Computation Offloading Scheduling for Periodic Tasks in Mobile Edge Computing

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

***Abstract*—Motivated by various delay sensitive applications, we address the problem of coordinating the offloading decisions of wireless devices that periodically generate computationally intensive tasks. We consider autonomous devices that aim at minimizing their own cost by choosing when to perform their tasks and whether or not to offload their tasks to an edge cloud through one of the multiple wireless links. We develop a game theoretical model of the problem, prove the existence of pure strategy Nash equilibria and propose a polynomial complexity algorithm for computing an equilibrium. Furthermore, we characterize the structure of the equilibria, and by providing an upper bound on the price of anarchy of the game we establish an asymptotically tight bound on the approximation ratio of the proposed algorithm. Our simulation results show that the proposed algorithm achieves significant performance gain compared to uncoordinated computation offloading at a computational complexity that is on average linear in the number of devices.**

***Index terms*—** computation offloading, edge computing, game theory, decentralized resource management

I. Introduction

The emergence of affordable wireless sensors, such as cameras, has given rise to a variety of Internet of Things (IoT) applications, including surveillance [1], tracking [2] and traffic monitoring [3]. These applications typically involve the periodic collection of sensory data, which need to be processed in a timely manner to ensure the stability of potential feedback control loops. Processing often involves some form of machine learning, e.g., visual analysis, which may be too computationally intensive to be performed in the sensors.

A promising solution to enable the timely processing of computationally intensive IoT tasks is mobile edge computing (MEC) [4]. Opposed to traditional remote cloud infrastructures such as Amazon and Azure [5], MEC provides computing resources close to the end users, i.e., at the network edge, which makes it a better candidate for meeting the requirements of delay sensitive IoT applications.

By offloading the computation to nearby edge clouds [6], devices may be able to significantly reduce their response times and energy consumption and thus to extend the lifetime of their batteries. Nevertheless, without coordination of their offloading decisions, devices in MEC systems might experience poor performance due to contention for communication and computing resources. Therefore, in order to use MEC systems to their full potential the autonomous devices need to coordinate their offloading decisions over time and across communication and computing resources.

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

There are several challenges facing the coordination of offloading decisions of autonomous devices in MEC systems. First, edge clouds are not as computationally powerful as remote clouds, and thus the response time and energy consumption of devices may be affected by contention for both communication and computing resources [7], [8], [9]. Second, MEC systems are likely to combine heterogeneous communication and computing resources, and thus coordination of offloading decisions involves not only deciding whether or not to offload the tasks, but also which of the communication and computing resources to use in the case of offloading. Third, IoT devices such as vehicles, drones and manufacturing machines [10] may be autonomous entities with different computing capabilities and tasks, and thus with individual interests in terms of response time and energy consumption requirements [11], [12]. Finally, besides the allocation of communication and computing resources, coordination for offloading periodic tasks involves deciding when to process the collected sensory data and it may also affect the optimal time for sensing, so as to minimize the age of information [13], [14]. All these challenges make the problem of coordinating the offloading decisions of autonomous devices inherently difficult.

In this paper we address this problem for a MEC system in which the devices aim at minimizing their cost defined as a linear combination of the task completion time and the energy consumption. Each device can choose autonomously the time slot for performing its periodic task and in the chosen time slot it can decide whether to perform the task locally or to offload it to an edge cloud through one of multiple heterogeneous wireless links.

We make three important contributions to solve the problem. First, based on a game theoretical treatment of the problem, we propose a polynomial time decentralized algorithm for coordinating the offloading decisions of the devices, and prove the convergence of the algorithm to a pure strategy Nash equilibrium. Second, we characterize the structure of the computed equilibrium. Third, we establish an asymptotically tight upper bound on the price of anarchy of the game, and by doing so we show that a the proposed algorithm has a bounded approximation ratio. We use simulations to assess the performance of the proposed algorithm in a variety of scenarios. Our results show that the algorithm can efficiently coordinate the offloading decisions of autonomous devices with periodic tasks at low computational complexity.

The rest of the paper is organized as follows. In Section II we present the system model and the problem

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Description
$\mathcal{N}$	Set of N devices
$\mathcal{A}$	Set of A APs
$\mathcal{T}$	Set of T time sots
D_i	Mean size of the input data for device i
L_i	Mean task complexity for device i
F_i^0	Computational capability of device i
v_i	Energy consumption of device i per CPU cycle
γ_i^T	Completion time weight for device i
γ_i^E	Energy consumption weight for device i
T_i^0	Local execution time for device i
E_i^0	Local execution energy consumption for device i
C_i^0	Local computing cost for device i
$R_{i,a}$	Uplink PHY rate of device i towards AP a
$P_{i,a}$	Transmit power of device i towards AP a
F^c	Cloud computing capability
$\mathfrak{D}_i$	Set of feasible decisions for device i
d_i	Decision of device i, $d_i! \in \mathfrak{D}_i$
$\mathbf{d}$	Strategy profile
$O_{(t,a)}(\mathbf{d})$	Set of $n_{(t,a)}(\mathbf{d})$ devices $i \in \mathcal{N}$ s.t. $d_i = (t,a)$ in $\mathbf{d}$
$O_{(t,c)}(\mathbf{d})$	Set of $n_{(t,c)}(\mathbf{d})$ devices offloading in time slot t in $\mathbf{d}$
$O(\mathbf{d})$	Set of all devices that offload their tasks in $\mathbf{d}$
$F_{i,t}^c(\mathbf{d})$	Cloud computing capability assigned to device i in $\mathbf{d}$
$\omega_{i,(t,a)}(\mathbf{d})$	Uplink rate of device i, $d_i = (t,a)$ in $\mathbf{d}$
$T_{i,(t,a)}^{tx}(\mathbf{d})$	Transmission time of device i, $d_i = (t,a)$ in $\mathbf{d}$
$E_{i,(t,a)}^{tx}(\mathbf{d})$	Energy consumption of device i, $d_i = (t,a)$ in $\mathbf{d}$
$T_{i,(t,c)}^{exe}(\mathbf{d})$	Cloud execution time for device i in time slot t in $\mathbf{d}$
$T_{i,(t,a)}^{off}(\mathbf{d})$	Total offloading time for device i, $d_i = (t,a)$ in $\mathbf{d}$
$C_{i,(t,a)}^{c}(\mathbf{d})$	Offloading cost of device i, $d_i = (t,a)$ in $\mathbf{d}$
$C_i(\mathbf{d})$	Cost of device i in $\mathbf{d}$

formulation. In Section IV we present the algorithm, prove its convergence and characterize the structure of computed equilibria. In Section V-B we provide a bound on the approximation ratio and in Section VI we show numerical results. In Section VII we discuss related work and in Section VIII we conclude the paper.

II. System Model

We consider an edge computing system that consists of N devices, A access points (APs) and an edge cloud. We denote by $\mathcal{N} = \{1, 2, ..., N\}$ and $\mathcal{A} = \{1, 2, ..., A\}$ the set of devices and the set of APs, respectively. For ease of reference, the key notations used in the paper are summarized in Table I.

We consider that device i generates a computationally intensive task periodically every T time units, and we characterize the task 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. Similar to related works on mobile cloud computing [15], [16], [17], we define the mean number of CPU cycles as $L_i{=}D_iE[X]$, where $E[X]$ is the mean number of CPU cycles required per data bit. We assume that $E[X]$ is known from previous measurements, which is reasonable for periodically generated tasks; in fact X is often modeled by a Gamma distribution [18], [19], [20], for which unbiased estimators exist [21]. It is important to note that our model is based on the average complexity and thus it does not depend on the distribution. Furthermore, the assumption of homogeneous task periodicities is reasonable for modeling the surveillance of homogeneous physical phenomena and in manufacturing systems as discussed in [22]. We leave the case of heterogeneous periodicities to be subject of future work.

We consider that time is slotted and we denote by $\mathcal{T}{=}\{1, 2, ..., T\}$ the set of time slots. Every device can choose a time slot $t \in \mathcal{T}$ for performing the whole task $< D_i, L_i >$ and in the chosen time slot t it can decide whether to perform the task locally or to offload the computation to the cloud server through an AP $a \in \mathcal{A}$. Thus, device $i \in \mathcal{N}$ can choose one element of the discrete set $\mathfrak{D}_i = \mathcal{T} \times \{\mathcal{A} \cup \{i\}\}$, where i corresponds to local computing. We denote by $d_i \in \mathfrak{D}_i$ the decision of device 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.

For a strategy profile $\mathbf{d} \in \mathfrak{D}$ 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 $O_{(t,c)}(\mathbf{d}) = \cup_{a\in\mathcal{A}} O_{(t,a)}(\mathbf{d})$, and the total number of devices that offload in time slot t as $n_{(t,c)}(\mathbf{d}){=}\sum_{a\in\mathcal{A}} n_{(t,a)}(\mathbf{d})$. Finally, we denote by $O(\mathbf{d}) = \cup_{t\in\mathcal{T}} O_{(t,c)}(\mathbf{d})$ the set of all devices that offload in strategy profile $\mathbf{d}$. In what follows we introduce our model of sharing communication and computing resources among devices that offload their tasks.

