Tobias Mahn, and Anja Klein, ”Energy-Efficient Application-Aware Mobile Edge Computing with Multiple Access Points,” in *Proc. of the 31st IEEE International Symposium on Personal on Personal, Indoor and Mobile Radio Communications (PIMRC) 2020*, September 2020.

Energy-Efficient Application-Aware Mobile Edge Computing with Multiple Access Points

Tobias Mahn, Anja Klein
Communications Engineering Lab, TU Darmstadt, Germany, {t.mahn, a.klein}@nt.tu-darmstadt.de

***Abstract*—The offloading decision making of multiple mobile units in a mobile edge computing (MEC) scenario with multiple access points, each with an attached cloudlet server, is investigated. This scenario bears a joint problem to be solved: the assignment of mobile units to access points, the allocation of the shared communication and computation resources and the offloading decision of each mobile unit. Additionally, the offloading decision is influenced by the availability of the required software for the computation of the task of the mobile unit that may not be available at the cloudlet and has to be downloaded before starting the computation. The proposed offloading decision making and resource allocation problem is formulated as a global optimization problem. To handle higher numbers of mobile units in the network and also achieve a greatly faster convergence, a game theoretic algorithm for the individual offloading decisions is introduced. In numerical simulations, the game based algorithm is shown to deliver results close to the optimum solution, while requiring only a small number of iterations until convergence to a Nash equilibrium.**

I. Introduction

With an ever increasing number of wireless sensors and battery powered mobile devices, offloading computation from local processors to other computation facilities in the network or on the internet is necessary to enable new machine to machine (M2M) services, allow computation intensive mobile applications, make new applications with low latency requirements possible or simply extend the battery life. Recent forecasts expect a slight growth in the number of smartphones and a growth in billions of M2M devices in the next years until 2022 [1]. Providing computation services at the edge of the networks, e.g. at cellular base stations or WiFi access points, close to the sensors and devices shall help to reduce both the computation delay as well as the backhaul traffic to distant cloud servers. Those computation services in Mobile Edge Computing (MEC) can be integrated into small servers at the network edge, called cloudlets or edge clouds [2], [3].

The right software is required to be installed at the cloudlet to be able to provide the computing capabilities for the corresponding application of a mobile unit (MU). Caching shall predict popular contents like files or services and store them at the edge of the network. This reduces the latency for the user to access a file or a service while simultaneously reducing the traffic in the backhaul network [4]. Nevertheless, it is not always possible to meet the needs of every MU when the storage capacity of the cloudlet is limited and not all content can be provided simultaneously. In MEC, the availability of the required software at the cloudlet can influence the decision of a MU whether offloading of its computation task is beneficial.

In the beginning, MEC has been investigated for small networks with one or multiple MUs and a single access point (AP) or cellular base station, e.g. in [5], [6]. The authors of [5] propose a network consisting of a single MU and a single AP. The computation task of the MU is arbitrarily splittable. Algorithms to split the task and compute the result in an energy-efficient way are evaluated by optimizing the power consumption of the local processor of the MU and the transmission power used to send the offloaded fraction from the MU to the AP. In [6], not a single MU but multiple MUs connect to a single AP. The proposed approach also optimizes the split of their task to be offloaded as well as the transmission power of the MUs.

MEC scenarios with more than one computation location have been studied in hierarchical offloading scenarios in [7]–[9]. MUs are still connected to a single AP and share the same radio access channel, but besides offloading the computation tasks only to the cloudlet, there is an additional cloud server available for computation. The cloud server offers more computation resources than the cloudlet, but the transmission of the task introduces a higher latency than offloading to the cloudlet. Compared to the previous scenarios, MUs have to decide where to offload the task. This introduces the challenge of multiple MUs coordinating their offloading decisions based on the sharing of communication and computation resources. The authors of [7]–[9] introduce game theoretic algorithms to MEC in order to model autonomous offloading decisions of the MUs based on the available resources. These algorithms can often converge to an equilibrium point much faster than approaches based on optimization and can also handle scenarios with a much higher number of MUs. The objective of all proposed algorithms is different. In [7], the authors are interested to minimize the maximum computation time and transmission latency of one task in a stream of arriving tasks. For a single task per MU, the authors of [8] propose an approach that minimizes the combined energy consumption of all MUs and the maximum computation time of all computation locations in the scenario. In [9], also a single task per MU is considered and the objective is to minimize the energy consumed by all MUs for the computation of their task while keeping a maximum computation time limit.

