Decentralized Scheduling for Offloading of Periodic Tasks in Mobile Edge Computing

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

***Abstract*—Motivated by various surveillance applications, we consider wireless devices that periodically generate computationally intensive tasks. The devices aim at maximizing their performance by choosing when to perform the computations and whether or not to offload their computations to a cloud resource via one of multiple wireless access points. We propose a game theoretic model of the problem, give insight into the structure of equilibrium allocations and provide an efficient algorithm for computing pure strategy Nash equilibria. Extensive simulation results show that the performance in equilibrium is significantly better than in a system without coordination of the timing of the tasks' execution, and the proposed algorithm has an average computational complexity that is linear in the number of devices.**

I. Introduction

Mobile edge computing (MEC) is considered to become an enabler of a variety of Internet of Things (IoT) applications that are based on a pervasive deployment of wireless sensors. Examples range from water pipeline surveillance [1], through pursuit problems and discrete manufacturing [2] to body area networks [3]. Many of these applications involve the periodic collection of sensory data, which need to be processed timely to enable control decisions. Processing often requires some form of data analytics, e.g., visual analysis, which is computationally demanding.

The key advantage of MEC compared to centralized cloud infrastructures is that computational resources are located close to the network edge [4]. Thus, even though MEC infrastructures may be less resource-rich than centralized clouds, such as Microsoft Azure or AWS, due to their proximity to the sensors they may be able to provide response times that make them suitable for computation offloading for real-time applications.

The proximity of MEC resources makes low response times for individual sensors possible, but when multiple wireless sensors attempt to offload to the MEC simultaneously, the response times might increase due to contention for the communication and the computational resources [5], [6], [7]. Coordination is thus essential for maintaining low response times in the case of MEC computation offloading.

Coordination for offloading periodic tasks involves deciding whether or not to offload the computations, deciding which of the available wireless communication channels to use for offloading, and in the case of periodic tasks, it involves deciding when to collect sensory data and when to offload the computation. In addition coordination should respect that sensors may be managed by different entities, with individual interests. The resulting coordination problem not only has a huge solution space with a combinatorial structure, but it also requires consideration of the potentially diverse requirements of the sensors in terms of response time and energy consumption for performing the computation. Efficient coordination of computation offloading for wireless sensors with periodic tasks is thus a complex problem.

In this paper we address this problem by considering the allocation of cloud and wireless resources among wireless devices that generate tasks periodically. The devices can choose the time slot in which to perform their periodic task, and can decide whether to offload their computation to a cloud through one of many access points or to perform the computation locally. We provide a game theoretical treatment of the problem, and prove the existence of pure strategy Nash equilibria. Our proof provides a characterization of the structure of the equilibria, and serves as an efficient decentralized algorithm for coordinating the offloading decisions of the wireless devices. We use extensive simulations to assess the benefits of coordinated computation offloading compared to uncoordinated computation offloading where devices choose a time slot at random, and in the chosen time slot play an equilibrium allocation. Our results show that the proposed algorithm computes equilibria with good system performance in a variety of scenarios in terms of task periodicity, the number of devices and the number of access points.

The rest of the paper is organized as follows. In Section II we present the system model and the problem formulation. In Section III we present algorithmic and analytical results. In Section IV we show numerical results and in Section V we discuss related work. Section VI concludes the paper.

II. System Model and Problem Formulation

We consider a computation offloading system that consists of N devices, A acces points (APs) and a cloud service. We denote by $\mathcal{N}=\{1,2,...,N\}$ and $\mathcal{A}=\{1,2,...,A\}$ the set of devices and the set of APs, respectively. Each device generates a computationally intensive task periodically every T time units. Device i's task is characterized by the mean size D_i of the input data and by the mean number of CPU cycles L_i required to perform the computation. We make the reasonable assumption that the number X of CPU cycles required per bit can be modeled by a random variable following a Gamma distribution [8], [9], and assume $E[X]$ to be known from previous measurements. Thus, assuming independence the mean number of CPU cycles can be expressed as $L_i = D_i E[X]$.

We consider that time is partitioned into T time slots, and we denote by $\mathcal{T}=\{1,2,...,T\}$ the set of time slots. Each device can choose one time slot in which it wants

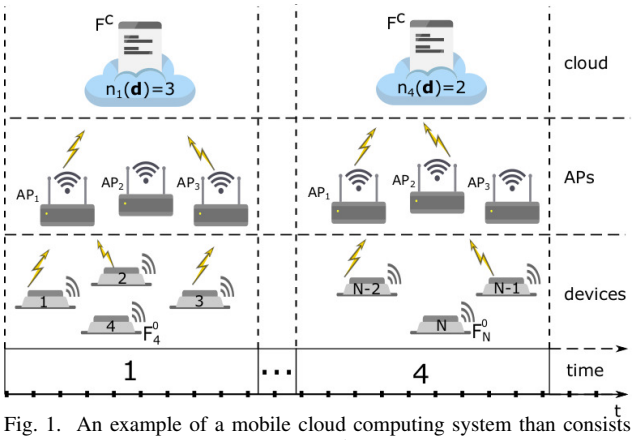

Fig. 1. An example of a mobile cloud computing system than consists of N devices, $T = 4$ time slots, and $A = 3$ APs.

to perform the computation and in the chosen time slot it can decide whether to perform the computation locally or to offload the computation to the cloud server through one of the APs. Therefore, each device $i \in \mathcal{N}$ can choose one element of the set $\mathfrak{D}_i = \{\mathcal{A} \cup \{0\}\} \times \mathcal{T}$, 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 its 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. The considered model of homogeneous task periodicities is reasonable for surveillance of homogeneous physical phenomena, we leave the case of heterogeneous periodicities to be subject of future work.

For a strategy profile $\mathbf{d}$ we denote by $O_{(t,a)}(\mathbf{d}) = \{i | d_i = (t, a)\}$ the set of devices that offload using AP a in time slot t, and we denote by $n_{(t,a)}(\mathbf{d}) = |O_{(t,a)}(\mathbf{d})|$ the number of devices that use AP a in time slot t. Furthermore, we define the set of all devices that offload in time slot t as $O_t(\mathbf{d}) = \cup_{a\in\mathcal{A}} O_{(t,a)}(\mathbf{d})$, and the total number of devices that offload in time slot t as $n_t(\mathbf{d})=\sum_{a\in\mathcal{A}} n_{(t,a)}(\mathbf{d})$. Finally, we denote by $O(\mathbf{d}) = \cup_{t\in\mathcal{T}} O_t(\mathbf{d})$ the set of all devices that offload in strategy profile $\mathbf{d}$.

A. Local computing

In the case of local computing each device has to use its own computing resources in order to perform the computation. We consider that different devices may have different computational capabilities and we denote by F_i^0 the computational capability of device i. Furthermore, we consider that the computational capability F_i^0 of device i is independent of the chosen time slot, and hence the time that is needed for device i to perform its computation task that requires L_i CPU cycles can be expressed as

$$T_i^0 = L_i / F_i^0. \tag{1}$$

In order to express the energy consumption in the case of local computing we denote by v_i the energy consumption per CPU cycle [10], and we express the energy that device i would spend on performing a computation task that requires L_i CPU cycles as

$$E_i^0 = v_i L_i. \tag{2}$$

B. Computation offloading

In the case of computation offloading the computation is performed in the cloud, but the input data for the computation task need to be transmitted through one of the APs. In what follows we introduce our communication and computation models that describe how the wireless medium and the cloud computing resources are shared among devices that offload their tasks, respectively.