A. Wireless resource sharing

We denote by $R_{i,a}$ the achievable uplink rate of device i towards AP (i.e., if device i was the only transmitter). We consider that $R_{i,a}$ depends on the the average channel conditions, modulation and coding scheme and the transmit power $P_{i,a}$. Thus, $R_{i,a}$ depends both on the device i and the AP a.

We denote by $\omega_{i,(t,a)}(\mathbf{d})$ the allocated uplink rate of device i at AP a in time slot t and we consider that $\omega_{i,(t,a)}(\mathbf{d})$ 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,

$$\omega_{i,(t,a)}(\mathbf{d}) = R_{i,a} \times f_a(n_{(t,a)}(\mathbf{d})). \tag{1}$$

This model is a good approximation for throughput sharing mechanisms in TDMA and OFDMA based MAC protocols [23].

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

$$T_{i,(t,a)}^{tx}(\mathbf{d}) = D_i / \omega_{i,(t,a)}(\mathbf{d}). \tag{2}$$

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 [24], [25]. The transmit power $P_{i,a}$ and the transmission time $T_{i,(t,a)}^{tx}(\mathbf{d})$ determine the energy consumption of device i for transmitting the input data of size D_i through AP a in time slot t

$$E_{i,(t,a)}^{tx}(\mathbf{d}) = P_{i,a} T_{i,(t,a)}^{tx}(\mathbf{d}). \tag{3}$$

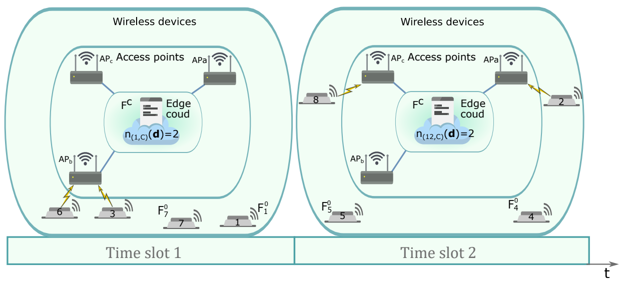

Fig. 1. An example of a mobile cloud computing system that consists of an edge cloud, $A = 3$ APs, $T = 2$ time slots and $N = 8$ devices.

B. Computing resource sharing

We denote by F^c the computational capability of the edge cloud, and we consider that the computational capability $F^c_{i,t}(\mathbf{d})$ that device i receives from the cloud in time slot t is a non-increasing function $f_i(n_{(t,c)}(\mathbf{d}))$ of the total number $n_{(t,c)}(\mathbf{d})$ of devices that offload in time slot t

$$F^c_{i,t}(\mathbf{d}) = F^c \times f_i(n_{(t,c)}(\mathbf{d})). \tag{4}$$

This model could be used for edge computing systems in which the computing resources are shared according to a time fair resource allocation policy.

Observe that the time needed for performing device i's task in the cloud depends on the chosen time slot through the number of offloaders, and can be expressed as

$$T^{exe}_{i,(t,c)}(\mathbf{d}) = L_i / F^c_{i,t}(\mathbf{d}). \tag{5}$$

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.

We can use (2) and (5) to express the task completion time in the case of offloading as the sum of the transmission time and the execution time,

$$T^{off}_{i,(t,a)}(\mathbf{d}) = T^{tx}_{i,(t,a)}(\mathbf{d}) + T^{exe}_{i,(t,c)}(\mathbf{d}). \tag{6}$$

In (6) we made the common assumption that the time needed to transmit the result of the computation from the edge cloud to the device can be neglected [7], [26], [27], [28], [29], [30], 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.

C. Local computing

Devices that perform local computing use their own computing resources only. We denote by F^0_i the computational capability of device i, and we consider that the computational capability F^0_i of device i is constant over time, and hence the time needed for device i to perform its task can be expressed as

$$T^0_i = L_i / F^0_i. \tag{7}$$

To model the corresponding energy consumption, we denote by v_i the energy consumption of device i per CPU cycle, which can be obtained through the measurement method proposed in [31]. We then express the expected energy consumption of device i for a task that requires on average L_i CPU cycles as

$$E^0_i = v_i L_i. \tag{8}$$

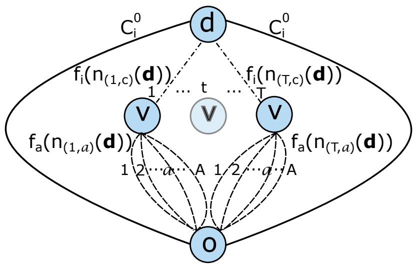

Fig. 2. Network model of the MSCOG.

Fig. 1 shows an example of a mobile edge computing system where devices can choose one out of two time slots to perform the computation. A device can offload its task to the cloud through one of three APs in its chosen time slot, if it decides to offload, e.g., in time slot 1 devices 3 and 6 offload their tasks through AP b, and devices 1 and 7 perform the computation locally.

III. Problem Formulation

In what follows we introduce the cost model of the devices and we formulate the multi-slot computation offloading problem.

A. Device Cost Model

To allow for flexibility in modeling the cost, we use the completion time γ^T_i and the energy consumption γ^E_i weights ($0 \le \gamma^T_i, \gamma^E_i \le 1$) to capture devices' preferences over the task completion time and the energy consumption, respectively. The weights can also be used to account for different units, which allows us to express the cost of device i as a linear combination of its completion time and its energy consumption, i.e.,

$$C^0_i = \gamma^T_i T^0_i + \gamma^E_i E^0_i. \tag{9}$$

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

$$C^c_{i,(t,a)}(\mathbf{d}) = \gamma^T_i T^{off}_{i,(t,a)}(\mathbf{d}) + \gamma^E_i E^{tx}_{i,(t,a)}(\mathbf{d}). \tag{10}$$

By (9) and (10) the cost of device i in strategy profile $\mathbf{d}$ is

$$C_i(\mathbf{d}) = \sum_{d_i \in \mathcal{T} \times \{i\}} \mathbf{1}_{(t,i)}(d_i) \cdot C^0_i + \sum_{d_i \in \mathcal{T} \times \mathcal{A}} \mathbf{1}_{(t,a)}(d_i) \cdot C^c_{i,(t,a)}(\mathbf{d}), \tag{11}$$

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.

Observe that in the above cost model, devices can adjust their objectives to the specific application requirements and to their current battery state by changing the values of the parameters γ^T_i and γ^E_i.

B. Multi-slot computation offloading game

We consider that devices are autonomous entities that follow their individual interests, and thus we consider that the objective of each device is to minimize its own cost (11), i.e., to find a strategy

$$d^*_i \in \arg\min_{d_i \in \mathfrak{D}_i} C_i(d_i, d_{-i}), \tag{12}$$

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. Given the autonomy of the devices, 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

$$((1,a),(1,b),(1,a),(1,4),(1,5))\xrightarrow[R_{3,b}>R_{3,a}]{3}((1,a),(1,b),(1,b),(1,4),(1,5))$$
$$\xrightarrow[2\frac{D_2}{R_{2,b}}+3\frac{L_2}{F^c}>C_2^0]{2}((1,a),(1,2),(1,b),(1,4),(1,5))\xrightarrow[C_4^0>2\frac{D_4}{R_{4,b}}+3\frac{L_4}{F^c}]{4}$$
$$((1,a),(1,2),(1,b),(1,b),(1,5))\xrightarrow[C_5^0>3\frac{D_5}{R_{5,b}}+4\frac{L_5}{F^c}]{5}((1,a),(1,2),(1,b),(1,b),(1,b))$$
$$\xrightarrow[R_{3,a}>\frac{2}{3}R_{3,b}]{3}((1,a),(1,2),(1,a),(1,b),(1,b))\xrightarrow[C_2^0>\frac{D_2}{R_{2,c}}+5\frac{L_2}{F^c}]{2}$$
$$((1,a),(1,c),(1,a),(1,b),(1,b))\xrightarrow[2\frac{D_5}{R_{5,b}}+5\frac{L_5}{F^c}>C_5^0]{5}((1,a),(1,c),(1,a),(1,b),(1,5))$$
$$\xrightarrow[\frac{D_4}{R_{4,b}}+4\frac{L_4}{F^c}>C_4^0]{4}((1,a),(1,c),(1,a),(1,4),(1,5))\xrightarrow[R_{2,b}>R_{2,c}]{2}$$
$$((1,a),(1,b),(1,a),(1,4),(1,5))$$

Fig. 3. A cyclic improvement path in a MSCOG with $N=5$ devices, $A=3$ APs, one cloud and $T=1$. Labeled arrows between strategy profiles indicate better improvement steps. A label above the arrow indicates a player that makes the improvement step, and a label below the arrow indicates the condition under which the performed action is an improvement step.

interchangeably). We refer to the game as the MSCOG. The MSCOG is a player specific network congestion game, as illustrated in Fig. 2.

Our objective is to study fundamental questions related to the existence and computability of strategy profiles from which no device would want to deviate, i.e., pure strategy Nash equilibria, defined as follows [32].

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,

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

Definition 2. 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, by definition, no device can perform a (best) improvement step in a NE. A game in which every (best) improvement path is finite is said to have the *finite (best) improvement property*.

