Tobias Mahn, and Anja Klein, "A Global Orchestration Matching Framework for Energy-Efficient Multi-Access Edge Computing," in *Proc. of the 10th IEEE International Conference on Cloud Networking (IEEE CloudNet 2021)*, November 2021.

A Global Orchestration Matching Framework for Energy-Efficient Multi-Access Edge Computing

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

***Abstract*—Multi-access edge computing (MEC) enables mobile units (MUs) to offload computation tasks to edge servers nearby. This translates in energy savings for the MUs, but creates a joint problem of offloading decision making and allocation of the shared communication and computation resources. In a MEC scenario with multiple MUs, multiple access points and multiple cloudlets the complexity of this joint problem grows rapidly with the number of entities in the network. The complexity increases even further when some MUs have a higher incentive to offload tasks due to a low battery level and are willing to pay in exchange for more resources. Our proposed energy-minimization approach with a flexible maximum offloading time constraint is based on matching theory. A global orchestrator (GO) collects all the system state information and coordinates the offloading preferences of the MUs. A MU can lower the maximum time constraint by a payment. The GO allocates the shared communication and computation resources accordingly to satisfy the time constraint. The computation load of the algorithm at each MU is reduced to a minimum as each MU only has to take a simple offloading decision based on its task properties and payment willingness. In numerical simulations, the proposed matching approach and flexible resource allocation scheme is tested for fast and reliable convergence, even in large networks with hundreds of MUs. Furthermore, the matching algorithm, tested with different resource allocation strategies, shows a significant improvement in terms of energy-efficiency over the considered reference schemes.**

I. Introduction

The Cisco Annual Internet Report 2020 forecasts a rapid deployment of billions of machine to machine (M2M) devices until 2023 [1]. These devices will enable new services for companies, cities and home automation, but most of them are strictly limited in their processing power and battery capacity. Multi-access Edge Computing (MEC) is expected to be the key technology to deliver computation resources at cloudlet servers in the vicinity of the M2M devices [2].

Mobile computing devices like smartphones, tablets and laptops can also profit from external computation resources and save their limited battery energy by offloading computation tasks to such cloudlets. Specifically, the authors of the studies [3] and [4] about the usage of smartphones have shown that users are already behaving very cautious about saving energy or recharging their device at battery levels of around $30\,\%$. Therefore, providing the possibility to flexibly allocate more communication and computation resources to mobile units (MUs) with low battery levels than to MUs with higher battery levels, enables these devices to extend the usage of computation extensive tasks like video editing or augmented reality applications.

Distributed algorithms for MEC scenarios with multiple MUs, multiple access points (APs) and multiple cloudlets have already been studied in [5]–[9]. Compared to smaller scenarios with single APs for communication or single cloudlets for computation, the MUs have to choose between multiple offloading options. The most beneficial assignment of MUs to APs and cloudlets has to be found while simultaneously considering that the limited communication and computation resources are shared between all offloading MUs. While the authors of [5], [6] and [7] propose algorithms based on game theory for the offloading decisions and solve the resource allocation in an optimal way by solving a Lagrange dual version of the problem, the authors of [8] and [9] consider matching theory as the basis for their algorithms.

In [5], the sum of computation times of all MUs shall be minimized. The problem is formulated as an exact potential game. The formulated problem minimizes the sum of all task computation times of the MUs. A joint objective of energy minimization and a monetary payment is proposed in [6]. While this model also considers the operation costs of the cloudlet, the weighting factors of the joint objective function can significantly affect the outcome of the algorithm. In [7], we propose an energy minimization game where the offloading decision is influenced by a fixed maximum offloading time constraint and the availability of the required software for the computation at the cloudlet. Furthermore, we define different types of tasks with individual characteristics.

Although game theory and, specifically, potential games offer many possibilities to model algorithms for MEC scenarios, the related field of matching theory enables broader possibilities. Most importantly, it can provide the ability to model different objectives for the MUs and the mobile network operator (MNO). The authors of [8] propose an algorithm based on the student project allocation (SPA) problem. The SPA problem is fitting for the offloading decision making problem as it is able to match two entities, each with individual preferences. In case of an offloading scenario, these entities are the MUs and the APs with cloudlets. The proposed algorithm is fast converging and reliable, but it shares the resources equally by a defined quota of offloading MUs. The algorithm with the equal resource allocation is best suited for scenarios in which all MUs have comparable task characteristics. The more different the tasks are, the less effective is this allocation strategy. In [9], the authors propose a global orchestrator (GO) that is performing the matching of the MUs to the APs. The

formulated problem shall jointly minimize the sum of energies of all MUs and the maximum offloading time in the network. The minimization of the maximum offloading time is a useful choice if the results of all tasks have to be considered as a cumulative result. In a scenario with independent MUs and tasks, an individual maximum time constraint for every MU is a better choice.