The previously described scenarios do not consider that a MU can reach multiple APs and cloudlets can be in a dense

scenario. In this case, the MU has to decide whether offloading is beneficial at all and to which AP to connect to have the most benefit from offloading. In this paper, the offloading decisions of MUs are investigated with respect to the joint consideration of communication and computation resources available in the network. The proposed network consists of multiple APs with attached cloudlets to which the MUs can connect. Joint optimization in MEC scenarios with multiple APs and MUs is also investigated in [10] and [11], but both consider multiple APs that forward offloaded tasks to only a single cloudlet server. In [10], the authors formulate an optimization problem that minimizes the combined energy consumption and maximum computation time. They use a game theoretic algorithm to find a faster converging approximation to the global optimization problem. The communication resources in [10] are modeled in a simplified way by a maximum channel rate that is equally divided by the number of offloading users. The authors of [11] also propose a joint minimization problem, but they aim to optimize the energy consumption together with a more abstract monetary cost function. Similar to [10], the global optimization problem is reformulated into a game theoretic approach. In comparison to [10], the communication model in [11] is more precise by considering the Shannon channel capacity. Furthermore, in [11] the authors formulate the resource allocation problem by Lagrange multipliers and solve the KKT-conditions which leads to an optimal resource allocation. A similar formulation of the optimal resource allocation problem using Lagrange multipliers has been proposed in [12] for a hierarchical offloading scenario.

In comparison to [10] and [11], in our scenario, each AP is equipped with a cloudlet server. Furthermore, the proposed scenario considers the availability of required software to compute an offloaded task at the cloudlet. If the software is not available at the chosen offloading location, a download from a cloud server is initiated. Regarding the formulation of a global optimization problem, a drawback of the described joint objective functions in [10] and [11] is the combination of two parameters of different units and possibly even different scales. The choice of the weighting factor between the two parameters can lead to vastly different results of the optimization problem. Therefore, we avoid such a formulation by proposing an energy minimization problem that is bound by a constraint for the maximum allowed computation time. A game theoretic algorithm is introduced to find a fast converging approximation of the global optimization problem. This algorithm includes an initialization strategy for the offloading problem, the separation of the offloading decisions from the resource allocation and an optimal allocation of the communication and computation resources.

In Section II, the scenario and most relevant concepts are introduced. They build a basis for the formulation of the optimization problem in Section III-A and the proposed game theoretic approach in Section III-B. The performance of the proposed algorithm is evaluated numerically in Section IV.

II. System Model

A. Scenario

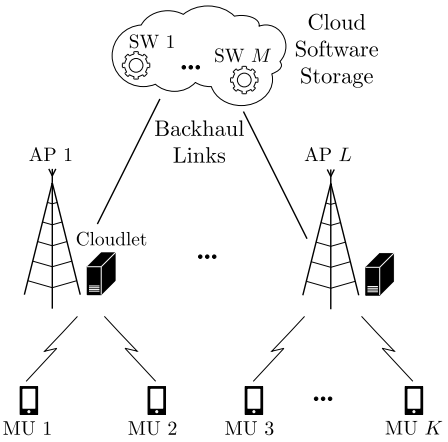

Fig. 1. Computation offloading scenario with K MUs and L APs

A scenario with multiple access points (APs), each with a cloudlet for edge computation, is considered. These APs are installed at different locations of the considered area. Mobile units (MUs) with low mobility can connect via shared radio channels to the APs. Possible examples for such an arrangement and use case are lecture halls, conference centers or stadiums.

As shown in Figure 1, the considered scenario consists of K MUs and L APs. Each MU $k \in \{1, \ldots, K\}$ has a non-splittable task of size s_k^{task} which is measured in bits. The task can either be computed locally or can be offloaded to one AP $l \in \{1, \ldots, L\}$. A task belongs to one of multiple types of applications, e.g. music, photo, video, calculation or gaming, and those require also different softwares to compute the result of the task. As the authors of [13] have shown, the types of applications can require a significantly different number of central processing unit (CPU) cycles for the computation. Therefore, a complexity factor c_k is introduced to measure the number of CPU cycles required to calculate one bit of the task of MU k. The decision of MU k for local computation or offloading to AP l is stored in a binary offloading decision vector $\mathbf{x}_k = [x_k^{\text{MU}}, x_{k,1}^{\text{AP}}, \ldots, x_{k,L}^{\text{AP}}]$, of which exactly one element is 1 and all other elements are 0.

The software for the computation is assumed to be available at the MU, but the cloudlet may not have it installed and, in this case, a download of the software from a remote cloud server to the AP is required to enable the computation of the offloaded task. A binary variable $a_{l,m}$ stores the information whether software m, with $m \in \{1, \ldots, M\}$, is installed at AP l. The concept of the software download can also be extended to microservices where not a single software packet but a combination of microservice modules is required for the computation of the task [14]. Missing microservices have to be fetched from a cloud server before the computation of the offloaded task can start.