IV. Computing Equilibria

We start with the analysis of the finiteness of improvement paths. Clearly, if a game has the finite improvement path property then it has a NE. As a first step, we show that this is not the case for the MSCOG, that is, improvement paths may be infinite, even in the case of a single time slot, i.e., $T=1$.

Lemma 1. *The MSCOG does not have the finite improvement property.*

Proof. We prove the lemma through the following example.

Example 1. *Consider a MSCOG with $\mathcal{N}=\{1,2,3,4,5\}$ players, $\mathcal{A}=\{a,b,c\}$ APs, one edge cloud and $\mathcal{T}=\{1\}$. Communication and computing resources are shared equally among the devices, i.e. $\omega_{i,(1,a)}(\boldsymbol{d})=\frac{R_{i,a}}{n_{(1,a)}(\boldsymbol{d})}$ and $F_{i,1}^c(\boldsymbol{d})=\frac{F^c}{n_{(1,c)}(\boldsymbol{d})}$, respectively. Furthermore, consider that devices aim at mininizing their completion times only, i.e., $\gamma_i^T=1$ and $\gamma_i^E=0$ for every $i \in \mathcal{N}$.*

Fig. 3 shows a cyclic improvement path starting from the strategy profile $((1,a),(1,b),(1,a),(1,4),(1,5))$, in which devices 1 and 3 offload through AP a, device 2 offloads through AP b and devices 4 and 5 perform the computation locally. The cycle shown in Fig. 3 consists of 9 improvement steps, each imposing a constraint on the system parameters. Given these constraints, an instance of the example can be formulated easily. Without loss of generality, we use the following set of parameters to illustrate the proof of the lemma: $D_i=2$ *Mb* for every $i\in\{2,3,4,5\}$, $L_2=L_3=3$ *Gcycles*, $L_4=5$ *Gcycles*, $L_5=10$ *Gcycles*, $R_{2,b}=\frac{1}{2}$ *Mb/s*, $R_{2,c}=\frac{1}{3}$ *Mb/s*, $R_{3,a}=5$ *Mb/s*, $R_{3,b}=6$ *Mb/s*, $R_{4,b}=8$ *Mb/s*, $R_{5,b}=4$ *Mb/s*, $F_2^0=\frac{3}{8}$ *GHz*, $F_4^0=\frac{16}{5}$ *GHz*, $F_5^0=\frac{50}{21}$ *GHz*, $F^c=15$ *GHz*. □

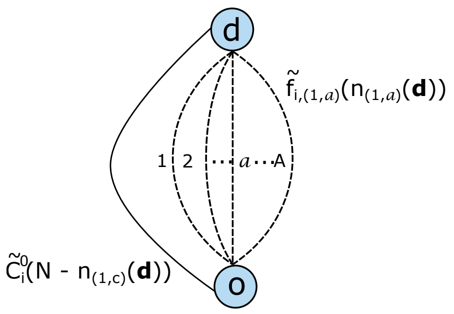
d
$\tilde{f}_{i,(1,a)}(n_{(1,a)}(\mathbf{d}))$
1
2
a
A
$\tilde{C}_i^0(N-n_{(1,c)}(\mathbf{d}))$
o

Fig. 4. Network model of the MSCOG for $T=1$.

In what follows we show that although improvement paths may be cyclic, the MSCOG possesses a NE and a NE can be computed in polynomial time.

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

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

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 [33]. 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 shown in Fig. 4. Let us define the local computation cost of player i in $\tilde{\Gamma}$ as $\tilde{C}^0{}_i(N-n_{(1,c)}(\mathbf{d}))=C_i^0-f_i(1+n_{(1,c)}(\mathbf{d}))+c$, and the cost of offloading through AP a as $\tilde{f}_{i,(1,a)}(n_{(1,a)}(\mathbf{d}))=f_{i,(1,a)}(n_{(1,a)}(\mathbf{d}))+c$, 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 $\tilde{C}^c{}_{i,(1,a)}=C_{i,(1,a)}^c-f_i(n_{(1,c)}(\mathbf{d}))$, 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. $\square$

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 $\boldsymbol{d}^*$ 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 [32]. $\square$

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

Proof. Clearly, all statements hold for $T=1$. Assume that $T>1$ and $\exists t,t' \in \mathcal{T}$ such that $n_{(t,c)}(\mathbf{d}^*) > n_{(t',c)}(\mathbf{d}^*)+1$. 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 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 $n_{(t,c)}(\mathbf{d}^*)=n_{(t',c)}(\mathbf{d}^*)$+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 $n_{(t,c)}(\mathbf{d}^*) < n_{(t',c)}(\mathbf{d}^*)$. 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 $n_{(t,c)}(\mathbf{d}^*) \geq n_{(t',c)}(\mathbf{d}^*)$. Second, assume that $n_{(t,c)}(\mathbf{d}^*) > n_{(t',c)}(\mathbf{d}^*)+1$ 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 $n_{(t,c)}(\mathbf{d}^*) \leq n_{(t',c)}(\mathbf{d}^*) + 1$ must hold. $\square$

We are now ready to formulate our main result concerning the existence of an equilibrium in the 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, based on the *MyopicBest* (MB) algorithm shown in Fig. 5. The MB algorithm adds players one at a time, and lets them play their best replies given the other players' strategies

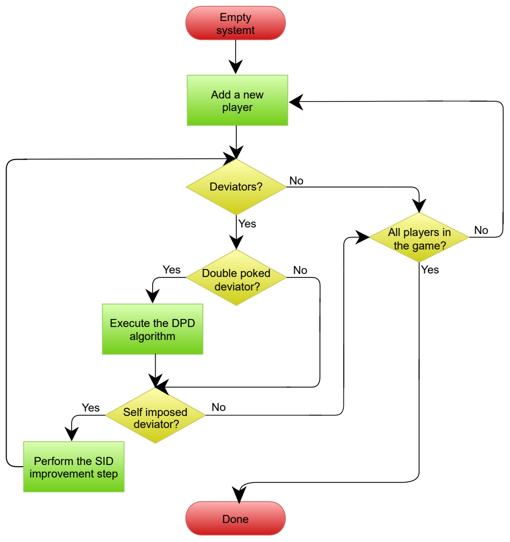

Fig. 5. Flow chart of the MB algorithm.

in a particular order. Our proof is based on an induction in the number N of players, and starts with the following result.

Theorem 4. *The MB algorithm terminates in a NE after N steps for the MSCOG with $N \leq T$ players.*

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. The MB algorithm computes such a strategy profile in N steps since each player upon it is added into the game chooses an empty time slot to perform its best reply and it has no incentive to deviate from the chosen strategy in the following induction steps. $\square$

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 $d_i^* = (t,i)$, 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 $j \in O_{(t,c)}(\mathbf{d}) \setminus O_{(t,a)}(\mathbf{d})$ may have an incentive to make an improvement step because their cloud computing cost has increased.

In order to define one of the possible best improvement paths that can be generated starting form the strategy profile $\mathbf{d} = (d_i^*, \mathbf{d}^*(N-1))$ we introduce the term deviator to denote a player that changes its strategy profile.

Definition 3. Consider two successive strategy profiles $\mathbf{d}'$ and $\mathbf{d}''$ in a best improvement path D that starts from an initial strategy profile $\mathbf{d}$. We say that the path D is a poke-new-deviator best improvement path if the following conditions hold:

1) If $\mathbf{d}'' = (d_i'', d_{-i}')$, then either $d_i'' = (t,i)$ or $d_i'' = (t,a)$ such that $n_{(t,a)}(\mathbf{d}') \geq n_{(t,a)}(\mathbf{d})$ (i.e., all deviators are either changing from offloading to local computing or from an offloading strategy to another

```
$(\mathbf{d}, t, A') = DPD(\mathbf{d}, \mathbf{d}^*(N-1), (t,a), A')$
1: /*Players that want to stop to offload*/
2: $D'_1 = \{j | d_j = (t,a), (t,j) = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d_{-j})\}$
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:     $d_i = (t,i)$
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. 6. Pseudo code of the DPD algorithm.

offloading strategy for which the number of offloaders is at least as in the initial strategy profile $\mathbf{d}$).

2) If $\mathbf{d}'' = (d''_i, d'_{-i})$, then the next best improvement step can be performed only by a player $j \in \mathcal{N} \setminus \{i\}$ such that $d''_j = d''_i$ (i.e. every next deviator has to be a player that wants to deviate from the strategy that has been chosen by the previous deviator).

Among all players that want to deviate from strategy profile $\mathbf{d} = (d^*_i, \mathbf{d}^*(N-1))$, the MB algorithm allows players $j \in O_{(t,a)}(\mathbf{d})$ to perform best improvement steps, using the *DoublePokeDeviator* (DPD) algorithm, which creates poke-new-deviator best improvement paths. The pseudo code of the DPD algorithm is shown in Fig. 6. According to the definition of a poke-new-deviator best improvement path, 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 $(t',b) \in \mathcal{T} \times \mathcal{A} \backslash \{(t,a)\}$ for which the number of offloaders in $\mathbf{d}$ is not smaller than the number of offloaders in the NE $\mathbf{d}^*(N-1)$. In each iteration, the DPD algorithm allows either one player of the first type, or one player of the second type to perform a best improvement step. In what follows we prove that the poke-new-deviator best improvement paths are finite and we provide an upper bound on the convergence of the DPD algorithm.

Proposition 1. *In a poke-new-deviator best improvement path generated by the DPD algorithm each player deviates at most once.*

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 resulting poke-new-deviator path terminates since only players that play the same strategy as the previous deviator are allowed to perform best improvement steps.

Otherwise, if player j's best improvement step is $(t',b) \in \mathcal{T} \times \mathcal{A} \backslash \{(t,a)\}$, 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 resulting poke-new-deviator path terminates because none of the players playing the same strategy as the previous deviator want to deviate. 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 poke-new-deviator best improvement path generated by the DPD algorithm, which proves the proposition. $\square$

Corollary 2. *The length of a poke-new-deviator best improvement path generated by the DPD algorithm is at most $N-1$.*

Proof. It follows from Proposition 1 that the DPD algorithm can generate a longest poke-new-deviator best improvement path upon induction step N if every player $j \in O(\mathbf{d}^*(N-1))$ performs an improvement step, which proves the result. $\square$

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.