In this work, a scenario with multiple MUs, APs and cloudlets is assumed. Each MU has a task that belongs to one of multiple types of tasks. The maximum time for offloading the task of a MU is bound by a maximum time constraint. While in [7], the maximum offloading time is fixed to be less than the local computation time, a more flexible time constraint is introduced in this paper. By having a more expensive subscription and offering an additional payment, the MUs can receive a bigger fraction of the shared communication and computation resources. A MU with a lower battery level is assumed to have a higher willingness for a payment. A SPA problem based matching algorithm is proposed to solve the offloading decision making problem and a GO is performing the matching. The three main contributions of this work are: 1. Introducing a matching framework with a GO that coordinates the communication and computation resources and chooses the most suitable AP and cloudlet for a MU to offload its task. 2. Simplifying the offloading decision of a MU to deciding whether offloading is beneficial or not and whether the MU can profit from receiving more resources in exchange for a payment. 3. Proposing a flexible resource allocation strategy based on the maximum offloading time and payment preference of a MU.

The paper is structured as follows: The scenario and most relevant concepts are introduced in Section II. In Section III, the resource management, optimization problem and matching framework are formulated. In Section IV, the proposed algorithm is evaluated numerically.

II. System Model

A. Scenario

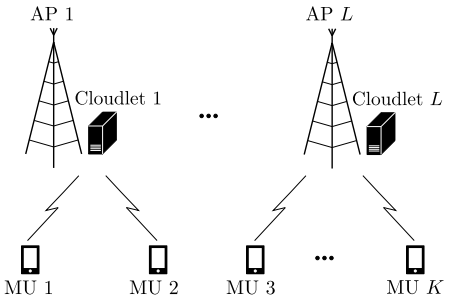

Fig. 1. Computation offloading scenario with K MUs and L APs

In the considered MEC scenario, multiple MUs are willing to connect to remote servers to offload computation tasks. These remote servers are modeled by cloudlet servers. Each of the cloudlet servers is positioned at one of the APs. The connection between MUs and APs is established via shared radio access channels. The scenario is shown in Figure 1 with K MUs and L APs. The MUs are modeled by the set $\mathcal{K} = \{1, \dots, K\}$ and the APs by the set $\mathcal{L} = \{1, \dots, L\}$.

B. Mobile Units and Local Computation

Every MU $k \in \mathcal{K}$ has a non-splittable task to be computed. The task can be described by its size s_k^{task}, measured in bits, and its computational complexity factor c_k, measured in central processing unit (CPU) cycles per bit. Although this complexity factor is only an approximation, it enables the comparability of different task types, e.g. video processing, calculations or Augmented Reality applications. The task types are chosen similar to [7]. The complexity of different task classes on CPUs has been studied in [10].

Because the task is non-splittable, each MU k has to take a binary decision whether the task should be computed locally on its own CPU, i.e. $x_k^{\text{MU}} = 1$, or the task should be offloaded to one of the cloudlet servers, i.e. $x_k^{\text{AP}} = 1$.

The frequency f_k^{MU} and processing power p_k^{calc} of the CPU of MU k are known. If MU k decides for local computation, the local processing time can be estimated by

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

The corresponding local processing energy is then given by

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

C. Remote Computation at an Access Point

A MU $k \in \mathcal{K}$ can be in the vicinity of multiple APs from $\mathcal{L}$, but it can be only connected to one of them simultaneously. Each AP l is assumed to have its own frequency band with a total bandwidth $b_l^{\max}$. When MU k is transmitting its task to AP l it uses an orthogonal frequency-division multiple access (OFDMA) transmission scheme. If multiple MUs decide to offload their tasks to AP l, each MU k only receives a fraction $b_{k,l}$ of the total bandwidth $b_l^{\max}$. All fractional bandwidths assigned to MUs can reach at most the total bandwidth $b_l^{\max}$, i.e. $\sum_{k=1}^{K} b_{k,l} \leq b_l^{\max}$. With estimates of the uplink channel gain $|h_{k,l}|^2$ and the white Gaussian noise power σ^2 for the radio access channel between MU k and AP l, an expression for the Shannon channel capacity can be found by

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

Knowing $r_{k,l}^{\text{AP}}$, the upload time from MU k to AP l is defined by

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

Similar to the shared bandwidth, the computation resources at the cloudlet of AP l are shared by all MUs deciding to offload to this AP. MU k receives a fraction $f_{k,l}^{\text{AP}}$ of the total computation frequency f_l^{maxAP}. The total amount of allocated

computation resources to all offloading MUs cannot exceed the total computation frequency, i.e. $\sum_{k=1}^{K} f_{k,l}^{\text{AP}} \leq f_l^{\text{maxAP}}$. The time for the processing of the task of MUs k at AP l can then be expressed by

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

The total offloading time is then defined by the sum of (4) and (5) as

$$T_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) = T_{k,l}^{\text{up}} + T_{k,l}^{\text{comp}}. \tag{6}$$

To calculate the offloading energy spent during the offloading process by the MU, two powers are introduced. The static power p_k^{static} represents the baseline power of all components of MU k spent both in idle state and during transmitting state. While transmitting its task, the radio hardware of MU k has an additional transmit power $p_{k,l}^{\text{trans}}$. With these two powers and (6), the energy of MU k for offloading its task to AP l is defined by

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

As the offloading time in (6) and the offloading energy in (7) are dependent on the shared resources, they will be denoted in the following by $T_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right)$ and $E_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right)$, respectively.