B. Local Computation

The CPU of MU k has a processing frequency f_k^{MU}. If MU k decides for the local computation of the task, the processing time can be calculated as

$$T_k^{\text{MU}} = \frac{c_k s_k^{\text{task}}}{f_k^{\text{MU}}} \tag{1}$$

and by multiplication with the calculation power p_k^{calc} of the CPU, the required computation energy is

$$E_k^{\text{MU}} = p_k^{\text{calc}} \cdot \frac{c_k s_k^{\text{task}}}{f_k^{\text{MU}}}. \tag{2}$$

C. Offloading Computation to an AP

Besides local computation, each MU k can decide for offloading the task to one of the L APs. Each AP is assumed to have a separate transmission band with a total bandwidth $b_l^{\max}$ at AP l. The radio access channel from MU k to AP l is assumed to use an orthogonal frequency-division multiple access (OFDMA) transmission scheme. There is a total bandwidth $b_l^{\max}$ available to transmit to AP l. Multiple MUs offloading tasks to the same AP have to share the available bandwidth. The corresponding allocated bandwidth for MU k offloading to AP l is expressed by $b_{k,l}$ and the sum of bandwidths assigned to the offloading MUs at AP l cannot exceed the total bandwidth $b_l^{\max}$, i.e. $\sum_{k=1}^{K} b_{k,l} \leq b_l^{\max}$. With a transmission power $p_{k,l}^{\text{trans}}$, the uplink channel gain $|h_{k,l}|^2$ and the white Gaussian noise power σ^2, the resulting transmission rate $r_{k,l}^{\text{MU}}$ from MU k to AP l can be expressed by the Shannon channel capacity

$$r_{k,l}^{\text{AP}} = b_{k,l} \log_2 \left(1 + \frac{p_{k,l}^{\text{trans}} |h_{k,l}|^2}{\sigma^2} \right). \tag{3}$$

For simplicity of the model, $|h_{k,l}|^2$ and σ^2 are assumed to be known at the AP. Using the transmission rate, the duration of the offloading from MU k to AP k can be defined as the upload time

$$T_{k,l}^{\text{up}} = \frac{s_k^{\text{task}}}{r_{k,l}^{\text{AP}}}. \tag{4}$$

Each AP is equipped with a cloudlet server that is able to compute tasks with a total computation frequency $f_l^{\text{AP max}}$. If more than one MU decides to offload to AP l, each MU will only receive a fraction of the total computation frequency f_l^{maxAP}. The assigned frequency of MU k offloading its task to AP l is denoted by $f_{k,l}^{\text{AP}}$. The sum of all assigned computation frequencies is bound by the total computation frequency, i.e. $\sum_{k=1}^{K} f_{k,l}^{\text{AP}} \leq f_l^{\text{maxAP}}$. The computation time of the task of MU k task at AP l can be determined by

$$T_{k,l}^{\text{comp}} = \frac{c_k s_k^{\text{task}}}{f_{k,l}^{\text{AP}}}. \tag{5}$$

If the software for the computation is not available, a download of software m with size s_m^{app} in bits from a central cloud server can be started simultaneously to the upload of the task. With a backhaul link transmission rate r_l^{down} from the cloud server to AP l, the download time of software m can be calculated as

$$T_{m,l}^{\text{down}} = (1 - a_{m,l}) \cdot \frac{s_m^{\text{app}}}{r_l^{\text{down}}}. \tag{6}$$

In a scenario with available software for calculation, the processing time for the offloading of the task of MU k to AP l can be written as the sum of (4) and (5). To include the new possibility that the downloading time of the software exceeds the upload time of the task, the maximum of (4) and (6) is taken, which leads to a new expression for the total offloading time in dependence of the shared resources:

$$\begin{aligned} T_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) &= \max\left\{T_{k,l}^{\text{up}}, T_{m,l}^{\text{down}}\right\} + T_{k,l}^{\text{comp}} \\ &= T_{k,l}^{\text{up}} + \max\left\{0, T_{m,l}^{\text{down}} - T_{k,l}^{\text{up}}\right\} + T_{k,l}^{\text{comp}}. \end{aligned} \tag{7}$$

A static power p_k^{static} is introduced for each MU k to model the power of all other components of the device while communicating or being idle during the remote computation. Furthermore, each MU k has a transmission power $p_{k,l}^{\text{trans}}$. The required energy for offloading the task of MU k to AP l can then be modeled by

$$\begin{aligned} E_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) &= \left(p_{k,l}^{\text{trans}} + p_k^{\text{static}}\right) \cdot T_{k,l}^{\text{up}} \\ &+ p_k^{\text{static}} \cdot \left(\max\left\{0, T_{m,l}^{\text{down}} - T_{k,l}^{\text{up}}\right\} + T_{k,l}^{\text{comp}}\right). \end{aligned} \tag{8}$$