```
$\mathbf{d}^* = MB(\mathcal{N}, \mathcal{T}, \mathcal{A}, F^c, F_i^0)$
1: Let $N \leftarrow 1$
2: **for** $N = 1 \ldots |\mathcal{N}|$ **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** $\exists j \in O_{(t',c)}(\mathbf{d}'), \exists d_j \in \mathfrak{D}_j$ s.t. $C_j(d_j, d'_{-j}) < C_j(d'_j, d'_{-j})$ **then**
11:      /*Players $j \in O_{(t,c)}(\mathbf{d}')$ 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. 7. Pseudo code of the MB 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 proved that the DPD algorithm generates a finite number of poke-new-deviator best improvement paths, each of them with a length of at most $N-1$. In the following we use this result for proving the convergence of the MB algorithm. The pseudo code of the algorithm is shown in Fig. 7.

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 2 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

```
$(\mathbf{d}, (t, a), A') = SID(\mathbf{d}, A')$
1: /*Players that offload and can decrease their offloading cost*/
2: $D_1 = \{j \in O(\mathbf{d}) | (t,a) = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d_{-j}) \in A', d_j \neq (t, a)\}$
3: /*Players that compute locally and want to start to offload*/
4: $D_2 = \{j \in L(\mathbf{d}) | (t,a) = \arg\min_{d \in \mathfrak{D}_j} C_j(d, d_{-j}) \in A'\}$
5: **if** $|D_1 \cup D_2| \neq \emptyset$ **then**
6:   /*Players that offload have priority*/
7:   **if** $D_1 \neq \emptyset$ **then**
8:     Take $i \in D_1$
9:   **else if** $D_2 \neq \emptyset$ **then**
10:    Take $i \in D_2$
11:  **end if**
12:  $d'_i = \arg\min_{d \in \mathfrak{D}_i} C_i(d, d_{-i})$
13:  Let $\mathbf{d} \leftarrow (d'_i, d_{-i})$
14:  Let $(t, a) \leftarrow d'_i$
15:  Update $A'$
16: **end if**
17: **return** $(\mathbf{d}, (t, a), A')$
```

Fig. 8. Pseudo code of the SID algorithm.

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 $j \in O_{(t',c)}(\mathbf{d}')$ 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 $j \in O_{(t',c)}(\mathbf{d}')$ 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 $n_{(t,c)}(\mathbf{d}) = n_{(t,c)}(\mathbf{d}') + 1$ 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 $j \in O_{(t',c)}(\mathbf{d}')$ that offload in the same time slot as the last deviator may want to change their offloading strategy to (t,a). Furthermore, players $j \in O_{(t',c)}(\mathbf{d}') \backslash O_{(t',b)}(\mathbf{d}')$ may want to stop to offload or to change to any offloading strategy $(t,a) \in \mathcal{T} \times \mathcal{A} \backslash \{(t',b)\}$ since their cloud computing cost increased. Let us assume that there is a player $j \in O_{(t',c)}(\mathbf{d}')$ 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 $n_{(t,c)}(\mathbf{d}) = n_{(t,c)}(\mathbf{d}') + 1$ 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 according to Proposition 2.

If case (ii) or case (iii) happens and there is no player $j \in O_{(t',c)}(\mathbf{d}')$ 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. 8. Observe that players from time slots $t \in \mathcal{T} \backslash \{t'\}$ 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 $j \in O(\mathbf{d}') \setminus O_{(t',c)}(\mathbf{d}')$ 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 $j \in O(\mathbf{d}') \setminus O_{(t',c)}(\mathbf{d}') \cup L(\mathbf{d}')$ 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 $n_{(t,c)}(\mathbf{d}) = n_{(t,c)}(\mathbf{d}') + 1$ 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 according to Proposition 2. Finally, let us consider case (iii) such that there is a player $j \in O_{(t',c)}(\mathbf{d}') \setminus O_{(t',b)}(\mathbf{d}')$ 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.

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 adding 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.*

C. Implementation considerations

Motivated by potential use cases that rely on the autonomy of devices [10], we consider that the MB algorithm can be implemented in a decentralized manner, by letting devices perform the best improvement steps one at a time.

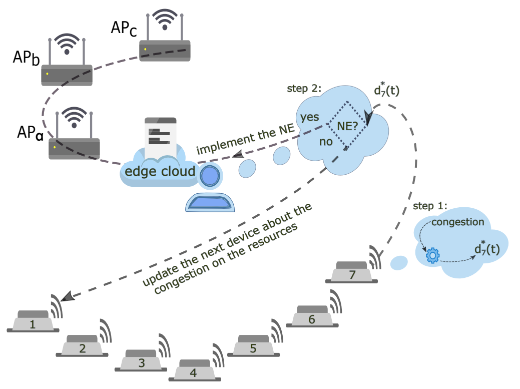

Fig. 9. Example of the interaction between the centralized entity and devices 1 and 7 in a decentralized implementation of the MB algorithm.

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 cloud resources, and the number of users sharing the APs and the cloud. In practice these information can be provided by a centralized entity that is the part of the infrastructure, e.g., the cloud, and that stores information about the mobile cloud computing system. The advantages of such a decentralized implementation compared to a centralized one are threefold. First, it supports the autonomy of the devices in MEC systems by allowing them to make their own offloading decisions based on the information provided by the centralized entity. Second, it can relieve the cloud from complex centralized management. Third, devices do not need to reveal their parameters, but only their most recent decisions.

Fig. 9 illustrates the interaction between the centralized entity located in the cloud and the devices during the process of computing a NE in a decentralized way using the MB algorithm. The centralized entity sends the information about the system (i.e. the information about the resources and the congestion on the resources in each time slot) to the devices that are allowed to update their best responses in a certain order. Given the most recent information provided by the centralized entity, each device upon its turn computes a set of best responses and sends its best offloading decision back to the centralized entity. After receiving the offloading decision of the device, the centralized entity sends the updated information about the congestion on the resources to the next device that is supposed to update its offloading decision. Observe that at some point in time, according to Theorem 3 and Theorem 6, the offloading decisions of all devices will be the same as the ones that they reported in the previous iteration and the reached state will be a NE of the MSCOG. Once such a state is reached, the centralized entity can implement the computed NE by allocating the communication and cloud resources to the devices according to their equilibrium offloading decisions.

V. NE Structure and Price of Anarchy

In what follows we characterize the structure of a NE computed by the MB algorithm and provide a bound on the price of anarchy of the game.

A. Equilibrium characterization

Recall that by Theorem 2, if a NE exists for $T \geq 1$ then the number of players is balanced across the time slots. Our next result characterizes a NE computed by the MB algorithm from the perspective of the number of offloaders per AP.

Lemma 2. *Consider a NE $\mathbf{d}^*(N-1)$ computed by the MB algorithm upon an induction step for some $T < N \leq |\mathcal{N}|$. Assume that a new player i enters the game and given the NE $\mathbf{d}^*(N-1)$ plays its best reply $d_i^*(N) = (t', a)$. If $n_{(t',a)}(\mathbf{d}^*(N-1)) > n_{(t,a)}(\mathbf{d}^*(N-1))$ for the same AP a and a time slot $t \in \mathcal{T} \setminus \{t'\}$, then the following holds*
(i) $n_{(t,c)}(\mathbf{d}^*(N-1)) = n_{(t',c)}(\mathbf{d}^*(N-1)) + 1$,
(ii) $A > 2$,
(iii) $n_{(t,a)}(\mathbf{d}^*(N-1) \geq n_{(t',a)}(\mathbf{d}^*(N-1)) - (A-2)$, and
(iv) if $n_{(t',a)}(\mathbf{d}^*(N-1)) - n_{(t,a)}(\mathbf{d}^*(N-1)) = k_a$ for $1 \leq k_a \leq A-2$ holds for every AP $a \in \mathcal{B} \subset \mathcal{A}$, $\mathcal{B} \neq \emptyset$, then $\sum_{a\in\mathcal{B}} k_a \leq A-2$ must hold.