1) Communication model: We consider that the uplink rate $\omega_{i,(t,a)}(\mathbf{d})$ that device i can achieve if it offloads through AP a in time slot t is a non-increasing function $f_a(n_{(t,a)}(\mathbf{d}))$ of the number $n_{(t,a)}(\mathbf{d})$ of devices that use the same AP a in time slot t. Furthermore, we consider that each device is characterized by PHY rate $R_{i,a}$, which depends on device specific parameters such as physical layer signal characteristics and the channel conditions. Therefore, the uplink rate of device i on AP a can be different from the uplink rates of the other devices on the same AP and can be expressed as

$$\omega_{i,(t,a)}(\mathbf{d}) = R_{i,a} \times f_a(n_{(t,a)}(\mathbf{d})). \tag{3}$$

This communication model can be used to model throughput sharing mechanisms in TDMA and OFDMA based MAC protocols [11].

Given the uplink rate $\omega_{i,(t,a)}(\mathbf{d})$, the time needed for device i to transmit the input data of size D_i through AP a in time slot t can be expressed as

$$T^{tx}_{i,(t,a)}(\mathbf{d}) = D_i / \omega_{i,(t,a)}(\mathbf{d}). \tag{4}$$

We consider that every device i knows the transmit power $P_{i,a}$ that it would use to transmit the data through AP a, where $P_{i,a}$ may be determined using one of the power control algorithms proposed in [12], [13]. The transmit power $P_{i,a}$ and the transmission time $T^{tx}_{i,(t,a)}(\mathbf{d})$ determine the energy consumption of device i for transmitting the input data of size D_i through AP a in time slot t

$$E^{tx}_{i,(t,a)}(\mathbf{d}) = P_{i,a} T^{tx}_{i,(t,a)}(\mathbf{d}). \tag{5}$$

2) Computation model: We denote by F^c the computational capability of the cloud service, and we consider that the computational capability $F^c_{i,t}(\mathbf{d})$ that device i would receive from the cloud in time slot t is a non-increasing function $f_i(n_t(\mathbf{d}))$ of the total number $n_t(\mathbf{d})$ of devices that offload in time slot t

$$F^c_{i,t}(\mathbf{d}) = F^c \times f_i(n_t(\mathbf{d})). \tag{6}$$

Therefore, the time needed for performing device i's task in the cloud may be different in different time slots, and given the number L_i of CPU cycles needed for the computation task it can be expressed as

$$T^{exe}_{i,t}(\mathbf{d}) = L_i / F^c_{i,t}(\mathbf{d}). \tag{7}$$

We consider that a single time slot is long enough for performing each user's task both in the case of local computing and in the case of computation offloading. This assumption is reasonable in the case of real time applications, where the worst-case task completion time must be less than a fraction of the periodicity.

Figure 1 shows an example of a mobile cloud computing system where devices can choose one slot out of four time slots to perform the computation. In the case of computation offloading, each device in the chosen time slot can offload its task to the cloud through one of three APs, e.g., in time slot 1 devices 1 and 2 offload their tasks through AP 1, device 3 offloads its task through AP 3, and device 4 performs the computation locally.

C. Cost Model

We consider that devices are interested in minimizing a linear combination of their computing time and their energy consumption, and denote by $0 \leq \gamma_i^T, \gamma_i^E \leq 1$ the corresponding weights, respectively. We can then express the cost of device i in the case of local computation as

$$C_i^0 = \gamma_i^T T_i^0 + \gamma_i^E E_i^0. \quad (8)$$

Similarly, we can express the cost of device i in the case of offloading through AP a in time slot t as

$$C_{i,(t,a)}^c(\mathbf{d}) = \gamma_i^T (T_{i,t}^{exe}(\mathbf{d}) + T_{i,(t,a)}^{tx}(\mathbf{d})) + \gamma_i^E E_{i,(t,a)}^{tx}(\mathbf{d}). \quad (9)$$

In (9) we made the common assumption that the time needed to transmit the result of the computation from the cloud service to the device can be neglected [5], [14], [15], [7], because for many applications (e.g., object recognition, tracking) the size of the output data is significantly smaller than the size D_i of the input data. We can thus express the cost of device i in strategy profile $\mathbf{d}$ as

$$C_i(\mathbf{d}) = \sum_{d_i \in \mathcal{T} \times \{0\}} \mathbf{1}_{(t,0)}(d_i) \cdot C_i^0 + \sum_{d_i \in \mathcal{T} \times \mathcal{A}} \mathbf{1}_{(t,a)}(d_i) \cdot C_{i,(t,a)}^c(\mathbf{d}), \quad (10)$$

where $\mathbf{1}_{(t,d)}(d_i)$ is the indicator function, i.e., $\mathbf{1}_{(t,d)}(d_i) = 1$ if $d_i = (t,d)$ and $\mathbf{1}_{(t,d)}(d_i) = 0$ otherwise.

D. Multi-slot computation offloading game

We consider that the objective of each device is to minimize its own total cost (10), i.e., to find a strategy

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

where $C_i(d_i, d_{-i})$ is the cost of device i if it chooses strategy d_i given the strategies d_{-i} of the other devices. Since devices may be autonomous entities with individual interests, we model the problem as a strategic game $\Gamma = < \mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i >$, in which the set of players is the set of devices (we use these two terms interchangeably). We refer to the game as the *multi-slot computation offloading game* (MSCOG). The MSCOG is a player specific network congestion game, as illustrated in Fig. 2.

Our objective is to answer the fundamental question whether there is a strategy profile from which no device would want to deviate, i.e., a pure strategy Nash equilibrium.

Definition 1. A pure strategy Nash equilibrium (NE) is a strategy profile $\mathbf{d}^*$ in which all players play their best replies to each others' strategies, that is,

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

Given a strategy profile $d = (d_i', d_{-i})$, an *improvement step* of device i is a strategy d_i' such that $C_i(d_i', d_{-i}) < C_i(d_i, d_{-i})$. A *best improvement* step is an improvement step that is a best reply. A *(best) improvement path* is a sequence of strategy profiles in which one device at a time changes its strategy through performing a *(best) improvement step*. We refer to the device that makes the best improvement step as the *deviator*. Observe that no device can perform a best improvement step in a NE.

III. Computing Equilibria

A. Single time slot ($T = 1$)

We start with considering the case $T = 1$, i.e., a single time slot.

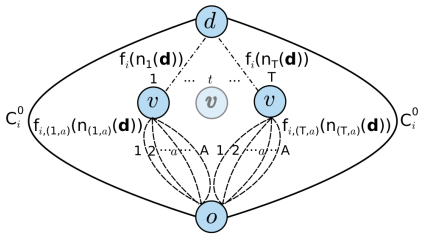

Fig. 2. Network model of the MSCOG.