The time for the transmission of the computation result back to the MU is omitted in our model, as the result can be assumed to be much smaller than the original task in many applications [11].

D. Global Orchestration

In order to be able to calculate its best offloading decision, each MU $k \in \mathcal{K}$ has to communicate with each AP $l \in \mathcal{L}$ about its possible fraction of the shared resources. A GO is introduced to collect all the required information centrally, manage the resource allocation and reduce the signaling overhead. The GO takes the leading position and can be one of the APs or a central entity in the backhaul of the network, controlled by the MNO. It has the two important functions:

1) Collect all the necessary system information, i.e. characteristics of the MUs and their tasks, information about the radio access channels, information about the available computation resources.
2) Compute the best suitable AP l for offloading the task of MU k and propose this option to the MU.

It is assumed that a MU k is not interested to communicate with all L APs to negotiate about the possible shares of the resources. Furthermore, it is assumed that the GO is acknowledged as a trustworthy instance by all MUs that is taking the best possible decision for it. The GO stores the best offloading option for MU k in an offloading decision vector $\mathbf{x}_k = [x_k^{\text{MU}}, x_{k,1}^{\text{AP}}, \ldots, x_{k,L}^{\text{AP}}]$ and this vector has exactly one non-zero element.

E. Flexible Maximum Offloading Time Constraint

In [7], the maximum offloading time is limited by a fixed constraint in the proposed optimization problem and game theoretic algorithm. This constraint ensures that offloading the task of MU k to an AP l is at least as fast as local computation, i.e.

$$T_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) \leq T_k^{\text{MU}}. \tag{8}$$

In this paper, a more flexible time constraint is introduced. Each MU k is assumed to have a subscription with the MNO that is running the L APs. Furthermore, it is assumed that a MU can spend additional monetary units in form of tokens to achieve a lower maximum computation time $T_k^{\max}$. By paying tokens, the MU can react to different battery levels and request the allocation of more resources for offloading. In general, P levels of subscription plans and an additional payment between 0 and Q tokens can be considered. The subscription plan of MU k is denoted by $v_k^{\text{sub}} \in \left\{v_1^{\text{sub}}, \ldots, v_P^{\text{sub}}\right\}$ and the payment by $v_k^{\text{pay}} \in \left\{v_0^{\text{pay}}, \ldots, v_Q^{\text{pay}}\right\}$. With the knowledge of both parameters, the GO can calculate the value of MU k as $v_k = v_k^{\text{sub}} \cdot v_k^{\text{pay}}$. This value influences the time factor

$$t_k = \begin{pmatrix} t_{1,0} & t_{1,1} & \cdots & t_{1,Q} \\ \vdots & \vdots & \ddots & \vdots \\ t_{P,1} & t_{P,2} & \cdots & t_{P,Q} \end{pmatrix}$$

that is responsible for a shorter or longer maximum offloading time constraint.

For more simplicity of the following explanations, we assume two levels of subscription and a maximum payment of 2 tokens. If MU k has a free or cheaper subscription, we set variable v_k^{sub} to 0. If it has a more expensive subscription with a shorter guaranteed maximum offloading time, α_k^{sub} is set to 1. We define a binary vector $\boldsymbol{v}_{\boldsymbol{k}}^{\text{pay}}$ that represents the willingness of MU k to pay extra monetary units. The vector has three binary elements $v_{k,0}^{\text{pay}}$, $v_{k,1}^{\text{pay}}$ and $v_{k,2}^{\text{pay}}$ representing that MU k spends 0, 1 or 2 extra tokens, respectively. With these two assumptions, a flexible maximum offloading time can be formulated as

$$\begin{aligned} T_k^{\max} &= \left(1 - v_k^{\text{sub}}\right) \cdot \left(v_{k,0}^{\text{pay}} \cdot t_0 + v_{k,1}^{\text{pay}} \cdot t_2 + v_{k,2}^{\text{pay}} \cdot t_3\right) \cdot T_k^{\text{MU}} \\ &+ v_k^{\text{sub}} \cdot \left(v_{k,0}^{\text{pay}} \cdot t_4 + v_{k,1}^{\text{pay}} \cdot t_5 + v_{k,2}^{\text{pay}} \cdot t_6\right) \cdot T_k^{\text{MU}}, \end{aligned} \tag{9}$$

where $t_1, \ldots, t_6$ are scaling factors for the local computation time T_k^{MU}. The adapted maximum offloading time constraint is

$$T_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) \leq T_k^{\max}. \tag{10}$$

III. Problem Formulation

A. Optimization Problem

The share of the communication resources allocated to MU k can be indicated by vector $\mathbf{b}_k = [b_{k,1}, \ldots, b_{k,L}]$. Similarly, the vector $\mathbf{f}_k = [f_{k,1}^{\text{AP}}, \ldots, f_{k,L}^{\text{AP}}]$ represents the share of the computation resources allocated to MU k. 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. Using the offloading decision stored in the offloading decision vector $\mathbf{x}_k$ and the energies for local computation defined in (2) and for offloading to AP l defined in (7), the GO can calculate the energy the computation of the task of MU k in dependence of the allocated resources and the decision by

$$E_k\left(\boldsymbol{x}_k, \boldsymbol{b}_k, \boldsymbol{f}_k\right) = x_k^{\text{MU}} \cdot E_k^{\text{MU}} + \sum_{l=1}^{L} x_{k,l}^{\text{AP}} \cdot E_{k,l}^{\text{AP}}\left(\boldsymbol{b}_k, \boldsymbol{f}_k\right). \quad (11)$$