Proof. We start with proving (i). The only reason why player i would choose to offload through AP a in time slot t', which is more congested than AP a in time slot t is that the cloud in time slot t is more congested than the cloud in time slot t', that is, $n_{(t,c)}(\mathbf{d}^*(N-1)) = n_{(t',c)}(\mathbf{d}^*(N-1)) + 1$ must hold, which proves (i).

We continue by proving (ii). It is easy to see that if $\mathcal{A} = \{a\}$, then player i's best reply has to be a strategy (t', a) for which $n_{(t',a)}(\mathbf{d}^*(N-1)) \leq n_{(t,a)}(\mathbf{d}^*(N-1))$ holds. Now, let us assume that $\mathcal{A} = \{a, b\}$. It follows from $n_{(t',a)}(\mathbf{d}^*(N-1)) > n_{(t,a)}(\mathbf{d}^*(N-1))$ and (i) that $n_{(t,b)}(\mathbf{d}^*(N-1)) > n_{(t',b)}(\mathbf{d}^*(N-1))+1$, and thus player $j \in O_{(t,b)}(\mathbf{d}^*(N-1))$ could decrease its cost by changing the strategy to (t', b). This contradicts $\mathbf{d}^*(N-1)$ being a NE, and proves (ii).

Next, we prove (iii). Assume that $n_{(t,a)}(\mathbf{d}^*(N-1)) < n_{(t',a)}(\mathbf{d}^*(N-1)) - (A-2)$ holds, which is equivalent to $n_{(t,c)}(\mathbf{d}^*(N-1)) - \sum_{b\neq a} n_{(t,b)}(\mathbf{d}^*(N-1)) < n_{(t',c)}(\mathbf{d}^*(N-1)) - \sum_{b\neq a} n_{(t',b)}(\mathbf{d}^*(N-1)) - (A-2)$. It follows from (i) that $n_{(t,c)}(\mathbf{d}^*(N-1)) = n_{(t',c)}(\mathbf{d}^*(N-1)) + 1$, and thus we have that $\sum_{b\neq a} n_{(t,b)}(\mathbf{d}^*(N-1)) > \sum_{b\neq a} n_{(t',b)}(\mathbf{d}^*(N-1)) + A - 1$ holds. Therefore, there is at least one AP $b \neq a$ such that $n_{(t,b)}(\mathbf{d}^*(N-1)) > n_{(t',b)}(\mathbf{d}^*(N-1))$, and thus player $j \in O_{(t,b)}(\mathbf{d}^*(N-1))$ could decrease its cost by changing the strategy to (t', b). This contradicts $\mathbf{d}^*(N-1)$ being a NE, and proves (iii).

Finally, we prove (iv). Assume that $\sum_{a\in\mathcal{B}} k_a > A-2$, which implies that $\sum_{a\in\mathcal{B}} \left(n_{(t',a)}(\mathbf{d}^*(N-1)) - n_{(t,a)}(\mathbf{d}^*(N-1))\right) > A-2$ holds. It follows from (i) that $n_{(t,c)}(\mathbf{d}^*(N-1)) = n_{(t',c)}(\mathbf{d}^*(N-1)) + 1$, and thus we have that $\sum_{b\in\mathcal{A}\setminus\mathcal{B}} n_{(t,b)}(\mathbf{d}^*(N-1)) > \sum_{b\in\mathcal{A}\setminus\mathcal{B}} n_{(t',b)}(\mathbf{d}^*(N-1)) + A - 1$ holds. Therefore, there is at least one AP $b \in \mathcal{A}\setminus\mathcal{B}$ such that $n_{(t,b)}(\mathbf{d}^*(N-1)) > n_{(t',b)}(\mathbf{d}^*(N-1))$, and thus player $i \in O_{(t,b)}(\mathbf{d}^*(N-1))$ could decrease its cost by changing the strategy to (t', b). This contradicts $\mathbf{d}^*(N-1)$ being a NE, and proves (iv). ☐

Observe that Theorem 2 provides an upper bound on the number of offloaders per AP for every NE, while Lemma 2 provides a lower bound on the number of offloaders per AP for a NE computed by the MB algorithm. Therefore, if $n_{(t,c)}(\mathbf{d}^*) = n_{(t',c)}(\mathbf{d}^*) + 1$, then by Theorem 2 we have that $n_{(t,a)}(\mathbf{d}^*) \leq n_{(t',a)}(\mathbf{d}^*) + 1$ and by Lemma 2 we have that $n_{(t,a)}(\mathbf{d}^*) \geq n_{(t',a)}(\mathbf{d}^*) - (A-1)$.

B. Price of Anarchy Bound

We have so far shown that a NE of the MSCOG can be computed in polynomial time and we characterized the structure of the computed NE. We now address the important question how far the system performance would be from optimal in a NE. We do so by quantifying the worst case difference between the system performance in a NE and the optimal performance using the price of anarchy (PoA). The PoA of the game is defined as the ratio of the worst case NE cost and the minimal cost, and it can be expressed as

$$PoA = \frac{\max_{\mathbf{d}^*} \sum_{i\in\mathcal{N}} C_i(\mathbf{d}^*)}{\min_{\mathbf{d}\in\mathfrak{D}} \sum_{i\in\mathcal{N}} C_i(\mathbf{d})}. \tag{13}$$

We first provide a bound of the PoA for $N \leq T$, in fact we show that a NE is optimal in this case.

Theorem 7. *A NE of the MSCOG for $N \leq T$ is the socially optimal strategy profile, i.e., $PoA = 1$ for $N \leq T$.*

Proof. We start with deriving a lower bound for an optimal solution $\bar{\mathbf{d}}$ of the MSCOG. The minimum offloading cost $\bar{C}_{i,a}$ that player i can achieve in $\bar{\mathbf{d}}$ is the cost when it offloads alone in a time slot $t \in \mathcal{T}$, i.e., $n_t(\bar{\mathbf{d}}) = 1$, through an AP a that provides maximum achievable uplink rate, i.e., $a = \arg\max_{b\in\mathcal{A}} R_{i,b}$. Consequently, we have that in the case of offloading $C_i(\bar{\mathbf{d}}) \geq \bar{C}_{i,a}$ holds. Otherwise, in the case of local computing, i.e., if $d_i = (t,i)$, we have that $C_i(\bar{\mathbf{d}}) = C_i^0$. Hence, we have that $C_i(\bar{\mathbf{d}}) \geq \min\{C_i^0, \bar{C}^c{}_{i,1}, \ldots, \bar{C}^c{}_{i,A}\}$ holds, and thus a lower bound on the optimal solution cost is given by $\sum_{i\in\mathcal{N}} \min\{C_i^0, \bar{C}^c{}_{i,1}, \ldots, \bar{C}^c{}_{i,A}\}$, i.e., $\sum_{i\in\mathcal{N}} C_i(\bar{\mathbf{d}}) \geq \sum_{i\in\mathcal{N}} \min\{C_i^0, \bar{C}^c{}_{i,1}, \ldots, \bar{C}^c{}_{i,A}\}$ and the equality holds for $N \leq T$.

We continue with characterizing the cost in a NE $\mathbf{d}^*$ for $N \leq T$. From Theorem 4 it follows that for $N \leq T$ there is at most one player per time slot, and thus in a NE $\mathbf{d}^*$ each player $i \in \mathcal{N}$ plays its best reply d_i^*, where $C_i(d_i^*, d_{-i}^*) = \min\{C_i^0, \bar{C}^c{}_{i,1}, \ldots, \bar{C}^c{}_{i,A}\}$. Therefore, $PoA = 1$ for $N \leq T$, which proves the theorem. □

In what follows we give an upper bound on the PoA of the MSCOG for $N > T$. We start with the definition of the set $\mathcal{R} = \mathcal{T} \times \{\mathcal{A} \cup \{c\} \cup \mathcal{N}\}$ of all resources in the system, and the set $O_{(t,i)}(\mathbf{d}) = \{i | d_i = (t,i)\}$ of players that use local computing resource i in time slot t. Observe that either $O_{(t,i)}(\mathbf{d}) = \emptyset$ or $O_{(t,i)}(\mathbf{d}) = \{i\}$, i.e., $n_{(t,i)}(\mathbf{d}) = |O_{(t,i)}(\mathbf{d})| \in \{0,1\}$, since players do not share their local computing resources. Furthermore, $\sum_{t\in\mathcal{T}} n_{(t,i)}(\mathbf{d}) \leq 1$ must hold since every player $i \in \mathcal{N}$ can choose only one time slot $t \in \mathcal{T}$ to perform its task. Next, we introduce the notion of player specific constants

$$w_{i,(t,a)} \triangleq \frac{D_i}{R_{i,a}}, w_{i,(t,c)} \triangleq \frac{L_i}{F^c}, w_{i,(t,i)} \triangleq \frac{L_i}{F_i^0}.$$