Theorem 1. *The MSCOG for $T = 1$ possesses a pure strategy Nash equilibrium.*

Proof. We prove the result by showing that the game is best response equivalent to a player specific congestion game $\tilde{\Gamma}$ on a parallel network, i.e., a singleton player specific congestion game [16]. Observe that if for $T = 1$ we contract the edge (v, d) in the network shown in Fig. 2, i.e., if we replace the edge (v, d) and its two end vertices v and d by a single vertex, then we obtain a parallel network. Let us define the local computation cost of player i in $\tilde{\Gamma}$ as $\tilde{C^0}_i(N - n_1(\mathbf{d})) = C_i^0 - f_i(1 + n_1(\mathbf{d})) + c$, and the cost of offloading through AP a as $\tilde{f}_{i,a}(n_{(1,a)}(\mathbf{d})) = f_{i,a}(n_{(1,a)}(\mathbf{d})) + c$, where c is a suitably chosen constant to make all costs non-negative. Observe that due to the contraction of the edge (v, d) the offloading cost is $\tilde{C^c}_{i,a} = C_{i,a}^c - f_i(n_1(\mathbf{d}))$, and thus the difference between the cost function of player i in $\tilde{\Gamma}$ and that in Γ only depends on the strategies of the other players. This in fact implies that $\tilde{\Gamma}$ and Γ are best-response equivalent, and thus they have identical sets of pure strategy Nash equilibria. Since $\tilde{\Gamma}$ is a singleton player specific congestion game, it has a NE, and so does Γ, which proves the result. □

Furthermore, a Nash equilibrium of the MSCOG can be found in polynomial time.

Corollary 1. *Consider a MSCOG with $T = 1$ and N players. Let $\mathbf{d}^*$ be a Nash equilibrium of the game, and consider that a new player is added to the game. Then there is a sequence of best responses that leads to a NE.*

Proof. The result follows from the best response equivalence to $\tilde{\Gamma}$, and from the proof of Theorem 2 in [17]. □

Unfortunately, the contraction technique used in the proof of Theorem 1 cannot be applied for $T > 1$, as the resulting game would no longer be a congestion game.

B. Multiple time slots ($T \geq 1$)

In order to answer the question for $T \geq 1$ we first show that if a pure strategy NE exists for $T \geq 1$ then its structure cannot be arbitrary.

Theorem 2. *Assume that $\mathbf{d}^*$ is a NE of the MSCOG with $T \geq 1$. Then the following must hold*
(i) $\min_{t' \in \mathcal{T}} n_{t'}(\mathbf{d}^*) \leq n_t(\mathbf{d}^*) \leq \min_{t' \in \mathcal{T}} n_{t'}(\mathbf{d}^*) + 1$ *for* $\forall t, t' \in \mathcal{T}$,
(ii) if $n_t(\mathbf{d}^*) = n_{t'}(\mathbf{d}^*) + 1$ *for some* $t' \in \mathcal{T} \setminus \{t\}$, *then* $n_{(t,a)}(\mathbf{d}^*) \leq n_{(t',a)}(\mathbf{d}^*) + 1$ *for every AP* $a \in \mathcal{A}$, *and*
(iii) if $n_{(t,a)}(\mathbf{d}^*) = n_{(t',a)}(\mathbf{d}^*) - k$ *for* $k > 1$ *and* $t' \neq t$, *then* $n_{t'}(\mathbf{d}^*) \leq n_t(\mathbf{d}^*) \leq n_{t'}(\mathbf{d}^*) + 1$.

Proof. Clearly, all statements hold for $T = 1$. Assume that $T > 1$ and $\exists t, t' \in \mathcal{T}$ such that $n_t(\mathbf{d}^*) > n_{t'}(\mathbf{d}^*) + 1$. Then $\exists a \in \mathcal{A}$ such that $n_{(t,a)}(\mathbf{d}^*) \geq n_{(t',a)}(\mathbf{d}^*) + 1$. Therefore, player $i \in O_{(t,a)}(\mathbf{d}^*)$ could decrease her cost by changing

$\mathbf{d}^* = MB(\mathcal{N}, \mathcal{TA}, F^c, F_i^0)$

1: Let $N \leftarrow 1$
2: **for** $N = 1 \dots |\mathcal{N}|$ **do**
3: Let $A' \leftarrow \emptyset$ /*APs with decreased number of offloaders*/
4: Let $i \leftarrow N$
5: $d_i^* = \arg\min_{d \in \mathfrak{D}_i} C_i(d, \mathbf{d}^*(N-1))$
6: Let $\mathbf{d} \leftarrow (d_i^*, \mathbf{d}^*(N-1))$
7: **if** $d_i^* = (t, a)$ s.t. $a \in \mathcal{A}$ **then**
8: /*Players $j \in O_{(t,a)}(\mathbf{d})$ play best replies*/
9: $(\mathbf{d}', t', A') = DPD(\mathbf{d}, \mathbf{d}^*(N-1), (t, a), A')$
10: **if** $\exists j \in O_t(\mathbf{d}'), \exists d_j \in \mathfrak{D}_j$ s.t. $C_j(d_j, d'_{-j}) < C_j(d'_j, d'_{-j})$ **then**
11: /*Players $j \in O_t(\mathbf{d}')$ play best replies*/
12: $d_j = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d'_{-j})$
13: Let $\mathbf{d} \leftarrow (d_j, d'_{-j})$, Update A'
14: **if** $\exists i \in O_{d_i}(\mathbf{d}), d_i \neq \arg\min_{d \in \mathfrak{D}_i} C_i(d, d_{-i}) \notin A'$ **then**
15: Let $(t, a) \leftarrow d_j$, **go to** 9
16: **else**
17: Let $\mathbf{d}' \leftarrow \mathbf{d}$
18: **end if**
19: **end if**
20: **if** $A' \neq \emptyset$ **then**
21: /*Players $j \in O(\mathbf{d}') \cup L(\mathbf{d}')$ play best replies*/
22: $(\mathbf{d}, (t, a), A') = SID(\mathbf{d}', A')$
23: **if** $\exists i \in O_{(t,a)}(\mathbf{d}), d_i \neq \arg\min_{d \in \mathfrak{D}_i} C_i(d, d_{-i}) \notin A'$ **then**
24: **go to** 9
25: **else if** $\exists i \in O(\mathbf{d}) \cup L(\mathbf{d}), d_i \neq \arg\min_{d \in \mathfrak{D}_i} C_i(d, d_{-i}) \in A'$ **then**
26: Let $\mathbf{d}' \leftarrow \mathbf{d}$, **go to** 22
27: **end if**
28: **end if**
29: **end if**
30: Let $\mathbf{d}^*(N) \leftarrow \mathbf{d}'$
31: **end for**
32: **return** $\mathbf{d}^*(N)$

Fig. 3. Pseudo code of the MB algorithm.

the strategy to offloading through AP a in time slot t'. This contradicts $\mathbf{d}^*$ being a NE and proves (i).