A global optimization problem minimizing the sum of the individual energies from (11) for all K MUs can be formulated as

$$\begin{aligned}
\underset{\substack{\boldsymbol{x}_k, \boldsymbol{b}_k, \boldsymbol{f}_k, \\ \forall k \in \mathcal{K}}}{\arg\min} \quad & \sum_{k=1}^{K} E_k\left(\boldsymbol{x}_k, \boldsymbol{b}_k, \boldsymbol{f}_k\right), && (12)\\
\text{s.t.} \quad & x_{k,l}^{\text{AP}} \cdot T_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) \leq T_k^{\max}, \quad \forall k \,\&\, \forall l, && (12a)\\
& \sum_{k=1}^{K} b_{k,l} \leq b_l^{\max}, && \forall l, \quad (12b)\\
& \sum_{k=1}^{K} f_{k,l}^{\text{AP}} \leq f_l^{\text{AP max}}, && \forall l, \quad (12c)\\
& b_{k,l}, f_{k,l}^{\text{AP}} \geq 0, && \forall k \,\&\, \forall l, \quad (12d)\\
& x_k^{\text{MU}}, x_{k,l}^{\text{AP}} \in \{0,1\}, && \forall k \,\&\, \forall l, \quad (12e)\\
& x_k^{\text{MU}} + \sum_{l=1}^{L} x_{k,l}^{\text{AP}} = 1, && \forall k. \quad (12f)
\end{aligned}$$

Constraint (12a) is the flexible maximum offloading time constraint. The shared resources are limited by upper bounds, the communication resources in constraint (12b) and the computation resources in constraint (12c). Constraint (12d) ensures that only positive quantities of the resources are assigned to the MUs. The binarity of the offloading decision variables is handled by constraint (12e) and constraint (12f) ensures that each MU is either choosing local computation or computation at one AP.

Optimization problem (12) is a mixed-integer non-linear program (MINLP). A MINLP is in general NP-hard [12].

B. Minimum Required Resources for Offloading

For the GO, it is important to estimate the share of the communication and computation resources that is necessary so that offloading becomes beneficial for MU k. The minimum resources to make offloading possible can be found by solving the offloading time constraint (10) with equality, i.e

$$T_{k,l}^{\text{up}} + T_{k,l}^{\text{comp}} = T_k^{\max} \quad (13)$$

$$\frac{s_k^{\text{task}}}{b_{k,l} \log_2\left(1 + \frac{p_{k,l}^{\text{trans}} |h_{k,l}|^2}{\sigma^2}\right)} + \frac{c_k s_k^{\text{task}}}{f_{k,l}^{\text{AP}}} = t_k \cdot \frac{c_k s_k^{\text{task}}}{f_k^{\text{MU}}}, \quad (14)$$

with $t_k \in \{t_1, \ldots, t_6\}$. After a few reformulations, we obtain:

$$\begin{aligned}
& c_k \cdot b_{k,l} \cdot \log_2\left(1 + \frac{p_{k,l}^{\text{trans}} |h_{k,l}|^2}{\sigma^2}\right) \cdot \left(f_{k,l}^{\text{AP}} - t_k \cdot f_k^{\text{MU}}\right)\\
& = f_{k,l}^{\text{AP}} \cdot f_k^{\text{MU}}. \quad (15)
\end{aligned}$$

As the expression $f_{k,l}^{\text{AP}} - t_k \cdot f_k^{\text{MU}}$ has to be greater than 0 to lead to a feasible solution, this already reveals a minimum $f_{k,l}^{\min}$ for the shared computation frequency at cloudlet l:

$$0 < f_{k,l}^{\text{AP}} - t_k \cdot f_k^{\text{MU}} \quad (16)$$

$$f_{k,l}^{\min} = t_k \cdot f_k^{\text{MU}} < f_{k,l}^{\text{AP}}. \quad (17)$$

Under the assumption that a value for the computation frequency $f_{k,l}^{\text{AP}}$ of MU k at cloudlet l is fixed, (15) can also be used to calculate the minimum required bandwidth $b_{k,l}^{\min}$ to fulfill the offloading time as

$$b_{k,l}^{\min} = \frac{f_{k,l}^{\text{AP}}}{c_k \cdot \log_2\left(1 + \frac{p_{k,l}^{\text{trans}} |h_{k,l}|^2}{\sigma^2}\right) \cdot \left(\frac{f_{k,l}^{\text{AP}}}{f_k^{\text{MU}}} - t_k\right)}. \quad (18)$$

C. Matching Approach

The SPA problem is used as a basis for the matching algorithm proposed in this section. In general, the SPA problem describes the assignment of student projects offered by teachers to the students. Each student creates an ordered list of the preferred projects and applies for the projects. Similarly, the teachers have preferences which students have the most suitable skills to work on their projects. Algorithms to solve the SPA problem shall find the most suitable matching of the students to the projects. In [13], the authors propose and compare different solution strategies to the SPA problem. The algorithm used in this paper is based on [8].