For a strategy profile $\mathbf{d}$ we define the total weight $w_{(t,a)}(\mathbf{d})$ associated with AP a in time slot t as $w_{(t,a)}(\mathbf{d}) \triangleq \sum_{j\in O_{(t,a)}(\mathbf{d})} w_{j,(t,a)}$, the total weight $w_{(t,c)}(\mathbf{d})$ associated with cloud c in time slot t as $w_{(t,c)}(\mathbf{d}) \triangleq \sum_{j\in O_{(t,c)}(\mathbf{d})} w_{j,(t,c)}$ and the total weight $w_{(t,i)}(\mathbf{d})$ associated with local computing resource i in time slot t as $w_{(t,i)}(\mathbf{d}) \triangleq \sum_{j\in O_{(t,i)}(\mathbf{d})} w_{j,(t,i)}$. Note that either $w_{(t,i)}(\mathbf{d}) = w_{i,(t,i)} = C_i^0$ or $w_{(t,i)}(\mathbf{d}) = 0$ must hold since $j \notin O_{(t,i)}(\mathbf{d})$ for $j \neq i$. Next, using the above notation we can express the system cost $C(\mathbf{d})$ as

$$C(\mathbf{d}) = \sum_{r\in\mathcal{R}} \sum_{i\in O_r(\mathbf{d})} n_r(\mathbf{d}) w_{i,r} = \sum_{r\in\mathcal{R}} n_r(\mathbf{d}) w_r(\mathbf{d}). \tag{14}$$

Finally, given an optimal strategy profile $\bar{\mathbf{d}}$, we can express the PoA of the MSCOG as

$$PoA = \frac{\max_{\mathbf{d}^*\in\mathfrak{D}} \sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*) w_r(\mathbf{d}^*)}{\sum_{r\in\mathcal{R}} n_r(\bar{\mathbf{d}}) w_r(\bar{\mathbf{d}})}. \tag{15}$$

Theorem 8. *Consider the MSCOG with $N > T$. Then $PoA \leq N + 1$.*

Proof. Let us denote by $\mathcal{R}_{d_i} \subset \mathcal{R}$ the set of resources that player i uses in strategy profile $\mathbf{d}$. Then, from the definition of a NE $\mathbf{d}^*$ we have the following

$$\sum_{r\in\mathcal{R}_{d_i^*}} n_r(\mathbf{d}^*) w_{i,r} \leq \sum_{r\in\mathcal{R}_{d_i^*}\cap\mathcal{R}_{\bar{d}_i}} n_r(\mathbf{d}^*) w_{i,r} + \tag{16}$$
$$\sum_{r\in\mathcal{R}_{d_i^*}\setminus\mathcal{R}_{\bar{d}_i}} \big(n_r(\mathbf{d}^*)+1\big) w_{i,r} \leq \sum_{r\in\mathcal{R}_{\bar{d}_i}} \big(n_r(\mathbf{d}^*)+1\big) w_{i,r}.$$

By summing inequality (16) over all players $i \in \mathcal{N}$ we obtain

$$\sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_{d_i^*}} n_r(\mathbf{d}^*) w_{i,r} \leq \sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_{\bar{d}_i}} \big(n_r(\mathbf{d}^*)+1\big) w_{i,r}, \tag{17}$$

and by reordering summations in (17) we obtain

$$\sum_{r\in\mathcal{R}} \sum_{i\in O_r(\mathbf{d}^*)} n_r(\mathbf{d}^*) w_{i,r} \leq \sum_{r\in\mathcal{R}} \sum_{i\in O_r(\bar{\mathbf{d}})} \big(n_r(\mathbf{d}^*) w_{i,r} + w_{i,r}\big). \tag{18}$$

By using the definition of the total weight $w_r(\mathbf{d})$ associated with resource $r \in \mathcal{R}$ in strategy profile $\mathbf{d}$, we can rewrite (18) as

$$\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*) w_r(\mathbf{d}^*) \leq \sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*) w_r(\bar{\mathbf{d}}) + \sum_{r\in\mathcal{R}} w_r(\bar{\mathbf{d}}). \tag{19}$$

Next, observe that $n_r(\mathbf{d}) \leq N$ must hold for any feasible strategy profile $\mathbf{d}$ and for every resource $r \in \mathcal{R}$, and that $|O_r(\mathbf{d})| \geq 1$ implies $n_r(\mathbf{d}) \geq 1$. Therefore, we have that $\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*) w_r(\bar{\mathbf{d}}) \leq N \sum_{r\in\mathcal{R}} n_r(\bar{\mathbf{d}}) w_r(\bar{\mathbf{d}})$ and $\sum_{r\in\mathcal{R}} w_r(\bar{\mathbf{d}}) \leq \sum_{r\in\mathcal{R}} n_r(\bar{\mathbf{d}}) w_r(\bar{\mathbf{d}})$. By using these observations in (19) we obtain the following inequality

$$\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*) w_r(\mathbf{d}^*) \leq (N+1) \sum_{r\in\mathcal{R}} n_r(\bar{\mathbf{d}}) w_r(\bar{\mathbf{d}}). \tag{20}$$

Finally, since $\sum_{r\in\mathcal{R}} n_r(\bar{\mathbf{d}}) w_r(\bar{\mathbf{d}}) > 0$ must hold, we can divide the right and the left side of inequality (20) by $\sum_{r\in\mathcal{R}} n_r(\bar{\mathbf{d}}) w_r(\bar{\mathbf{d}})$ to obtain

$$\frac{\sum_{r\in\mathcal{R}} n_r(\mathbf{d}^*) w_r(\mathbf{d}^*)}{\sum_{r\in\mathcal{R}} n_r(\bar{\mathbf{d}}) w_r(\bar{\bar{\mathbf{d}}})} \leq N + 1. \tag{21}$$

Since (21) holds for any NE of the MSCOG, it also holds for the worst case NE, and thus from (15) we have that

$$PoA \leq N + 1. \tag{22}$$

which proves the theorem. □

In what follows we investigate the tightness of the above bound on the PoA of the MSCOG.

Proposition 3. *There is an infinite family of instances of* *the MSCOG for which $PoA = N - \epsilon$, where*

$$\epsilon = \frac{(N-1)\sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}{\frac{1}{N}C_k^0 + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}. \quad (23)$$

Proof. Consider a MSCOG with $\mathcal{T} = \{t\}$, $\mathcal{N} = \{1, 2, ..., N\}$, $\mathcal{A} = \{a\}$ and cloud c. Furthermore, let us consider a strategy profile $\mathbf{d}^*$ in which $d_i^* = (t, a)$ for every player $i \in \mathcal{N}$. Given the minimum offloading cost $\bar{C}_{i,a} = \gamma_i^T(\frac{D_i}{R_{i,a}} + \frac{L_i}{F^c}) + \gamma_i^E P_{i,a}\frac{D_i}{R_{i,a}}$ that player i can achieve when it offloads alone, we can express the cost $C_i(\mathbf{d}^*)$ of player i and the system cost $C(\mathbf{d}^*)$ in strategy profile $\mathbf{d}^*$ as $C_i(\mathbf{d}^*) = N\bar{C}_{i,a}$ and $C(\mathbf{d}^*) = N\sum_{i\in\mathcal{N}}\bar{C}_{i,a}$, respectively. Next, let us assume that $C_i(\mathbf{d}^*) = N\bar{C}_{i,a} = C_i^0$ holds for every player $i \in \mathcal{N}$. It is easy to see that $\mathbf{d}^*$ is a NE of the MSCOG since there is no player $i \in \mathcal{N}$ that can decrease its cost by changing the strategy to local computing. Furthermore, it is easy to see that $\mathbf{d}^*$ is the worst case NE since all players achieve the same cost as they achieve in the case of local computing.

Now, let us assume that there is a player $k \in \mathcal{N}$ such that $C_k^0 - (N-1)\bar{C}_{k,a} \geq \sum_{i\in\mathcal{N}\setminus\{k\}}(C_i^0 - \bar{C}_{i,a})$ and $2\bar{C}_{k,a} \geq \sum_{i\in\mathcal{N}\setminus\{k\}}(C_i^0 - \bar{C}_{i,a})$ hold. From the first inequality it follows that the smallest cost saving that player k can achieve through offloading is larger than the largest cost saving that all other players can achieve together and thus in an optimal solution $\bar{\mathbf{d}}$ we have that $k \in O(\bar{\mathbf{d}})$ must hold. Furthermore, from the second inequality it follows that the smallest increase in the offloading cost of player k is higher that the largest cost saving that all other players can achieve together, and thus in an optimal solution $O(\bar{\mathbf{d}}) = \{k\}$ must hold. Therefore, the minimum system cost $C(\bar{\mathbf{d}})$ is achieved if player k is the only one offloading its computation, i.e., $C(\bar{\mathbf{d}}) = \bar{C}_{k,a} + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0$.

Next, let us find a constant ϵ for which $C(\mathbf{d}^*) = (N - \epsilon)C(\bar{\mathbf{d}})$ is satisfied, i.e,

$$N\sum_{i\in\mathcal{N}}\bar{C}_{i,a} = (N-\epsilon)(\bar{C}_{k,a} + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0). \quad (24)$$

From (24) we obtain the following

$$\epsilon = \frac{N\sum_{i\in\mathcal{N}\setminus\{k\}}(C_i^0 - \bar{C}_{i,a})}{\bar{C}_{k,a} + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}. \quad (25)$$

Since according to our first assumption $\bar{C}_{i,a} = \frac{1}{N}C_i^0$ for every $i \in \mathcal{N}$, we can express ϵ as a function of local computing costs, i.e.,

$$\epsilon = \frac{(N-1)\sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}{\frac{1}{N}C_k^0 + \sum_{i\in\mathcal{N}\setminus\{k\}} C_i^0}, \quad (26)$$

which proves the proposition. □

Since it is possible to construct an infinite number of instances of the MSCOG with $PoA = N - \epsilon$ and $\lim_{N\to\infty}\frac{N+1}{N-\epsilon} = 1$, we can formulate the following result.