III. Problem Formulation

A. Optimization Problem

The resources allocated to MU k can be indicated by vector to $\mathbf{b}_k = [b_{k,1}, \ldots, b_{k,L}]$ for the bandwidths of the radio channels and to $\mathbf{f}_k = [f_{k,1}^{\text{AP}}, \ldots, f_{k,L}^{\text{AP}}]$ for the CPU frequencies at the APs. If MU k decides for offloading its task to AP l, the entries $b_{k,l}$ and $f_{k,l}^{\text{AP}}$ are non-zero while all others are 0. The required computation energy for MU k can be written in dependence of the allocated communication and computation resources and the binary offloading decision vector as

$$E_k\left(\boldsymbol{x}_k, \boldsymbol{b}_k, \boldsymbol{f}_k\right) = x_k^{\text{MU}} \cdot E_k^{\text{MU}} + \sum_{l=1}^{L} x_{k,l}^{\text{AP}} \cdot E_{k,l}^{\text{AP}}\left(\boldsymbol{b}_k, \boldsymbol{f}_k\right). \tag{9}$$

The sum of the computation energies of all K MUs shall be minimized as the objective of an optimization problem which

can be formulated as

$$\underset{\substack{\boldsymbol{x}_k,\boldsymbol{b}_k,\boldsymbol{f}_k,\\ \forall k\in\mathcal{K}}}{\arg\min} \quad \sum_{k=1}^{K} E_k\left(\boldsymbol{x}_k,\boldsymbol{b}_k,\boldsymbol{f}_k\right), \tag{10}$$

$$\text{s.t.} \quad x_{k,l}^{\text{AP}}\cdot T_{k,l}^{\text{AP}}\left(b_{k,l},f_{k,l}^{\text{AP}}\right)\le T_k^{\text{MU}}, \quad \forall k\,\&\,\forall l, \tag{10a}$$

$$\sum_{k=1}^{K} b_{k,l}\le b_l^{\max}, \qquad \forall l, \tag{10b}$$

$$\sum_{k=1}^{K} f_{k,l}^{\text{AP}}\le f_l^{\text{AP max}}, \qquad \forall l, \tag{10c}$$

$$b_{k,l},f_{k,l}^{\text{AP}}\ge 0, \qquad \forall k\,\&\,\forall l, \tag{10d}$$

$$x_k^{\text{MU}},x_{k,l}^{\text{AP}}\in\{0,1\}, \qquad \forall k\,\&\,\forall l, \tag{10e}$$

$$x_k^{\text{MU}}+\sum_{l=1}^{L} x_{k,l}^{\text{AP}}=1, \qquad \forall k. \tag{10f}$$

Constraint (10a) ensures an upper time limit for the offloaded task. An offloaded task should be computed at least as fast as the local computation on the CPU of the MU itself to guarantee a positive quality of experience when deciding for offloading. The upper bounds of the shared resources in constraints (10b) and (10c) have already been introduced in Section II. The nonnegativity constraint (10d) prevents the possibility of assigning negative resources to MUs. The last two constraints (10e) and (10f) handle the binary nature of the decision variables and guarantee that each MU has to take exactly one decision where to compute its task.

In this form, problem (10) is a mixed-integer non-linear program (MINLP). Although a reformulation of this optimization problem and linearization of the objective function is possible to make it compatible with MINLP-solvers like BARON, the number of constraints increases rapidly with the numbers K of MUs and L of APs. Therefore, a game theoretic approach is investigated in the next subsection.

B. Game Theoretic Approach

The energy minimization problem can be reformulated and approximated by an iterative algorithm, where the offloading decisions of the MUs are modeled as a potential game. Each MU tries to minimize its own energy for the computation of its task, but each individual offloading decision has an influence on the allocation of the communication and computation resources for all MUs. If all MUs simultaneously decide for offloading of the task to the same AP l, there is a high possibility that the shared resources are not sufficient so that no MU can fulfill the maximum offloading time constraint.