The set $\mathcal{K}$ of MUs corresponds to the set of students and the set $\mathcal{L}$ of APs to the teachers in the SPA problem. The communication resources in form of the bandwidth $b_l^{\max}$ and the computation resources in form of the CPU frequency $f_l^{\max}$ are the equivalent of the projects. A MU k offloading its task to AP l is denoted by (k,l). If a MU k is computing its task locally, the empty set $\emptyset$ is assigned for the matching, because it does not require resources and is not matched to an AP.

A matching Y is defined as a subset of all offloading possibilities of the MUs and APs, i.e. $Y \subseteq \mathcal{K} \times \mathcal{L}$. Each MU k has preferences $\succ$ and a preference list PL_k^{MU} to indicate and store the locations with the highest benefit when offloading. MU k prefers an AP l over AP l' if the total energy for offloading to AP l is smaller than to AP l', i.e.

$$(k,l) \succ (k,l') \leftrightarrow E_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) \leq E_{k,l'}^{\text{AP}}\left(b_{k,l'}, f_{k,l'}^{\text{AP}}\right). \quad (19)$$

Similarly, every AP l has preferences $\succ$ and a preference list PL_l^{AP}. An example for the preferences of MU 1 in a scenario with 5 APs could be

$$(1,3) \succ (1,1) \succ (1,2) \succ (1,5) \succ \emptyset. \quad (20)$$

Note that, in this example AP 4 is out of reach for MU 1 as the maximum offloading time is always violated and/or offloading costs more energy than local computation.

The proposed matching algorithm is ensured to terminate with a feasible solution by the following definitions.

Definition 1 (Blocking Pairs)**.** A subset of MUs $\mathcal{K}^{\text{block}} \subseteq \mathcal{K}$ and an AP $l \in \mathcal{L}$ are called a blocking pair if one of the following conditions holds:

1) The MUs in $\mathcal{K}^{\text{block}}$ are not part of the matching set Y, i.e. $\mathcal{K}^{\text{block}} \cup Y = \emptyset$.
2) The MUs in $\mathcal{K}^{\text{block}}$ prefer AP l over their current AP l', i.e. $(\mathcal{K}^{\text{block}}, l) \succ (\mathcal{K}^{\text{block}}, l')$.
3) AP l prefers MUs in $\mathcal{K}^{\text{block}}$ over at least one of the MUs $\mathcal{K}^{\text{AP } l}$ already matched to it, i.e. $(\mathcal{K}^{\text{block}}, l) \succ (\mathcal{K}^{\text{AP } l}, l)$.

Definition 2 (Stability)**.** The matching set Y is stable if no blocking pairs exist.

Before starting the matching process, the GO can sort the MUs based on their task characteristics s_k^{task} and c_k, local CPU frequency f_k^{MU} or local processing energy E_k^{MU}. If the energy minimization of the MUs was not chosen to be the main objective of the GO, it could also prefer MUs with more expensive subscriptions and higher payments in the sorting process. This profit maximization is not considered in this paper.

The matching process starts with the evaluation of the payment preference of the MUs. The GO is informed about the subscription of the MUs and accordingly proposes offloading possibilities to the MUs. As introduced in Section II, the MUs decide for one of the possibilities and, thereby, on the payment of tokens to the MNO based on their tasks and battery levels. Then, the GO can generate the preference list PL_k^{MU} for each MU k. As shown in the example preference list, only possible offloading locations are stored. If the required bandwidth $b_{k,l}$ to offload the task of MU k to AP l exceeds the total available bandwidth $b_l^{\max}$, the infeasible solution is neglected.

Now, the actual matching process can begin. Each MU k is assigned to its most preferred AP l, i.e. (k, l), and the required communication and computation resources are reserved. We assume a fixed value higher than $f_{k,l}^{\min}$ for the CPU frequency at AP l and calculate $b_{k,l}^{\min}$ as shown in (18). After all MUs are matched, the APs that have more than the maximum available resources reserved, have to take a decision which MUs should offload and which MUs have to be unmatched. The easiest way to achieve the most beneficial set $\mathcal{K}^{\text{AP } l}$ of offloading MUs to AP l, is the exclusion of the MU that has the smallest energy saving E_k^{worst} by offloading their task, i.e.

$$E_k^{\text{worst}} = E_k^{\text{MU}} - E_{k,l}^{\text{AP}}. \tag{21}$$

This exclusion step is repeated until the AP has only reserved a feasible amount of communication and computation resources.

Afterwards, the GO reassigns an excluded MU k to the next AP l' in its preference list PL_k^{MU}. This procedure is repeated until all APs are not able to take more MUs for offloading. A MU k that is not assigned to any of its preferred APs, i.e. $\text{PL}_k^{\text{MU}} = \emptyset$, will compute its task locally.

The matching procedure is described as pseudo-code in Algorithm 1.

