prove the existence of an equilibrium allocation, provide an efficient algorithm with quadratic complexity for computing one, and provide a bound on the price of anarchy.

Besides providing efficient distributed algorithms for computing equilibria, the importance of our contribution lies in the fact that while games with an elastic cloud are player-specific singleton congestion games for which the existence of equilibria is known [24], the non-elastic cloud model does not fall in this category of games and thus no general equilibrium existence result exists.

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," *ETSI White Paper*, pp. 1–16, 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*, 2013, pp. 1285–1293.

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

[8] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 1. ACM, 2001, pp. 50–61.

[9] W. Yuan and K. Nahrstedt, "Energy-efficient cpu scheduling for multimedia applications," *ACM TOCS*, pp. 292–331, 2006.

[10] X. Li, Q. Xue, and M. C. Chuah, "Casheirs: Cloud assisted scalable hierarchical encrypted based image retrieval system," in *INFOCOM 2017*. IEEE, 2017, pp. 1–9.

[11] X. Zhang, J. Schiffman, S. Gibbs, A. Kunjithapatham, and S. Jeong, "Securing elastic applications on mobile devices for cloud computing," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 127–134.

[12] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of IEEE INFOCOM*, May 2017.

[13] 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.*, pp. 23–32, Apr. 2013.

[14] 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.

[15] 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.

[16] 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.

[17] M. Xiao, N. B. Shroff, and E. K. Chong, "A utility-based power-control scheme in wireless cellular systems," *IEEE/ACM Transactions on networking*, vol. 11, no. 2, pp. 210–221, 2003.

[18] C. U. Saraydar, N. B. Mandayam, and D. J. Goodman, "Efficient power control via pricing in wireless data networks," *IEEE transactions on Communications*, vol. 50, no. 2, pp. 291–303, 2002.

[19] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 4th ed. Morgan Kaufmann Publishers Inc., 2011.

[20] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stößer, "Cloud computing–a classification, business models, and research directions," *Business & Information Systems Engineering*, vol. 1, no. 5, pp. 391–399, 2009.

[21] 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.

[22] 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.

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

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

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

[26] T. Roughgarden, "Intrinsic robustness of the price of anarchy," *Journal of the ACM (JACM)*, vol. 62, no. 5, p. 32, 2015.

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

[28] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proc. of the 9th ACM SIGCOMM conference*, 2009, pp. 280–293.

[29] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.

[30] 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.

[31] 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.

[32] G. Hardin, "The tragedy of the commons," *Science*.

[33] 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.

[34] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *ACM Mob. Comput. Commun. Rev.*, pp. 19–26, Jan 1998.

[35] 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.

[36] 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.

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

[38] 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.

[39] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM ToN*, pp. 2795–2808, 2016.

[40] 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.

[41] 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.

Slađana Jošilo is a Ph.D. student at the Department of Network and Systems Engineering in KTH, Royal Institute of Technology. She received her M.Sc. degree in electrical engineering from the University of Novi Sad, Serbia in 2012. She worked as a research engineer at the Department of Power, Electronics and Communication Engineering, University of Novi Sad from 2013 to 2014. Her research interests are design and analysis of distributed algorithms for exploiting resources available at the network edge using game theoretical tools.

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