Game Porperties: We define a general strategic form game $\mathcal{G}$ similar to [15] as an ordered triplet

$$\mathcal{G}=\left(\mathcal{K},\{\mathcal{S}_k\}_{k\in\mathcal{K}},\{u_k\}_{k\in\mathcal{K}}\right) \tag{11}$$

with a set $\mathcal{K}$ of players, a set $\{\mathcal{S}_k\}$ of strategies for each player and a utility function $\{u_k\}$ for each player. The set $\mathcal{K}=\{1,\dots,K\}$ of players is consists of the MUs of the considered scenario. Each player has to choose between local computation and offloading to one of the L APs which can be modeled as strategy set

$$\mathcal{S}_k=\left\{ s_k=(x_k^{\text{MU}},x_{k,1}^{\text{AP}},\dots,x_{k,L}^{\text{AP}}) \;\middle|\; x_k^{\text{MU}},x_{k,l}^{\text{AP}}\in\{0,1\};\; x_k^{\text{MU}}+\sum_{l=1}^{L} x_{k,l}^{\text{AP}}=1 \right\}. \tag{12}$$

The played strategies of all MUs can be written by the corresponding strategy profile $\boldsymbol{s}=(s_k,s_{-k})$. In this formulation, the played strategies are separated into the strategy of MU k by $s_k\in\mathcal{S}_k$ and the played strategies of all other MUs by $(s_1,\dots,s_{k-1},s_{k+1},\dots,s_K)\in\mathcal{S}_{-k}=\prod_{j\neq k}\mathcal{S}_j$. Finally, the utility function for MU k can be defined by the required energy for the computation of its task from (9) as

$$u_k(\boldsymbol{s})=\begin{cases}\infty, & \text{for } T_{k,l}^{\text{AP}}(\boldsymbol{s})>T_k^{\text{MU}},\\ E_k\left(s_k,\boldsymbol{b}_k,\boldsymbol{f}_k\right), & \text{else.}\end{cases} \tag{13}$$

A Nash equilibrium (NE) is a strategy profile $\boldsymbol{s}^*$ where a player k cannot decrease its utility function when deviating from its strategy s_k^*, i.e. $u_k(s_k^*,s_{-k})\le u_k(s_k,s_{-k}),\forall s_k\in\mathcal{S}_k$. A NE does not always exist for a general strategic form game, but it is proven in [16] that a potential game has at least one.

Another important property found in [16] is the finite improvement property (FIP). The player k is said to follow an improvement path, where the utility function in the next iteration $i+1$ is lower than the previous one i, i.e. $u_k(s_k[i+1],s_{-k})\le u_k(s_k[i],s_{-k})$. If all players have a finite improvement path, the game possesses the FIP. It also guarantees that a player k will reach its lowest possible utility function in a finite number of iterations while all other players do not change their strategies. For the proposed model, it means that each MU k can reach its optimal offloading decision when the offloading decisions of all other MUs are unchanged.

Resource Allocation Strategy: The previously described game and the strategies of the players apply to the offloading decisions of the MUs, but not to the shared resources. When a strategy profile $\boldsymbol{s}(i)$ of the offloading decisions is available in the current iteration i, this optimization variable can be excluded from optimization problem (10). Furthermore, it is assumed that the maximum computation time constraint is neglected for the resource allocation as it will be checked during the offloading decision of the MU. The resulting resource optimization problem can be formulated as

$$\underset{\boldsymbol{b}_k,\boldsymbol{f}_k,\forall k\in\mathcal{K}}{\arg\min} \quad \sum_{k=1}^{K} E_k\left(\boldsymbol{b}_k,\boldsymbol{f}_k\right), \tag{14}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} b_{k,l}\le b_l^{\max}, \qquad \forall l, \tag{14a}$$

$$\sum_{k=1}^{K} f_{k,l}^{\text{AP}}\le f_l^{\text{AP max}}, \qquad \forall l, \tag{14b}$$

$$b_{k,l},f_{k,l}^{\text{AP}}\ge 0, \qquad \forall k\,\&\,\forall l. \tag{14c}$$

It is shown in [12] that the shared communication and computation resources are independent after excluding the offloading decisions. Therefore, it is also possible to formulate (14) using Lagrangian multipliers and solve the Karush-Kuhn-Tucker-conditions for an optimal allocation strategy. The results are the fraction of the total bandwidth $b_l^{\max}$ for MU k offloading its task to AP l as

$$b_{k,l}^* = b_l^{\max} \cdot \frac{\sqrt{\frac{x_{k,l}^{\text{AP}}(p_{k,l}^{\text{trans}}+p_k^{\text{static}})s_k^{\text{task}}}{\log_2\left(1+\frac{p_{k,l}^{\text{trans}}|h_{k,l}|^2}{\sigma^2}\right)}}}{\sum_{k=1}^{K}\sqrt{\frac{x_{k,l}^{\text{AP}}(p_{k,l}^{\text{trans}}+p_k^{\text{static}})s_k^{\text{task}}}{\log_2\left(1+\frac{p_{k,l}^{\text{trans}}|h_{k,l}|^2}{\sigma^2}\right)}}} \tag{15}$$

and the fraction of the total available processing frequency f_l^{maxAP} at the AP l reserved for the computation of the task of MU k

$$f_{k,l}^{\text{AP}\,*} = f_l^{\text{maxAP}} \cdot \frac{\sqrt{x_{k,l}^{\text{AP}} p_k^{\text{static}} c_k s_k^{\text{task}}}}{\sum_{k=1}^{K}\sqrt{x_{k,l}^{\text{AP}} p_k^{\text{static}} c_k s_k^{\text{task}}}}. \tag{16}$$

These fractions of the shared resources are used in the following two algorithms to allocate the shared resources according to the available strategy profiles.