Algorithm 1 Flexible Matching Approach

```
Optional: GO sorts the set K of MUs
Empty matching set Y, all MUs undecided x_k = 0 ∀k;
for ∀k ∈ K do
   GO offers offloading possibilities for 0, 1 and 2 tokens;
   MU k decides for payment;
   GO generates PL_k^MU;
end for
% While any MU is unmatched
while ∃k ∉ Y do
   % Add MUs to the Matching Y
   for ∀k ∈ K do
      GO matches MU k to first entry in PL_k^MU, i.e. (k, l);
      Update matching Y = Y ∪ (k, l);
      Remove first entry from PL_k^MU;
   end for
   Exclude all MUs with E_k^MU − E_{k,l}^AP < 0 from matching Y
   for ∀l ∈ L do
      % Check if too much bandwidth is required at AP l
      while Σ_{k=1}^K b_{k,l} ≥ b_l^max do
         Find MU k with smallest energy savings E_k^worst;
         Remove MU k from matching Y = Y\(k, l);
      end while
      % Check if too much CPU frequency is required at AP l
      while Σ_{k=1}^K f_{k,l}^AP ≥ f_l^{AP max} do
         Find MU k with smallest energy savings E_k^worst;
         Remove MU k from matching Y = Y\(k, l);
      end while
   end for
end while
return Matching Y
```

One of the main advantages of the proposed matching algorithm is its small complexity. The maximum number of iterations is increasing linearly with the number of MUs and APs, i.e. $K \cdot L$. Each additional MU can have at maximum a preference list of length L if could offload to all L APs. Compared to the exponentially increasing number of different permutations $(L+1)^K$ of the offloading decisions of all K MUs, the algorithm can easily handle much larger networks.

IV. Numerical Results

To model the scenario described in the previous sections, we model an area of $200\,\text{m} \times 200\,\text{m}$ with 4 APs placed in the corners. All MUs are placed randomly inside this area with a minimum distance of $10\,\text{m}$ to any AP. We assume each MU to have one task to be computed locally or offloaded to one of the APs. The task belongs to one of three task classes:

- 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, enough communication and computation resources need to be available
- Type 3 (computation tasks, e.g. MATLAB): small size, high complexity, offloading is always preferred, enough computation resources need to be available

We tried to align the assumed task parameters close to measurements from [14] and [10]. The simulation parameters for the types of tasks are chosen similar to [7] and are summarized in Table I.

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

Each MU k has a local calculation power $p_k^{\text{calc}} = 1\,\text{W}$, a transmission power $p_{k,l}^{\text{trans}} = 200\,\text{mW}$ and a static power $p_{k,l}^{\text{static}} = 100\,\text{mW}$. The MUs are assumed to have a local comutation frequency of $f_k^{\text{MU}} = 2\,\text{GHz}$ and the cloudlets at the APs a total computation frequency of $f_l^{\text{maxAP}} = 20\,\text{GHz}$.

The radio access channel of AP l has a maximum available bandwidth of $b_l^{\max} = 10\,\text{MHz}$ and is separated from the radio access channels of the other APs. A white Gaussian noise power of $\sigma^2 = 10^{-13}$ W is assumed. With the Euclidean distance $d_{k,l}$, the channel gain between MU k and AP l is given by $|h_{k,l}|^2 = 1/d_{k,l}^3$.

The assumed factors $t_1, \ldots, t_6$ for flexible maximum time constraint that is dependent on the subscription and the payment willingness of the MUs are summarized in Table II.

TABLE II
DEPENDENCE OF MAXIMUM TIME FACTOR ON PAYMENT

Payment	0 Token	1 Token	2 Tokens
Cheap Subscription	$t_1 = 120\%$	$t_2 = 110\%$	$t_3 = 100\%$
Expensive Subscription	$t_4 = 100\%$	$t_5 = 90\%$	$t_6 = 80\%$

The following simulation results are obtained by at least 200 Monte Carlo runs per data point. Each run has randomly placed users, a random task type, a random subscription and a payment willingness dependent on a random battery level. A scenario with only local computation and a scenario with an assignment of the MUs to their closest AP by Euclidean distance are considered as basic reference schemes. In the latter case, all MUs are connected to their closest AP, the resources are distributed based on these decisions and all MUs that cannot fulfill the maximum offloading time constraint are changed to local computation.

For the proposed matching algorithm, four different resource allocation schemes are compared:

- Allocation 1: There exists a quota of how many MUs can connect to each AP l. The resources are shared equally among all MUs that decide for offloading to AP l.
- Allocation 2: Each MU k receives a cloudlet computation frequency $f_{k,l}^{\text{AP}}$ of 2.5 GHz or 3 GHz according to its subscription. The required bandwidth to fulfill the strict time constraint $T_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) \leq T_k^{\text{local}}$ is calculated using Equation (18).
- Allocation 3: Similar to Allocation 2, but with the proposed flexible bandwidth $T_{k,l}^{\text{AP}}\left(b_{k,l}, f_{k,l}^{\text{AP}}\right) \leq T_k^{\max}$.