Corollary 3. *The upper bound $N + 1$ on the PoA of the MSCOG is asymptotically tight.*

Proof. To show the tightness of the bound, let consider a MSCOG with $\mathcal{T} = \{1\}, \mathcal{N} = \{1, 2, 3\}, \mathcal{A} = \{a\}$ and cloud c where players aim at minimizing the completion time of their tasks only, i.e., $\gamma_i^T = 1$ and $\gamma_i^E = 0$ for every $i \in \mathcal{N}$. Furthermore, let us consider the system with the following set of parameters: $F^c = 5$ *GHz*, $L_1 = 300$ *Gcycles*, $L_2 = 3$ *Gcycles*, $L_3 = 6$ *Gcycles*, $D_1 = 80$ *Mb*, $D_2 = 2$ *Mb*, $D_3 = 4$ *Mb*, $R_{1,a} = 2$ *Mb/s*, $R_{2,a} = R_{3,a} = 5$ *Mb/s*, $F_i^0 = 1$ *GHz*, for $i \in \mathcal{N}$. Hence, we have $C_1^0 = 300$, $C_2^0 = 3$, $C_3^0 = 6$, $\bar{C}_{1,a} = 100$, $\bar{C}_{2,a} = 1$, $\bar{C}_{3,a} = 2$, $\epsilon = \frac{18}{109}$.

It is easy to verify that $\mathbf{d}^* = \big((1,a),(1,a),(1,a)\big)$ is a NE in which $C_i(\mathbf{d}^*) = 3\bar{C}_{i,a} = C_i^0$ and $C(\mathbf{d}^*) = 309$ hold. Furthermore, it is easy to see that $\bar{\mathbf{d}} = \big((1,a),(1,2),(1,3)\big)$ is an optimal solution in which $C(\bar{\mathbf{d}}) = 109$ holds. Hence, we have $\frac{C(\mathbf{d}^*)}{C(\bar{\mathbf{d}})} = N - \epsilon = \frac{309}{109} \approx 2.84$. □

VI. Numerical Results

In the following we show simulation results to evaluate the cost performance and the computational efficiency of the MB algorithm. Similar to [34], [35] 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 [36]. For simplicity we assign a bandwidth of $B_a = 5$ *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 [37]. Given the noise power P_n we calculate the PHY rate $R_{i,a}$ as $R_{i,a} = B_a\log_2(1 + P_{i,a}d_{i,a}^{-\alpha}/P_n)$. 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. According to the specification reported in [38], the clock rate achievable for NVIDIA Tegra 2 is up to 1 *GHz*, and thus we consider that the computational capability F_i^0 of device i is uniformly distributed on $[0.5, 1]$ *GHz*. Based on the approximate relative computational parameters for devices and clouds reported in [39], we consider that the computation capability of the cloud is $F^c = 100$ *GHz* and we assume 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 [31]. 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 four algorithms as a basis for comparison for the proposed *MB* algorithm. In the first algorithm devices 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 $T = 1$. In the fourth algorithm, the offloading decisions of the devices are made according to a socially optimal strategy profile $\mathbf{d}^*$. We define the

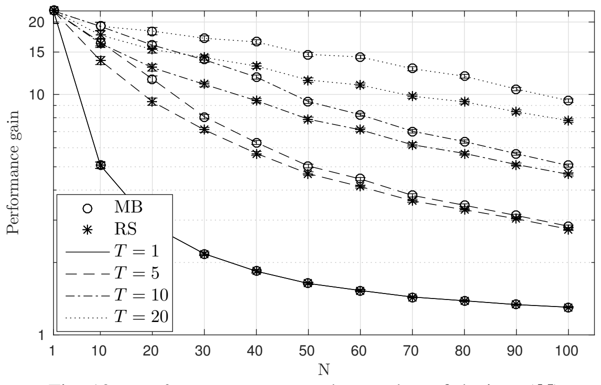

Fig. 10. *Performance gain* vs. the number of devices (N).

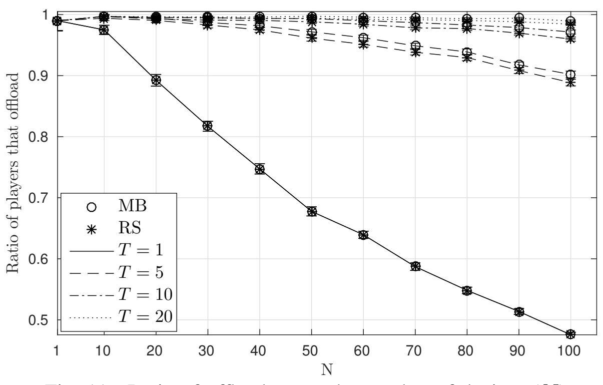

Fig. 11. Ratio of offloaders vs. the number of devices (N).

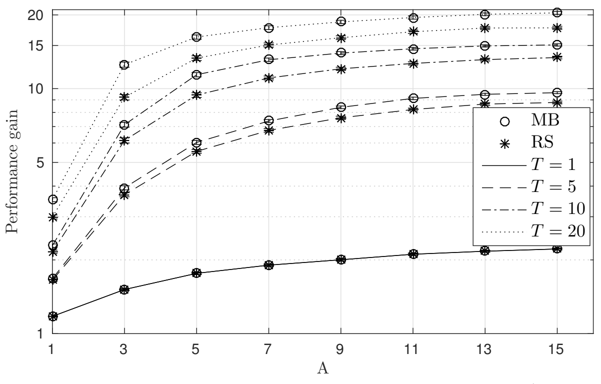

Fig. 12. *Performance gain* vs. the number of APs (A).

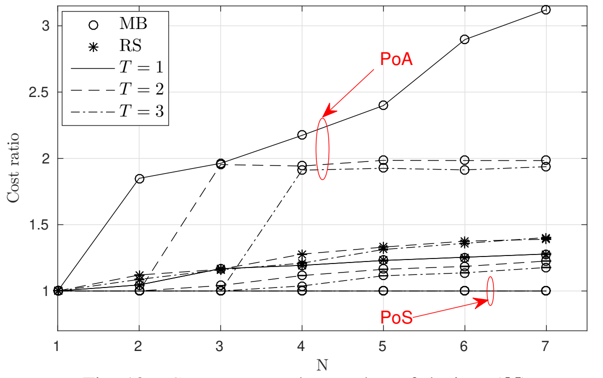

Fig. 13. *Cost ratio* vs. the number of devices (N).

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 and we define the *cost-approximation ratio* of an algorithm as the ratio between the system cost reached by the algorithm and the system cost reached when all devices choose offloading decisions according to the socially optimal strategy profile. Unfortunately, computing the socially optimal strategy profile was computationally feasible only for small problem instances due to the combinatorial nature of the corresponding system cost minimization problem, which is a 0-1 non-linear program. The results shown are the averages of 100 simulations, together with 95% confidence intervals.

A. Performance gain vs. number of devices

We start with evaluating the *performance gain* as a function of the number N of devices for $A = 4$ APs. Fig. 10 shows the *performance gain* of the MB algorithm, the RS algorithm and the deterministic worst case $T = 1$. The results show that the *performance gain* decreases with the number of devices for all algorithms. 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. 11 shows the ratio of players that offload for the same set of parameters as in Fig. 10. 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. 10. 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. 10.

B. Performance gain vs number of APs

In order to evaluate the *performance gain* as a function of the number A of APs, we consider a system that consists of $N = 50$ devices. Fig. 12 shows the *performance gain* of the MB algorithm, the RS algorithm and the deterministic worst case $T = 1$. We observe that the *performance gain* achieved by the algorithms increases 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

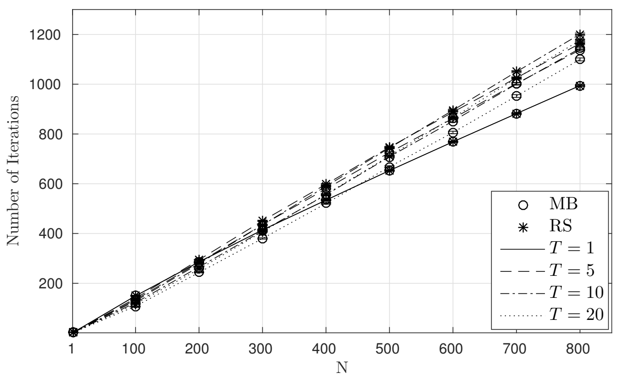

Fig. 14. Number of iterations vs. the number of devices (N).