Initialization: The game theoretical algorithm can be initialized by any strategy profile $\boldsymbol{s}$. Nevertheless, a well chosen starting point can improve the convergence speed and possibly also the results of the algorithm. An intuitive approach is the assignment of each MU to the nearest AP by Euclidean distance. If the portion of the shared resources allocated to MU k is not enough to fulfill its maximum offloading time constraint (10a), the MU will decide for local computation.

Algorithm 1 Initialization (Offloading to the Nearest AP)

All MUs decide to offload to their closest AP
(by Euclidic distance)
Calculate $\boldsymbol{b}_k^*, \boldsymbol{f}_k^* \forall k$ for this strategy profile $\boldsymbol{s}$
if (10a) is not fulfilled for some MUs **then**
  Set corresponding offloading decisions
  of these MUs to local computation
end if

Game Theoretic Algorithm: After an initial strategy profile is available, the iterative algorithm can be started. It is summarized as pseudo-code in Algorithm 2. An iteration counter i is used to store the number of changes in the offloading decision before the NE is reached. The communication and computation resources from (15) and (16) are calculated in each step for all possible strategies $\{\mathcal{S}_k\}$ of MU k who is the active player. The algorithm makes use of the finite strategy set of each MU k and the finite improvement path until this MU reaches its minimum utility function. After one MU reaches the minimum, the next MU $k+1$ has to decide whether it can lower its utility function. The algorithm terminates if all MU cannot decrease their utility function and therefore, do not change their offloading decision anymore. These final offloading decisions are stored in the strategy profile $\mathbf{s}^*$.

Algorithm 2 Iterative Game Theoretic Algorithm

Set NE = False and $i = 0$
while NE == False **do**
  $k = 1$; reset = 0;
  while reset = 0 && $k \leq K$ **do**
    AP calculates $\{\boldsymbol{b}_k^*, \boldsymbol{f}_k^*\}$ for $(s_k', s_{-k}[i]), \forall s_k' \in \mathcal{S}_k$
    MU k checks, if (10a) is fulfilled, otherwise sets corresponding utility to $u_k(s_k', s_{-k}[i]) = \infty$
    if $u_k(\mathbf{s}[i]) > u_k(s_k', s_{-k}[i]), s_k' \in \mathcal{S}_k$ **then**
      Set $\mathbf{s}[i+1] = (s_k', s_{-k}[i])$; $i = i+1$; reset = 1;
    else if $k == K$ **then**
      Set NE = True;
    else
      $k = k+1$; reset = 1;
    end if
  end while
end while
return NE $\mathbf{s}^*$ of game G and corresponding resource allocation $\{\mathbf{r}_k^*\}$

IV. Numerical Results

For the numerical simulations, the described scenario is modeled by 4 APs placed in the corners of a $200\,\text{m} \times 200\,\text{m}$ square and a varying number of MUs placed inside the square with a minimum distance of 10 m to an AP. Each MU is assumed to have one task to compute. The task belongs to one of three different types of applications, which can be characterized as follows:

- Type 1 (audio, photo, etc.): small to medium size, low complexity, offloading or local computation is highly dependent on the network quality and the offloading decisions of the other MUs
- Type 2 (video processing): high task size, high complexity, offloading is always preferred
- Type 3 (computation tasks, e.g. MATLAB): small size, high complexity, offloading is only preferred if the software is installed at the AP

The assumed simulation parameters for all types of tasks are chosen to be close to measurements from [13] and [17] and are summarized in Table I. Each MU has a computation frequency

TABLE I
Simulation parameters of different task types

	Type 1	Type 2	Type 3
Task Size s^{task}	10 MB	100 MB	1 MB
Task Complexity c	200	1000	2000
Software Size s^{app}	200 MB	2000 MB	1000 MB

of $f_k^{\text{MU}} = 1\,\text{GHz}$, while the cloudlets are more powerful and offer a total computation frequency of $f_l^{\text{maxAP}} = 8\,\text{GHz}$. The powers associated with each MU k are local calculation power $p_k^{\text{calc}} = 1\,\text{W}$, transmission power $p_{k,l}^{\text{trans}} = 200\,\text{mW}$ and static power $p_{k,l}^{\text{static}} = 100\,\text{mW}$.