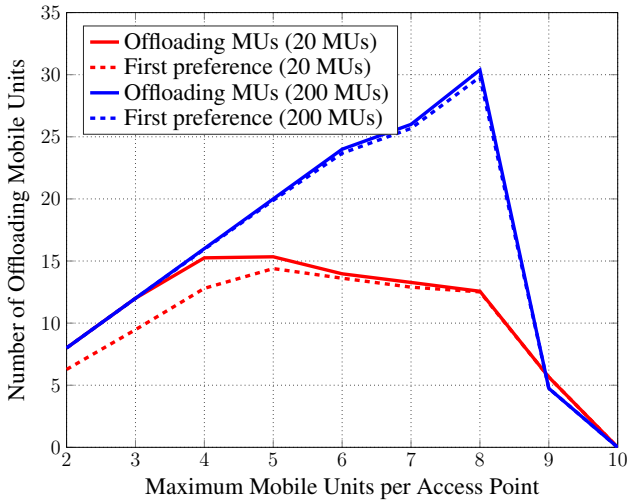

Fig. 2. Number of offloading MUs for different resource sharing quotas

- Allocation 4: Similar to Allocation 3, but a MU k receives additional 0.5 GHz computation frequency $f_{k,l}^{\text{AP}}$ for every token it pays.

In the first simulation result in Figure 2, the influence of the fixed quota found in the allocation scheme 1 is shown. The communication and computation resources are shared equally based on the quota, similar to [8]. We compare the number of MUs that can connect to each AP against the total number of MUs that are matched for offloading. Two extreme cases are considered. A small scenario with $K = 20$ MUs is shown in red and a large scenario with $K = 200$ MUs in blue. The dotted lines show how many of the offloading MUs got matched to their first preference from their individual preference list. In the scenario with 20 MUs, only up to 15 out of the 20 MUs offload their tasks, because some MUs have a small task of type 1 and a bad channel quality so that the maximum offloading time constraint cannot be fulfilled. In the scenario, with 200 MUs, many more MUs are matched for offloading. Most of these MUs have computation intensive tasks of type 2 or 3. The dotted line shows that nearly all matched MUs are offloading to their most preferred AP. If the MUs are assumed to have tasks with significantly different characteristics, the matching algorithm with the equal resource allocation strategy 1 only prefers the MUs with the largest tasks and energy savings as it is a greedy strategy. The steep drop in both blue lines is the effect of the equal resource allocation strategy. While the resources are sufficient to serve up to 8 MUs per AP, most MUs cannot fulfill their maximum time constraint with a quota of 9 or more MUs per AP. Therefore, in allocation scheme 1, the choice of a correct quota is crucial to the number of matched MUs.

The next simulation result in Figure 3 compares the two reference schemes and the four resource allocation schemes in terms of the energy requirements of the MUs. The sum of the energies of all MUs for the computation or offloading of their tasks is shown for a varying number of MUs. For allocation scheme 1, a quota of 7 MUs per AP is chosen.

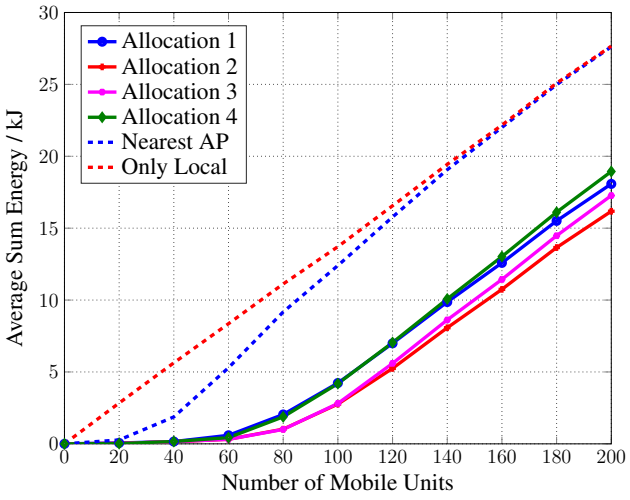

Fig. 3. Average sum energy for different numbers K of MUs, $f_l^{\text{maxAP}} = 20\,\text{GHz}$

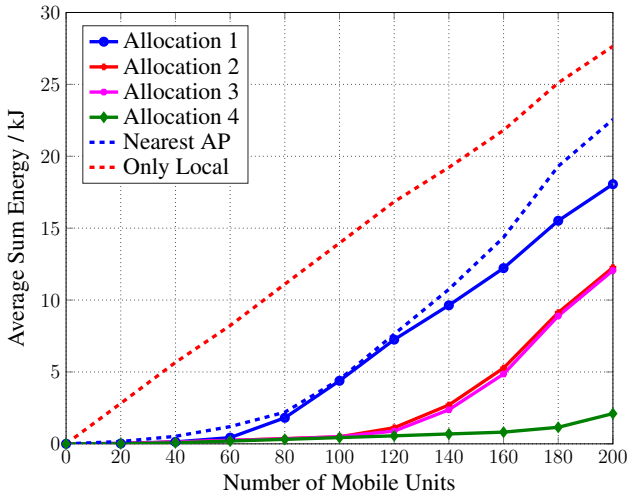

Fig. 4. Average sum energy for different numbers K of MUs, $f_l^{\text{maxAP}} = 100\,\text{GHz}$

Allocation schemes 2 and 3 are performing slightly better than the equal allocation scheme 1 and the most flexible scheme 4. Allocation scheme 4 is performing worse in this scneario, because the computation resources at the cloudlets are limited. As this scheme allocates more computation resources to MUs that are willing to pay tokens, the average number of offloading MUs is smaller than for allocation schemes 2 and 3.