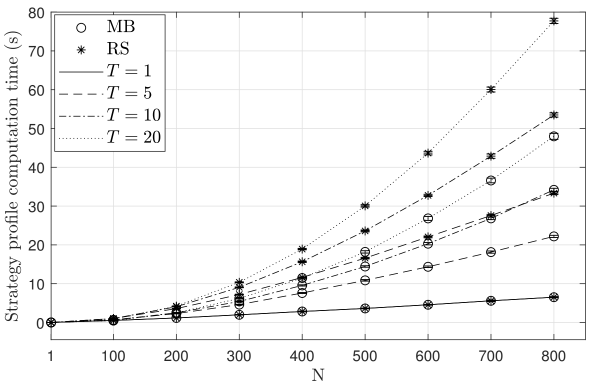

Fig. 15. Strategy profile computing time vs. the number of devices (N).

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. Cost-approximation ratio vs. number of devices

Fig. 13 shows the average *cost-approximation ratio* for the MB and the RS algorithms, and the computed *price of anarchy* (PoA) and the *price of stability* (PoS) as a function of the number N of devices. The results are shown for three values of the number T of time slots (i.e. $T \in \{1, 2, 3\}$) for a system with $A = 4$ APs. We show the results only up to 7 devices, because the computation of the socially optimal strategy profile was infeasible for larger problem instances. The results show that the MB and the RS algorithms have the same *cost-approximation ratio* for all values of N when $T = 1$, which is because the two algorithms are equivalent in this case, and thus they compute the same equilibrium strategy profiles. On the contrary, for $T > 1$ and for $N > 1$, a strategy profile computed by the RS algorithm is not an equilibrium, and the *cost-approximation ratio* of the RS algorithm is higher than that of the MB algorithm. We can also observe that the gap between the algorithms increases with T, which is because the imbalance in the average congestion per time slot increases with T in the case of the RS algorithm due to the random time slot selection. The results also show that for $T = 1$ the computed PoA increases linearly with N with a slope lower than 1 and that for $T > 1$ the computed PoA remains close to 2. Therefore, the results indicate that the worst case scenario for which the PoA of the game is asymptomatically close to N (c.f. Theorem 8 and Proposition 3 from Section V-B) is not likely to happen. Finally, we observe that the computed PoS is equal to 1 in all cases, which suggests that the MB algorithm is able to compute a socially optimal equilibrium of offloading decisions.

D. Computational Complexity

In order to assess the computational efficiency of the algorithms 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 strategy profile. Fig. 14 and Fig. 15 show the average number of iterations and the corresponding average strategy profile computation time for the MB and the RS algorithms, respectively. The results are shown as a function of the number N of devices in a system with $A = 4$ APs for four different values of the number T of time slots (i.e., $T \in \{1, 5, 10, 20\}$). 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. 14 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. These results coincide with the results in Fig. 15 that show that the MB algorithm is at least as good as the RS algorithm (i.e. the two algorithms consume the same amount of time for computing a strategy profile when $T = 1$ and the MB algorithm performs better than the RS algorithm for $T > 1$). 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. 14 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.

Finally, Fig. 15 shows the wall clock computing time of the MB and RS algorithms as a function of the number of devices N. The results show that the MB algorithm is faster than the RS algorithm in all cases, and the computing times are slightly super-linear due to the increasing time to compute a best response. Overall, we conclude that the proposed MB algorithm can compute efficient equilibrium allocations for periodic task offloading at low computational complexity.

VII. Related Work

The scheduling of periodic tasks received significant attention for real-time systems [40], [41], but without considering communications. Similarly, the scheduling of communication resources has been considered without considering computation [42]. Most works that considered both communication and computation focused on a single device case [43], [31], [8], [44], [45], and thus they do not consider the sharing of communication and computing resources.

Related to our work are recent works on energy efficient computation offloading for multiple mobile users [46], [47], [48]. [46] proposed a genetic algorithm for maximizing the throughput in a partitioning problem for mobile data stream applications, while [47] considered a two-tiered cloud infrastructure with user mobility in a location-time workflow framework and proposed a heuristic for minimizing the cost of users. [48] 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 [49], [50], [51], [7], [52], [53], [54], [55]. Compared to [49], we characterize the structure of the computed equilibrium, prove the bound on the price of anarchy and show an example of a better reply cycle. [50] considered a two-stage game, where first each mobile user chooses the parts of its task to offload with the objective to minimize the energy consumption and the task completion time, and then the cloud allocates computational resources to the offloaded parts with the objective to maximize its profit. [51] considered a three-tier cloud architecture with stochastic task arrivals, and provided a distributed algorithm for the computing a mixed strategy equilibrium. [53] 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. [7] 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. [52] 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. [55] considered multiple wireless links, equal bandwidth sharing and a non-elastic cloud, and provided a polynomial time algorithm for computing equilibria. Our work differs significantly from these works, as we consider the problem of scheduling periodic tasks over time and across heterogeneous communication resources and to the best of our knowledge, this is the first work that bridges the gap between early works on scheduling [41] and recent works on the resource allocation in computation offloading systems [7], [55].

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 [33], thus the proposed algorithm and our proof of existence advance the state of the art in the study of equilibria in network congestion games.

VIII. Conclusion

We provided a game theoretical analysis of computation offloading in a mobile edge computing system where devices generate tasks periodically. We proved the existence of pure strategy Nash equilibria, characterized their structure and based on our constructive proof we proposed a decentralized algorithm for computing an equilibrium allocation of offloading decisions. We proved that the proposed algorithm has a bounded approximation ratio and quadratic worst case complexity. Our numerical results show that the performance in an equilibrium computed by the proposed algorithm is significantly better than in a strategy profile in which offloading decisions are not coordinated over time. 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] L. Xiao and Z. Wang, "Internet of things: A new application for intelligent traffic monitoring system," *Journal of networks*, vol. 6, no. 6, p. 887, 2011.

[3] M. Ayazoglu, B. Li, C. Dicle, M. Sznaier, and O. I. Camps, "Dynamic subspace-based coordinated multicamera tracking," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2462–2469.

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

[5] S. R. Group *et al.*, "The leading cloud providers continue to run away with the market," Tech. rep, Tech. Rep., 2017.

[6] A. Reznik, R. Arora, M. Cannon, L. Cominardi, W. Featherstone, R. Frazao, F. Giust, S. Kekki, A. Li, D. Sabella *et al.*, "Developing software for multi-access edge computing," *ETSI, White Paper*, no. 20, 2017.

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *Proc. of IEEE PDS*, pp. 974–983, 2015.

[8] 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.

[9] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 207–220, 2018.

[10] "IoT ONE: Use Cases," https://www.iotone.com/usecases.

[11] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[12] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[13] Q. He, G. Dán, and V. Fodor, "Minimizing age of correlated information for wireless camera networks," in *INFOCOM WKSHPS*, 2018, pp. 547–552.

[14] I. Kadota, A. Sinha, and E. Modiano, "Optimizing age of information in wireless networks with throughput constraints," in *IEEE INFOCOM*, 2018, pp. 1844–1852.

[15] Z. Sheng, C. Mahapatra, V. C. Leung, M. Chen, and P. K. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," vol. 6, no. 1, pp. 114–126, 2018.

[16] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

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

[18] 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.

[19] ——, "Pace: A new approach to dynamic voltage scaling," *IEEE Transactions on Computers*, no. 7, pp. 856–869, 2004.

[20] W. Yuan and K. Nahrstedt, "Energy-efficient CPU scheduling for multimedia applications," *ACM Trans. on Computer Systems (TOCS)*, vol. 24, no. 3, pp. 292–331, 2006.

[21] A.-B. Shaibu and H. A. Muttlak, "Estimating the parameters of the normal, exponential and gamma distributions using median and extreme ranked set samples," *Statistica*, vol. 64, no. 1, pp. 75–98, 2004.

[22] T. Hoßfeld, F. Metzger, and P. E. Heegaard, "Traffic modeling for aggregated periodic iot data," in *Proc. of IEEE ICIN (Workshop)*, 2018, pp. 1–8.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] S. Jošilo and G. Dán, "Joint allocation of computing and wireless resources to autonomous devices in mobile edge computing," in *Proc. of ACM MECOMM*, 2018, pp. 13–18.

[29] ——, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. of IEEE INFOCOM*, 2019, pp. 2467–2475.

[30] S. Jošilo and G. Dán, "Joint management of wireless and computing resources for computation offloading in mobile edge clouds," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[31] 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.

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

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

[34] E. Balevi and R. D. Gitlin, "Optimizing the number of fog nodes for cloud-fog-thing networks," *IEEE Access*, vol. 6, pp. 11 173–11 183, 2018.

[35] S. Sigg, P. Jakimovski, and M. Beigl, "Calculation of functions on the rf-channel for iot," in *2012 3rd IEEE International Conference on the Internet of Things*. IEEE, 2012, pp. 107–113.

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

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] S. Jošilo and G. Dán, "Decentralized scheduling for offloading of periodic tasks in mobile edge computing," in *Proc of IFIP NETWORKING*, 2018.

[50] 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.

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

[52] 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.

[53] 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.

[54] 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.

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

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 in 2013-2014. Her research interests are design and analysis of decentralized algorithms for exploiting resources available at the network edge using game theoretical tools.

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