We continue by proving (ii). Assume that there is an AP a such that $n_{(t,a)}(\mathbf{d}^*) > n_{(t',a)}(\mathbf{d}^*) + 1$ holds. Since $n_t(\mathbf{d}^*) = n_{t'}(\mathbf{d}^*) + 1$, we have that player $i \in O_{(t,a)}(\mathbf{d}^*)$ could decrease her cost by changing the strategy from (t,a) to (t',a). This contradicts $\mathbf{d}^*$ being a NE and proves (ii).

Finally, we prove (iii). First, assume that $n_t(\mathbf{d}^*) < n_{t'}(\mathbf{d}^*)$. Since $n_{(t,a)}(\mathbf{d}^*) < n_{(t',a)}(\mathbf{d}^*) - 1$, we have that player $i \in O_{(t',a)}(\mathbf{d}^*)$ could decrease her cost by changing the strategy from (t',a) to (t,a). This contradicts $\mathbf{d}^*$ being a NE and proves that $n_t(\mathbf{d}^*) \geq n_{t'}(\mathbf{d}^*)$. Second, assume that $n_t(\mathbf{d}^*) > n_{t'}(\mathbf{d}^*) + 1$ holds. Since $n_{(t,a)}(\mathbf{d}^*) < n_{(t',a)}(\mathbf{d}^*) - 1$, there is at least one AP $b \neq a$ such that $n_{(t,b)}(\mathbf{d}^*) \geq n_{(t',b)}(\mathbf{d}^*) + 1$, and thus player $i \in O_{(t,b)}(\mathbf{d}^*)$ could decrease her cost by changing the strategy to (t', b). This contradicts $\mathbf{d}^*$ being a NE and proves that $n_t(\mathbf{d}^*) \leq n_{t'}(\mathbf{d}^*) + 1$ must hold. □

In what follows we prove our main result concerning the existence of an equilibria in general case.

Theorem 3. *The MSCOG for $T \geq 1$ possesses a pure strategy Nash equilibrium.*

We provide the proof in the rest of the section.

C. The MyopicBest (MB) Algorithm

We prove Theorem 3 using the MB algorithm, shown in Fig. 3. The MB algorithm adds players one at a time, and lets them play their best replies given the other players' strategies. Our proof is thus based on an induction in the number N of players, and starts with the following result.

$(\mathbf{d}, t, A') = DPD(\mathbf{d}, \mathbf{d}^*(N-1), (t, a), A')$

1: /*Players that want to stop to offload*/
2: $D'_1 = \{j | d_j = (t, a), (t, 0) = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d_{-j})\}$
3: /*Player that want to change offloading strategy*/
4: $D'_2 = \{j | d_j = (t, a), (t', b) = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d_{-j}) \notin A', (t, a) \neq (t', b)\}$
5: **while** $|D'_1 \cup D'_2| > 0$ **do**
6: /*Players that want to stop to offload have priority*/
7: **if** $|D'_1| > 0$ **then**
8: Take $i \in D'_1$
9: $d_i = (t, 0)$
10: **else**
11: Take $i \in D'_2$
12: Let $d_i = \arg\min_{d \in \mathcal{T} \times \mathcal{A}} C_i(d, d_{-i})$
13: Let $(t, a) \leftarrow d_i$
14: **end if**
15: Let $\mathbf{d} \leftarrow (d_i, d_{-i})$
16: Update A', D'_1, D'_2
17: **end while**
18: **return** $(\mathbf{d}, t, A')$

Fig. 4. Pseudo code of the DPD algorithm.

Theorem 4. *The MB algorithm terminates in a NE for $N \leq T$.*

Proof. It is easy to see that if a strategy profile $\mathbf{d}^*(N)$ is a NE for $N \leq T$ then by Theorem 2 there is at most one player per time slot, and the MB algorithm computes such a strategy profile. □

We continue by considering the case $N > T$. Let us assume that for $N - 1 \geq T$ there is a NE $\mathbf{d}^*(N-1)$ and that upon induction step N a new player i enters the game and plays her best reply d_i^* with respect to $\mathbf{d}^*(N-1)$. After that, players can make best improvement steps one at a time starting from the strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$. If $d_i^* = (t, 0)$, then $n_{(t,a)}(\mathbf{d}) = n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for every $(t, a) \in \mathcal{T} \times \mathcal{A}$, and thus $\mathbf{d}$ is a NE. Otherwise, if $d_i^* = (t, a)$, for some $a \in \mathcal{A}$, some players $j \in O_{(t,a)}(\mathbf{d})$ may have an incentive to make an improvement step because their communication and cloud computing costs have increased, and some players $j \in O_t(\mathbf{d}) \setminus O_{(t,a)}(\mathbf{d})$ may have an incentive to make an improvement step because their cloud computing cost has increased. Among these players, the MB algorithm allows players $j \in O_{(t,a)}(\mathbf{d})$ to perform best improvement steps, using the *DoublePokeDeviator* (DPD) algorithm shown in Fig. 4. There are two types of players that can make a best improvement step using the DPD algorithm. The first type are players $j \in O_{(t,a)}(\mathbf{d})$ for which a best reply is to stop to offload. The second type are players $j \in O_{(t,a)}(\mathbf{d})$ for which a best reply is an offloading strategy $(t', b) \in \mathcal{T} \times \mathcal{A} \setminus \{(t, a)\}$ for which the number of offloaders in $\mathbf{d}$ is not smaller than the number of offloaders in the NE $\mathbf{d}^*(N-1)$. The DPD algorithm allows either one player of the first type, or one player of the second type to perform a best improvement step, and as we show next it terminates in a finite number of steps.

Proposition 1. *Let $\mathbf{d}$ be a strategy profile in which there is at least one player $j \in O_{(t,a)}(\mathbf{d})$ that can be chosen by the DPD algorithm. Then the length of a best improvement path generated by the DPD algorithm is at most $N - 1$.*

Proof. Let us denote by $\mathbf{d}'$ a strategy profile after a player $j \in O_{(t,a)}(\mathbf{d})$ performs its best improvement step. First, observe that if player j's best improvement step is to stop to offload, then the DPD algorithm terminates since it

allows only players that play the same strategy as the last deviator to perform best improvement steps. Furthermore, if $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$, then $n_{(t,a)}(\mathbf{d}') = n_{(t,a)}(\mathbf{d}^*(N-1))$ for every $(t,a) \in \mathcal{T} \times \mathcal{A}$, and thus $\mathbf{d}'$ is a NE.

Otherwise, if player j's best improvement step is $(t',b) \in \mathcal{T} \times \mathcal{A} \setminus \{(t,a)\}$, then $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d}) + 1$ holds, and we can have one of the following: (1) there is no player $j' \in O_{(t',b)}(\mathbf{d})$ that wants to deviate from (t',b), (2) there is a player $j' \in O_{(t',b)}(\mathbf{d})$ that wants to deviate from (t',b).

If case (1) happens then the DPD algorithm terminates, because there is no player that plays the same strategy as the last deviator and that can decrease its cost using the DPD algorithm. Otherwise, if case (2) happens then a new best improvement step can be triggered, which will bring the system to a state where $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d})$ holds.