TABLE III
NUMBER OF OFFLOADING MUS FOR THE DIFFERENT RESOURCE ALLOCATION STRATEGIES FOR A SCENARIO WITH 120 MUS

Allocation Strategy	1	2	3	4
No. of offloading MUs	23.97	28.41	27.49	23.86
No. of first preference	22.08	22.17	20.98	17.33
Percentage of first preference	92.1%	78.1%	76.3%	72.7%

In Table III, this influence on the number of offloading MUs is demonstrated in detail. The number of offloading MUs and the number of MUs that are offloading to their most preferred AP is shown for a scenario with 120 MUs. Allocation schemes 2 and 3 match around 4 MUs more to the APs for offloading. Furthermore, the table shows the inflexibility of allocation scheme 1 with the fixed quota. 92.1% of the matched MUs are matched to their first preference. The proposed allocation schemes 2-4 with more flexibility can adapt better to the scenario with different task types. These schemes support that also MUs with smaller tasks are matched to less preferred APs and still save energy compared to local computation.

If the 4 APs are assumed to be much more powerful with a total computation frequency of $f_l^{\text{maxAP}} = 100\,\text{GHz}$, the computation bottleneck from the previous result is exchanged with a bottleneck of the shared communication resources. Figure 4 shows the sum of energies of all MUs for this case. As the quota is unchanged at 7 MUs per AP, allocation scheme 1 is performing only slightly better than the reference scheme of matching MUs to their nearest AP. The equal resource allocation scheme proves to be heavily dependent on the right choice of the quota. For scenarios with fluctuating maximum resources, a constant estimation of the right quota and adaptation of the resource split is necessary.

The proposed flexible allocation schemes 2-4 do not require a quota. Allocation schemes 2 and 3 are again performing very similar to each other. On average 38.52 MUs and 39.03 MUs are able to offload in a scenario with 200 MUs for these schemes. The most flexible resource allocation scheme 4 now shows a clear advantage due to its adaptive allocation of the computation resources relative to the payment of the MUs. For the case of 200 MUs, 67.60 MUs are offloading on average and the sum energy is 82.55% less than the sum energy of allocation scheme 3.

V. CONCLUSION

The joint offloading decision making and resource allocation problem is rapidly increasing in complexity with an increasing network size. The proposed matching algorithm based on the SPA problem with a GO coordinating the offloading decisions and the available resources is able to easily handle large networks with hundreds of MUs and multiple APs. MUs with a demand for more communication and computation resources due to low battery levels can increase the problem complexity even further. The proposed flexible time constraint and resource allocation strategy takes this into account. It adapts the resources according to payments from the MUs to the MNO. In numerical simulations, the convergence of the matching algorithm is shown. The resource allocation strategy is tested against less flexible allocation strategies. In a scenario with plenty of computation resources, the proposed allocation strategy clearly outperforms the other tested strategies.

ACKNOWLEDGEMENT

This work has been performed in the context of the DFG Collaborative Research Center (CRC) 1053 MAKI and the BMBF project Open6GHub. This work has been supported by DAAD with funds from the German Federal Ministry of Education and Research (BMBF).

REFERENCES

[1] Cisco, "Cisco annual internet report (2018–2023)," Tech. Rep. C11-741490-01, 2020.

[2] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "MEC in 5G networks," *ETSI white paper*, vol. 28, pp. 1–28, 2018.

[3] D. Ferreira, A. K. Dey, and V. Kostakos, "Understanding human-smartphone concerns: a study of battery life," in *Proc. of the International Conference on Pervasive Computing*. Springer, 2011, pp. 19–33.

[4] S. Hosio, D. Ferreira, J. Goncalves, N. van Berkel, C. Luo, M. Ahmed, H. Flores, and V. Kostakos, "Monetary assessment of battery life on smartphones," in *Proc. of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 1869–1880.

[5] S. Jošilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. of the IEEE INFOCOM 2019 - Conference on Computer Communications*, 2019, pp. 2467–2475.

[6] 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)*, 2019, pp. 1–6.

[7] T. Mahn and A. Klein, "Energy-efficient application-aware mobile edge computing with multiple access points," in *Proc. of the 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–7.

[8] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in IoT fog computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7475–7484, 2018.

[9] F. Chiti, R. Fantacci, and B. Picano, "A matching game for tasks offloading in integrated edge-fog computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, pp. 1–14, 2020.

[10] S. Melendez and M. P. McGarry, "Computation offloading decisions for reducing completion time," in *Proc. of the 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2017, pp. 160–164.

[11] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems," in *2018 IEEE international conference on communications (ICC)*, 2018, pp. 1–6.

[12] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.

[13] K. Yahiro and M. Yokoo, "Game theoretic analysis for two-sided matching with resource allocation," in *Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems*, 2020, pp. 1548–1556.

[14] 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.