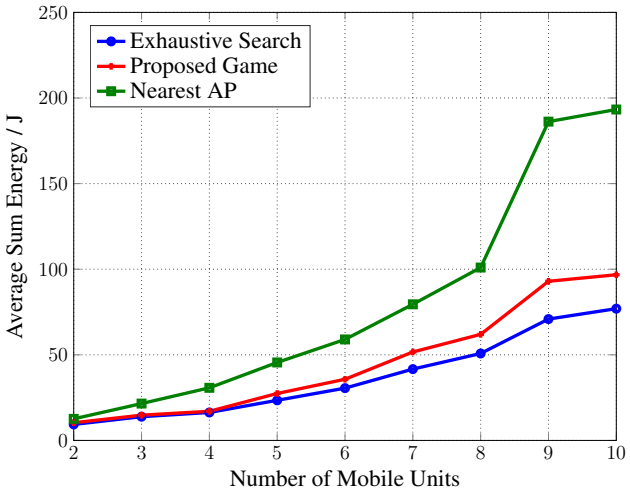

Fig. 2. Average sum energy of all MUs for different numbers K of MUs

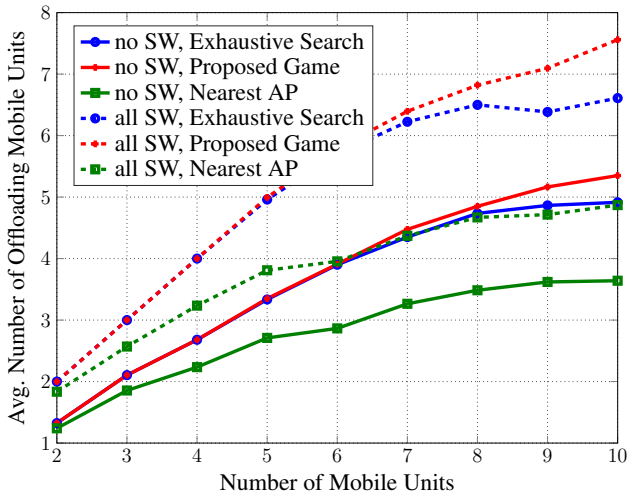

Fig. 3. Avergage number of offloading MUs for different numbers K of MUs

Each AP l is assumed to have a separate radio channel with a maximum available bandwidth of $b_l^{\max} = 10\,\text{MHz}$ and a white Gaussian noise power of $\sigma^2 = 10^{-13}$ W. The channel gain between MU k and AP l is given by $|h_{k,l}|^2 = 1/d_{k,l}^3$ with the Euclidean distance $d_{k,l}$. Each AP l is connected with a backhaul link transmission rate of $r_l^{\text{cloud}} = 1\,\text{Gbit/s}$ to the central cloud server for software downloads.

In the following results, 200 Monte Carlo runs with randomly placed users and random task type are performed per data point. The optimal solution is obtained by exhaustive search over all permutations of the possible offloading decisions which scales by the number of possible offloading decisions per MU to the power of the number K of MUs, i.e. $(L+1)^K$. For 10 MUs there are already about $5^{10} \approx 9.76$ million different possible combinations of offloading decisions.

The first result in Figure 2 shows the sum energy of all MUs on average over all Monte Carlo runs. It is assumed that no software was available at the APs. In the plot, we compare three possible methods to calculate the offloading decisions and corresponding sum energy of the MUs: offloading to the nearest AP which is equal to the initialization strategy, the proposed game theoretic algorithm and the optimal solution obtained by exhaustive search. The proposed game is close to the optimal solution and already 80.7% better for 4 MUs and 99.7% better for 10 MUs than the assignment to the nearest AP.

An important aspect of this paper is the new model of the software availability. This raises the question whether the results show any difference in comparison to offloading models without considering it. Figure 3 shows the number of offloading MUs when increasing the total number K of MUs in the scenario. While the dotted lines show the number of offloading MUs in a scenario with all required software available at the APs, the solid lines show the difference with the newly introduced possible absence of software. Compared to the case of available software, the number of MUs deciding for offloading in a system with required software download is considerably lower. It shows that our model of a possible software download can improve the offloading decisions of the MUs. Interestingly, the proposed game seems to slightly overestimate the optimal number of offloading MUs compared to the optimal solution. Therefore, the assigned fractions of the shared resources per offloading MU on average are slightly lower than the fractions computed by the optimal solution. This is one of the reasons for the gap between the proposed game and the optimal solution in Figure 2.