In what follows we show that none of the players that has changed its offloading strategy in one of the previous best improvement steps would have an incentive to deviate again. Let us consider a player j' that changed its strategy from (t',b) to another offloading strategy, and let us assume that in one of the subsequent best improvement steps one of the players changes its offloading strategy to (t',b), and thus it brings the system to a state where $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d}) + 1$ holds. We observe that player j that has changed its strategy from (t,a) to (t',b) before player j' deviated from (t',b) would have no incentive to deviate from its strategy (t',b) after a new player starts offloading through AP b in time slot t'. This is because (t',b) was its best response while player j' was still offloading through AP b in time slot t', i.e, while $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d}) + 1$ was true. Therefore, a new best improvement step can be triggered only if there is another player that wants to change from (t',b) to another offloading strategy. If this happens, $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d})$ will hold again, and thus the maximum number of players that offload through AP b in time slot t' will be at most $n_{(t',b)}(\mathbf{d})+1$ in all subsequent best improvement steps. Consequently, player j would have no incentive to leave AP b in time slot t' in the subsequent steps. Therefore, each player deviates at most once in a best improvement path generated by the DPD algorithm, and thus the algorithm terminates in at most $N-1$ best improvement steps, which proves the proposition. □

The DPD algorithm may be called multiple times during the execution of the MB algorithm, but as we show next for any fixed N, it is called a finite number of times.

Proposition 2. *The DPD algorithm is executed a finite number of times for any particular N.*

Proof. Let us assume that the DPD algorithm has been called at least once during the execution of the MB algorithm, and let us denote by $\mathbf{d}'$ the most recent strategy profile computed by the DPD algorithm. Now, let us assume that in the next best improvement step generated by the MB algorithm a player $i \in O(\mathbf{d}') \cup L(\mathbf{d}')$ changes its strategy to $(t,a) \in \mathcal{T} \times \mathcal{A}$. Starting from a strategy profile $\mathbf{d} = ((t,a), d'_{-i})$ players $j \in O_{(t,a)}(\mathbf{d})$ are allowed to perform the next best improvement step using the DPD algorithm.

Observe that players $j' \in O_{(t,a)}(\mathbf{d}')$ that in the previous best improvement steps changed their strategy to (t,a) using the DPD algorithm and triggered one of the players to leave the same strategy (t,a) would have no incentive to perform a best improvement step using the DPD algorithm. This is because the previous deviators $j' \in O_{(t,a)}(\mathbf{d}')$ brought $n_{(t,a)}(\mathbf{d}')$ to its maximum, that is to $n_{(t,a)}(\mathbf{d}^*(N-1)) + 1$, which decreased again to $n_{(t,a)}(\mathbf{d}^*(N-1))$ after the next deviator left strategy (t,a). Since the number of previous deviators $j' \in O_{(t,a)}(\mathbf{d}')$ that have no incentive to perform a new best improvement step using the DPD algorithm increases with every new best improvement path generated by the DPD algorithm, players will stop performing best improvement steps using the DPD algorithm eventually, which proves the proposition. □

So far we have proven that the DPD algorithm generates a finite number of finite best improvement paths. In the following we use this result for proving the convergence of the MB algorithm.

Proof of Theorem 3. We continue with considering all conditions under which the DPD algorithm may have terminated. First, let us assume that the last deviator's best improvement step is a strategy within time slot t'. The proof of Proposition 1 shows that the DPD algorithm terminates if one of the following happens: (i) starting from a strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$ all players performed their best improvement steps, (ii) some players did not deviate and the last deviator's strategy was $(t',0)$, i.e., the last deviator changed to local computing in time slot t', (iii) some players did not deviate and there was no player that wanted to change from the last deviator's strategy $(t',b) \in \mathcal{T} \times \mathcal{A}$.

Let us first consider case (i), and the last deviator that performed its best improvement step. If its best improvement step was to stop to offload, $n_{(t,a)}(\mathbf{d}') = n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for every $(t,a) \in \mathcal{T} \times \mathcal{A}$. Otherwise, if a best improvement step of the last deviator was to change its offloading strategy to (t',b), we have that $n_{(t,a)}(\mathbf{d}') \geq n_{(t,a)}(\mathbf{d}^*(N-1))$ for every $(t,a) \in \mathcal{T} \times \mathcal{A}$, where the strict inequality holds only for (t',b), and $n_{(t',b)}(\mathbf{d}') = n_{(t',b)}(\mathbf{d}^*(N-1)) + 1$. Since there is no offloading strategy for which the number of offloaders is less than the number of offloaders in the NE $\mathbf{d}^*(N-1)$, there is no player $j \in O(\mathbf{d}')$ that can decrease its offloading cost. Furthermore, there is no player that wants to change its strategy from local computing to offloading, and thus a strategy profile computed by the DPD algorithm is a NE.

If case (ii) or case (iii) happen the MB algorithm allows players that offload in the same time slot as the last deviator to perform any type of best improvement steps. Furthermore, if case (ii) happens and there are no APs with decreased number of offloaders compared with the NE $\mathbf{d}^*(N-1)$, i.e., $n_{(t,a)}(\mathbf{d}') = n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for every $(t,a) \in \mathcal{T} \times \mathcal{A}$, then the strategy profile $\mathbf{d}'$ computed by the DPD algorithm is a NE. Observe that $n_{(t,a)}(\mathbf{d}') = n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for every $(t,a) \in \mathcal{T} \times \mathcal{A}$ if strategy profile $\mathbf{d}'$ is obtained by the DPD algorithm starting from strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$.

Otherwise, if case (ii) happens such that there is a strategy $(t,a) \in \mathcal{T} \times \mathcal{A}$ for which $n_{(t,a)}(\mathbf{d}') < n_{(t,a)}(\mathbf{d}^*(N-1))$ holds, then players $j \in O_{t'}(\mathbf{d}')$ that offload in the same time slot as the last deviator may want to change their offloading strategy to (t,a). Let us assume that there is a player

$j \in O_{t'}(\mathbf{d}')$ that wants to change its offloading strategy to (t,a) and let us denote by $\mathbf{d}$ a resulting strategy profile. Since $n_{(t,a)}(\mathbf{d}) = n_{(t,a)}(\mathbf{d}') + 1$ and $n_t(\mathbf{d}) = n_t(\mathbf{d}') + 1$ hold, some players $j \in O_{(t,a)}(\mathbf{d})$ may want to perform a best improvement step using the DPD algorithm, which can happen only a finite number of times accoring to Proposition 2.

We continue the analysis by considering case (iii). Observe that if there is a strategy (t,a) for which $n_{(t,a)}(\mathbf{d}') < n_{(t,a)}(\mathbf{d}^*(N-1))$ players $j \in O_{t'}(\mathbf{d}')$ that offload in the same time slot as the last deviator may want to change their offloading strategy to (t,a). Furthermore, players $j \in O_{t'}(\mathbf{d}') \setminus O_{(t',b)}(\mathbf{d}')$ may want to stop to offload or to change to any offloading strategy $(t,a) \in \mathcal{T} \times \mathcal{A} \setminus \{(t',b)\}$ since their cloud computing cost increased. Let us assume that there is a player $j \in O_{t'}(\mathbf{d}')$ that wants to change its offloading strategy to $(t,a) \in \mathcal{T} \times \mathcal{A} \setminus \{(t',b)\}$ and let us denote by $\mathbf{d}$ the resulting strategy profile. Since $n_{(t,a)}(\mathbf{d}) = n_{(t,a)}(\mathbf{d}') + 1$ and $n_t(\mathbf{d}) = n_t(\mathbf{d}') + 1$ hold, some players $j \in O_{(t,a)}(\mathbf{d})$ may want to perform a best improvement step using the DPD algorithm, which can happen only a finite number of times accoring to Proposition 2.

If case (ii) or case (iii) happens and there is no player $j \in O_{t'}(\mathbf{d}')$ that wants to deviate, the MB algorithm allows players from the other time slots $t \in \mathcal{T} \setminus \{t'\}$ to perform best improvement steps using *SelfImposedDeviator* (SID) algorithm shown in Fig. 5. Observe that players from time slots $t \in \mathcal{T} \setminus \{t'\}$ are not poked to deviate by the other players, and only reason why they would have an incentive to deviate is that $n_{(t,a)}(\mathbf{d}') < n_{(t,a)}(\mathbf{d}^*(N-1))$ holds for some strategies $(t,a) \in \mathcal{T} \times \mathcal{A}$. The SID algorithm first allows one of the players $j \in O(\mathbf{d}') \setminus O_{t'}(\mathbf{d}')$ that already offloads to perform a best improvement step, and if there is no such player the SID algorithm allows one of the players $j \in L(\mathbf{d}')$ that performs computation locally to start to offload. Let us assume that there is a strategy (t,a) for which $n_{(t,a)}(\mathbf{d}') < n_{(t,a)}(\mathbf{d}^*(N-1))$ holds and that there is a player $j \in O(\mathbf{d}') \setminus O_{t'}(\mathbf{d}') \cup L(\mathbf{d}')$ that wants to deviate to strategy (t,a). We denote by $\mathbf{d}$ the resulting strategy profile, after player j performs its best improvement step. Since $n_{(t,a)}(\mathbf{d}) = n_{(t,a)}(\mathbf{d}') + 1$ and $n_t(\mathbf{d}) = n_t(\mathbf{d}') + 1$ hold, some players $j \in O_{(t,a)}(\mathbf{d})$ may want to perform a best improvement step using the DPD algorithm, which can happen only a finite number of times accoring to Proposition 2. Finally, let us consider case (iii) such that there is a player $j \in O_{t'}(\mathbf{d}') \setminus O_{(t',b)}(\mathbf{d}')$ that wants to stop to offload because its cloud computing cost increased. Let us denote by $\mathbf{d}$ a strategy profile after player j changes its strategy from $(t',a) \neq (t',b)$ to local computing. We have that $n_{(t',a)}(\mathbf{d}) = n_{(t',a)}(\mathbf{d}') - 1$, and if $n_{(t',a)}(\mathbf{d}') = n_{(t',a)}(\mathbf{d}^*(N-1))$ we have that players $j' \in O(\mathbf{d}) \setminus O_{(t',a)}(\mathbf{d})$ may have an incentive to change their offloading strategy to (t',a) if doing so decreases their offloading cost. We have seen that a best improvement step of this type can trigger the DPD algorithm a finite number of times according to Proposition 2. Now, let us assume that a player $j' \in O_{(t,b)}(\mathbf{d})$, where $(t,b) \in \mathcal{T} \times \mathcal{A} \setminus \{(t',a)\}$, changes its offloading strategy from (t,b) to (t',a), and that by doing so it does not trigger the DPD algorithm. The resulting strategy profile $\mathbf{d} = ((t',a), d_{-j'})$ is such that $n_{(t,b)}(\mathbf{d}) = n_{(t,b)}(\mathbf{d}') - 1$ holds, and if $n_{(t,b)}(\mathbf{d}') = n_{(t,b)}(\mathbf{d}^*(N-1))$ some players may have an incentive to change their offloading strategy to (t,b) if doing so decreases their offloading cost.

$(\mathbf{d}, (t,a), A') = SID(\mathbf{d}, A')$

```
/*Players that offload and can decrease their offloading cost*/
D1={j∈O(d)|(t,a)=arg min_{d∈D_j} C_j(d,d_{-j})∈A', d_j≠(t,a)}
/*Players that compute locally and want to start to offload*/
D2={j∈L(d)|(t,a)=arg min_{d∈D_j} C_j(d,d_{-j}) ∈ A'}
if |D1 ∪ D2| ≠ ∅ then
  /*Players that offload have priority*/
  if D1 ≠ ∅ then
    Take i ∈ D1
  else if D2 ≠ ∅ then
    Take i ∈ D2
  end if
  d'_i = arg min_{d∈D_i} C_i(d, d_{-i})
  Let d ← (d'_i, d_{-i})
  Let (t,a) ← d'_i
  Update A'
end if
return (d, (t,a), A')
```

Fig. 5. Pseudo code of the SID algorithm.

We continue by considering the case where all subsequent best improvement steps are such that deviators change to a strategy for which the number of offloaders is less than the number of offloaders in the NE $\mathbf{d}^*(N-1)$ and by doing so they do not trigger the DPD algorithm. Therefore, the resulting best improvement path is such that the cost of each deviator decreases with every new best improvement step it makes. Assume now that after $k \geq 2$ improvement steps player j' wants to return back to strategy (t,b). By the definition of the resulting best improvement path, the cost of player j' in the $(k+1)$-th improvement step is not only less than the cost in the k-th best improvement step, but also less than its cost in the first best improvement step. Therefore, player j' will not return to a strategy it deviated from, and thus it will deviate at most $T \times A - 1$ times. Consequently, when there are no players that can trigger the DPD algorithm, players that change their startegy from local computing to offloading using the SID algorithm, can only decrease their offloading cost in the subsequent best improvement steps, and thus they would have no incentive to stop to offload. Since the number of players is finite, the players will stop changing from local computing to offloading eventually, which proves the theorem. □

Even though the convergence proof of the MB algorithm is fairly involved, the algorithm itself is computationally efficient, as we show next.