The game theoretic algorithm is also tested for the number of iterations until it converges to a NE. For the simulation result in Figure 4, each AP is assumed to randomly have one of the three required software installed. The solid lines represent the average number of iterations the algorithm requires for different for both homogenous task types and mixed task types at the MUs. Especially for MUs with only type 2 (video processing) or only type 3 (computation tasks) tasks, the number of iterations is higher as there is more competition about the communication and computation resources. The gray area in Figure 4 represents the number of iterations during 200 Monte Carlo simulations with different placements of the MUs and its upper bound is the worst case number of iterations that occurred during the simulations. Even for 25 MUs, the worst case number of iterations was only 20. As the lower bound is at 0 iterations for all numbers K of MUs, it shows that our proposed initialization method is already producing a reasonable starting point to support a fast convergence. Furthermore, the result in Figure 4 shows that the game theoretic algorithm can easily be scaled up for a much higher number K of MUs compared to the exhaustive search solution.

V. Conclusion

Considering the software availability, the offloading problem is modeled as an energy minimization problem. To overcome the rapidly increasing complexity of this MINLP for-

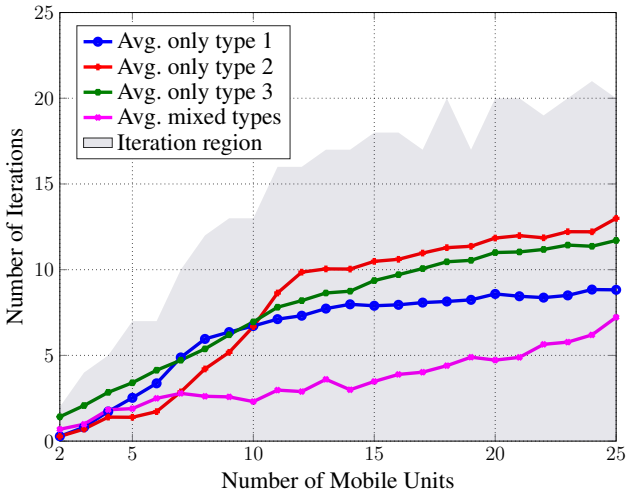

Fig. 4. Number of iterations of the game theoretic algorithm for different numbers K of MUs

mulation, a game theoretic algorithm, an initialization strategy and a resource allocation method are proposed. The numerical results show a significant improvement of the proposed game theoretic algorithm over the simpler strategy of assigning MUs to the closest AP. Furthermore, the game has a considerably lower computational complexity and can handle a much higher number of MUs in the network than the optimal solution by exhaustive search. The software availability produces a noticeable difference in the willingness of the MUs to offload their computation.

ACKNOWLEDGEMENT

This work has been performed in the context of the DFG Collaborative Research Center (CRC) 1053 MAKI - subprojects B3 and C7. This work has been supported by DAAD with funds from the German Federal Ministry of Education and Research (BMBF).

REFERENCES

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 20172022," Tech. Rep. C11-738429-01, 2019.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing - a key technology towards 5G," *ETSI White Paper*, vol. 11, 2015.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[4] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching, and computing in wireless systems: A survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 7–38, 2017.

[5] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[6] H. Q. Le, H. Al-Shatri, and A. Klein, "Efficient resource allocation in mobile-edge computation offloading: Completion time minimization," in *Proc. of the IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2513–2517.

[7] V. Cardellini, V. D. N. Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. L. Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, vol. 157, no. 2, pp. 421–449, 2016.

[8] M.-H. Chen, M. Dong, and B. Liang, "Multi-user mobile cloud offloading game with computing access point," in *Proc. of 5th IEEE International Conference on Cloud Networking (Cloudnet)*, 2016, pp. 64–69.

[9] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[10] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of the IEEE INFOCOM 2017 - Conference on Computer Communications*, 2017, pp. 1–9.

[11] Q. Li, J. Zhao, and Y. Gong, "Cooperative computation offloading and resource allocation for mobile edge computing," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2019, pp. 1–6.

[12] T. Mahn, D. Becker, H. Al-Shatri, and A. Klein, "A distributed algorithm for multi-stage computation offloading," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. IEEE, 2018, pp. 1–6.

[13] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing." *Proc. of the 2nd USENIX Conf. Hot Topics on Cloud Computing*, pp. 1–4, 2010.

[14] S. Newman, *Building microservices: designing fine-grained systems*. O'Reilly Media, Inc., 2015.

[15] S. Lasaulce and H. Tembine, *Game theory and learning for wireless networks: fundamentals and applications*. Academic Press, 2011.

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

[17] A. Carroll, G. Heiser *et al.*, "An analysis of power consumption in a smartphone." in *USENIX annual technical conference*, vol. 14. Boston, MA, 2010, pp. 21–21.