Theorem 5. *When a new player i enters the game in an equilibrium $\mathbf{d}^*(N-1)$, the MB algorithm computes a new equilibrium $\mathbf{d}^*(N)$ after at most $N \times T \times A - 2$ best improvement steps.*

Proof. In the worst case scenario the DPD algorithm generates an $N-2$ steps long best improvement path, and a player that offloads in the same time slot as the last deviator, but not through the same AP changes to local computing, because its cloud computing cost increased. Observe that the worst case scenario can happen only if $|O(\mathbf{d}^*(N-1))| = N-1$ holds. Furthermore, $N-2$ players will have an opportunity to deviate using the DPD

algorithm and a player that offloads in the same time slot as the last deviator will have an opportunity to stop to offload only if $n_{(t,a)}(\mathbf{d}^*(N-1)) = n_{(t',b)}(\mathbf{d}^*(N-1))$ holds for every $(t,a),(t',b) \in \mathcal{T} \times \mathcal{A}$. Furthermore, in the worst case scenario, the best improvement path generated by the DPD algorithm is followed by an $N \times (T \times A - 1)$ long best improvement path, in which deviators change to a strategy for which the number of offloaders is less than the number of offloaders in the NE $\mathbf{d}^*(N-1)$ and by doing so they do not trigger the DPD algorithm. Therefore, a NE can be computed in at most $N - 2 + N \times (T \times A - 1)$ best improvement steps. □

By addding players one at a time, it follows that the MB algorithm has quadratic worst case complexity.

Theorem 6. *The MB algorithm computes a NE allocation in $O(N^2 \times T \times A)$ time.*

Implementation considerations: The MB algorithm can be implemented in a decentralized manner, by letting devices perform the best improvement steps one at a time. For computing a best response, besides its local parameters (e.g. D_i, L_i, F_i^0), each device i requires information about achievable uplink rates, available MEC resources, and the number of users sharing the APs and the cloud. In practice these information can be provided by the MEC. As discussed in [5], [18], [7], two main advantages of such a decentralized implementation compared to a centralized one are that the MEC can be relieved from complex centralized management, and devices do not need to reveal their parameters, but only their most recent decisions.

IV. Numerical Results

In the following we show simulation results to evaluate the cost performance and the computational efficiency of the MB algorithm. We consider that the devices are placed uniformly at random over a square area of $1km \times 1km$, while the APs are placed at random on a *regular grid* with A^2 points defined over the area. We consider that the channel gain of device i to AP a is proportional to $d_{i,a}^{-\alpha}$, where $d_{i,a}$ is the distance between device 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 suburban areas [19]. For simplicity we assign a bandwidth of 5 *MHz* to every AP a, and the data transmit power of $P_{i,a}$ is drawn from a continuous uniform distribution on $[0.05, 0.18]$ *W* according to measurements reported in [20]. We consider that the uplink rate of a device connected to an AP a scales directly proportional with the number of devices offloading through AP a. The computational capability F_i^0 of device i is drawn from a continuous uniform distribution on $[0.5, 1]$ *GHz*, while the computation capability of the cloud is $F^c = 100$ *GHz* [21]. We consider that the computational capability that a device receives from the cloud scales inversely proportional with the number of devices that offload. 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 is set to $10^{-11}(F_i^0)^2$ according to measurements reported in [10], [9]. The weights attributed to energy consumption γ_i^E and the response time γ_i^T are drawn from a continuous uniform distribution on $[0, 1]$.

We use three algorithms as a basis for comparison for the proposed *MB* algorithm. In the first algorithm players choose a time slot at random, and implement an equilibrium allocation within their chosen time slots. We refer to this algorithm as the *RandomSlot* (RS) algorithm. The second algorithm considers that all devices perform local execution. The third algorithm is a worst case scenario where all devices choose the same time slot and implement an equilibrium allocation within that time slot. Observe that this corresponds to $T = 1$. We define the *performance gain* of an algorithm as the ratio between the system cost reached when all devices perform local execution and the system cost reached by the algorithm. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

A. Performance gain vs number of devices

Fig. 6 shows the *performance gain* as a function of the number N of devices for $A = 4$ APs. The results show that the *performance gain* decreases with the number of devices for the MB algorithm for all values of T, for the RS algorithm and for the deterministic worst case $T = 1$. This is due to that the APs and the cloud get congested as the number of devices increases. The performance gain of the MB algorithm is up to 50% higher than that of the RS algorithm for $T > 1$; the gap between the two algorithms is largest when the ratio N/T is approximately equal to 4. The reason is that as T increases the average number of offloaders per time slot remains balanced in the case of the MB algorithm. On the contrary, in the case of the RS algorithm some time slots may be more congested than others, since the players choose their time slot at random. However, the average imbalance in the number of offloaders per time slot decreases as the number of devices increases, thus the results are similar for large values of N. At the same time, the performance gain of the MB algorithm compared to that of the deterministic worst case $T = 1$ is almost proportional to the number T of time slots, and shows that coordination is essential for preventing severe performance degradation. It is also interesting to note that for $T = 1$ the *performance gain* decreases with N at a much higher rate than for $T > 1$, which is due to the fast decrease of the number of offloaders, as we show next.

Fig. 7 shows the ratio of players that offload for the same set of parameters as in Fig. 6. The results show that in the worst case, for $T = 1$, the ratio of players that offload decreases almost linearly with N, which explains the fast decrease of the *performance gain* observed in Fig. 6. On the contrary, for larger values of T the ratio of players that offload appears less sensitive to N. We observe that the ratio of players that offload is in general higher in equilibrium than in the strategy profile computed by the RS algorithm, which explains the superior performance of MB observed in Fig. 6.

B. Performance gain vs number of APs

Fig. 8 shows the *performance gain* as a function of the number A of APs for $N = 50$ devices. We observe that the *performance gain* achieved by the algorithms increases

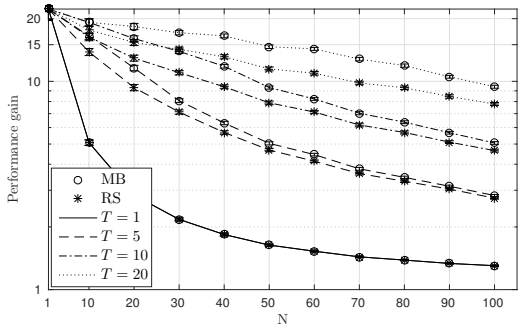

Fig. 6. *Performance gain* vs number of devices (N).

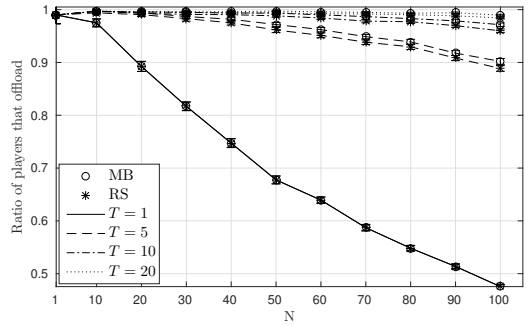

Fig. 7. Ratio of offloaders vs. number of devices (N).

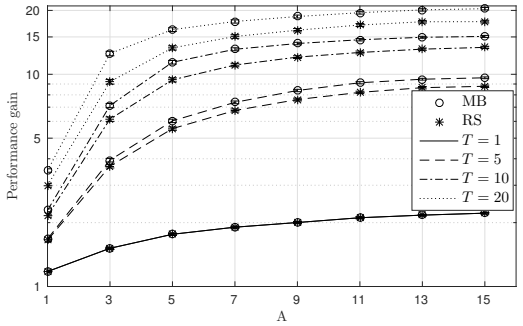

Fig. 8. *Performance gain* vs number of APs (A).

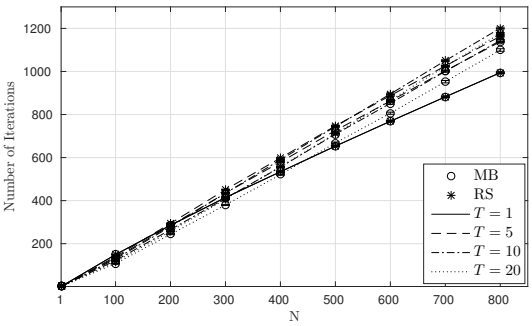

Fig. 9. Number of iterations vs number of devices (N).

monotonically with the number of APs for all values of T with a decreasing marginal gain. The reason is that once $T \times A \geq N$ every device can offload its task through its favorite AP without sharing it, and hence the largest part of the offloading cost comes from the computing cost in the cloud. However, a small change in the *performance gain* is still present even for very large values of A because the density of the APs over a region becomes larger as A increases, and hence the channel gain, which depends on the distance between the device and the APs becomes larger on average. The results also show that MB always outperforms RS, and its *performance gain* compared to that of RS increases with T. Most importantly, the number of APs required for a certain performance gain is almost 50% lower using the MB algorithm compared to the RS algorithm for higher values of T, i.e., significant savings can be achieved in terms of infrastructural investments.

C. Computational Complexity

In order to assess the computational efficiency of the MB algorithm we consider the number of iterations, defined as the number of induction steps plus the total number of update steps over all induction steps needed to compute a NE. Fig. 9 shows the number of iterations as a function of the number N of devices for $A = 4$ APs. The results show that the number of iterations scales approximately linearly with N for both algorithms, and indicates that the worst case scenario considered in Theorem 6 is unlikely to happen. The first interesting feature of Fig. 9 is that the number of iterations is slightly less in the case of the MB algorithm than in the case of the RS algorithm for all values of T, except for $T = 1$ for which the two algorithms are equivalent. The reason is that in the case of the MB algorithm the number of offloaders per time slot is more balanced, and hence the devices have less incentive to deviate when a new device enters the system, and their updates are always at least as good as in the case of RS algorithm, since the MB algorithm allows devices to change between time slots. On the contrary, in the case of the RS algorithm some of the time slots may be very congested, and the devices that offload within these time slots have a higher incentive to deviate when a new device enters the system. The second interesting feature of Fig. 9 is that the number of iterations is smaller for larger values of T for smaller values of N, but for larger values of N the results are reversed. The reason is that for smaller values of N the time slots are less congested on average as T increases, and hence the devices do not want to update their strategies so often. On the contrary, as N increases the benefit of large values of T becomes smaller, because the congestion per time slots increases, and hence devices may want to update their strategies more often.

Overall, our results show that the proposed MB algorithm can compute efficient allocations for periodic task offloading at low computational complexity.

V. Related Work

The scheduling of periodic tasks received significant attention for real-time systems [22], [23], but without considering communications. Similarily, the scheduling of communication resources has been considered without considering computation [24]. Most works that considered both communication and computation considered a single device [25], [10], [6], [26], [27], and thus they do not consider the allocation of resources between devices.

Related to our work are recent works on energy efficient computation offloading for multiple mobile users [28], [29], [30]. [28] proposed a genetic algorithm for maximizing the throughput in a partitioning problem for mobile data stream applications, while [29] proposed a heuristic for minimizing the users' cost in a two-tiered cloud infrastructure with user mobility in a location-time workflow framework. [30] considered minimizing mobile users' energy consumption by joint allocation of wireless and cloud resources, and proposed an iterative algorithm.

A few recent works provided a game theoretic treatment of the mobile computation offloading problem for a

single time slot [31], [32], [5], [18], [33], [34], [7]. [31] considers a two-stage game, where first each mobile user chooses the parts of its task to offload, and then the cloud allocates computational resources to the offloaded parts. [32] considered a three-tier cloud architecture, and provided a distributed algorithm for the computing a mixed strategy equilibrium. [33] considered tasks that arrive simultaneously and a single wireless link, and showed the existence of equilibria when all mobile users have the same delay budget. [5] showed that assuming a single wireless link and link rates determined by the Shannon capacity of an interference channel, the resulting game is a potential game. [18] 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. [7] considered multiple wireless links, equal bandwidth sharing and a non-elastic cloud, and provided a polynomical time algorithm for computing equilibria. Compared to these works, our model of periodic tasks considers the scheduling of tasks over time slots and wireless resources, and is thus a first step towards bridging the gap between early works on scheduling [23] and recent works on computation offloading [5], [7].

From a game theoretical perspective the importance of our contribution is the analysis of a player-specific network congestion game for which the existence of equilibria is not known in general [16], thus the proposed algorithm and our proof of existence advance the state of the art in the study of equilibria in network congestion games.

VI. Conclusion

We provided a game theoretic treatment of computation offloading for periodic tasks. We proved the existence of equilibrium allocations, characterized their structure and provided a polynomial time decentralized algorithm for computing equilibria. Simulations show that the proposed algorithm achieves good system performance for a wide range of system sizes and task periodicities. Our results show that periodic computation offloading can be efficiently coordinated using low complexity algorithms despite the vast solution space and the combinatorial nature of the problem. An interesting open question is whether our results can be extended to devices with heterogeneous periodicities, we leave this question subject of future work.

References

[1] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "Pipeneta wireless sensor network for pipeline monitoring," in *Proc. of IPSN*, 2007, pp. 264–273.

[2] S. Oh, P. Chen, M. Manzo, and S. Sastry, "Instrumenting wireless sensor networks for real-time surveillance," in *Proc. IEEE ICRA*, May 2006, pp. 3128–3133.

[3] X. Zhu, S. Han, P. C. Huang, A. K. Mok, and D. Chen, "Mbstar: A real-time communication protocol for wireless body area networks," in *Proc. of ERCTS*, Jul. 2011, pp. 57–66.

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

[5] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *Proc. of IEEE PDS*, pp. 974–983, 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] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of IEEE INFOCOM*, May 2017.

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

[9] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of Usenix HotCloud*, 2010.

[10] 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.

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

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

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

[14] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[15] 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.

[16] I. Milchtaich, "The equilibrium existence problem in finite network congestion games," in *Proc. of WINE*, 2006, pp. 87–98.

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

[18] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

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

[20] E. Casilari, J. M. Cano-García, and G. Campos-Garrido, "Modeling of current consumption in 802.15. 4/zigbee sensor motes," *Sensors*, vol. 10, no. 6, pp. 5443–5468, 2010.

[21] 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.

[22] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Trans. on Computers*, vol. 39, pp. 1175–1185, Sep. 1990.

[23] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-Time Syst.*, vol. 28, no. 2-3, pp. 101–155, Nov. 2004.

[24] I. H. Hou, "Packet scheduling for real-time surveillance in multihop wireless sensor networks with lossy channels," *IEEE Trans. on Wireless Comm.*, vol. 14, no. 2, pp. 1071–1079, Feb 2015.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] V. Cardellini et al., "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, 2015.

[33] 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.

[34] 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.