Tobias Mahn, Maximilian Wirth, and Anja Klein, ”Game Theoretic Algorithm for Energy Efficient Mobile Edge Computing with Multiple Access Points,” in *Proc. of the 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (Mobile Cloud)*, April 2020.

Game Theoretic Algorithm for Energy Efficient Mobile Edge Computing with Multiple Access Points

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

***Abstract*—This paper considers a Mobile Edge Computing scenario with multiple mobile units (MUs), multiple access points (APs) and one cloudlet server. The MUs have to decide whether offloading their computation tasks to the cloudlet is energy wise beneficial. As there are multiple APs available to connect the MUs to the cloudlet and communication and computation resources have to be shared among all MUs, each MU also has to choose the AP for transmission that minimizes its offloading energy under the given fraction of the overall resources. The problem is formulated as a energy minimization problem with a maximum offloading time constraint. MUs not only need to consider the energy required for local computation or offloading, but simultaneously avoid an overlong processing time of offloaded computation. This joint offloading decision and resource allocation is divided into two subproblems in the proposed approach. The resource allocation problem is reformulated by using Lagrange multipliers and closed-forms for the calculation of the shared resources are found. These results can be integrated into the proposed game theoretic algorithm for the offloading decision problem. The algorithm is based on a potential game and therefore, can be proven to converge to a Nash equilibrium. Numerical results show a benefit of the proposed resource allocation strategy, a performance of the proposed game algorithm near the optimal solution and a fast algorithm execution time that can even be significantly improved by proposed sorting metrics.**

***Index Terms*—mobile edge computing, joint optimization, resource allocation strategy, game theory**

I. Introduction

The popularity of smartphones and other battery-driven mobile devices in combination with an ever increasing number of demanding applications leads to the necessity to offload computation tasks. Even in the early days of smartphones with less available applications, the battery life of the devices has been the greatest concern of the users [1]. Cloud services with ubiquitous and elastic resources like Amazon EC2 or Microsoft Azure were proposed as the solution to computation offloading [2]. In the previous years with quickly rising traffic on the internet and billions of connected devices, a shift from sending the data to central cloud servers towards processing it at the edge of the network happened. This so called Mobile Edge Computing (MEC) shall offer significantly lower latency during the transmission of computation tasks [3] and enable applications that require nearly real-time computation like augmented reality or natural speech processing [4]. Especially with the 5G mobile networks, MEC plays an important role to guarantee low latency for communication and computing [5]. Furthermore, a recent forecast predicts billions of new machine to machine (M2M) devices until 2022. Usually, M2M devices are limited in their computational capabilities and available MEC servers, e.g. at cellular base stations or WiFi access points, are necessary to make M2M services possible or to enable more complex services [6].

MEC has often been studied in small scenarios with a single access point (AP) and a single MEC server or cloudlet. In early publications about MEC like [7], the authors even considered only a single mobile unit (MU) that has to decide between local computation and offloading to a MEC server. In this case, the single MU does not have to share the resources for communication and computation in the scenario. The authors of [8] and [4] investigate MEC scenarios with multiple MUs. Each MU has to decide whether offloading its task or local computation is more beneficial based on the partial network resources available to it. Both papers use optimization techniques to minimize their respective objectives. While the authors of [8] aim for a minimal power consumption of all MUs in the scenario, the authors of [4] optimize the maximum offloading and computation time in the considered network.

In [9] and [10], the scenarios are more complex. Multiple MUs can choose between local computation or offloading their tasks via one of multiple APs to one central cloudlet. Both papers are based on game theoretic models and investigate the offloading decisions of the MUs based on the limited and shared resources of the considered networks. In [9], the authors investigate a minimization of a joint objective of the required energy and time for the computation of the tasks of all MUs. The authors of [10] investigate a joint objective of the computation energy together with an abstract monetary cost function. While the available resources are split equally among offloading MUs in [9], in [10] a proportional fair resource allocation technique is shown. For the fair allocation, the offloading decisions are decoupled from the resource allocation and the resource allocation problem is reformulated by the use of Langrange multipliers. A similar result for the proportional fair allocation of shared resources is also obtained by our group for a multi-stage offloading scenario in [11].

This paper considers a similar MEC scenario with multiple MUs and multiple APs that are connected to a central cloudlet server. Based on its share of the communication and computation resources, each MU has to decide whether it should offload its task or better compute its task locally. In contrast to [9] and [10], a joint objective function is avoided. Previous

calculcations and simulations have shown that such a formulation is strongly dependent on the choice of the weighting factors between the two objective. Therefore, we propose a game theoretic algorithm for an energy minimization problem with a maximum offloading time constraint, which is based on the *Join and Play Best Replies* (JPBR) algorithm from [9]. The proportional fair allocation of the shared resources based on the characteristic parameters of each MU is integrated into the JPBR algorithm. Furthermore, it is shown how a relaxed time constraint influences the energy minimization problem and how a sorting strategy can lead to a significant improvement of the numerical results.

In Section II, we introduce the scenario and most relevant definitions of our offloading model. Afterwards in Section III, the offloading problem is formulated as a global energy minimization problem. This described problem is used as a basis for the game theoretic algorithm in Section IV. In Section V, the performance of the proposed approach is evaluated numerically.

II. System Model

A. Scenario

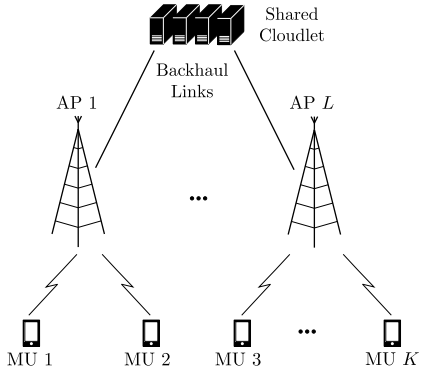

Fig. 1. Computation offloading scenario with K MUs, L APs and one shared cloudlet server

A set $\mathcal{K} = \{1, 2, \ldots, K\}$ of MUs is considered. Similarly to [12], it is assumed that the set of MUs remains unchanged during one offloading period. The MUs are located in an area that is covered by a set $\mathcal{L} = \{1, 2, \ldots, L\}$ of APs. The MUs can connect through radio access channels to the L APs. Each of the APs has a backhaul link to a shared cloudlet server that is able to perform computation for the MUs. In the following, only energies and times of the MUs are considered as the APs and the cloudlet are assumed to be connected to a steady power supply.

Each MU k has a non-splittable task that can be computed locally or offloaded to the cloudlet for remote computation. The task is characterized by its size t_k in bits and a complexity factor c_k, which is measured in central processing unit (CPU) cycles per bit. The complexity factor is necessary to account for the different computation requirements of a task, e.g. applying filters on a photo usually requires much less CPU cycles than the execution of a face detection algorithm in a video file. In [13], the authors investigated such measures for different types of computation tasks.

B. Local Computation

First, the possibility of local computation on the CPU of the MU is introduced. The CPU of each MU k has a processing frequency f_k^{MU} and requires a calculation power p_k^{calc} during operation under full load. Combined with the characteristics of the task of MU k, the local computation time can be calculated as

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

Similarly, the local computation energy is

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

C. Computation at the Cloudlet

As an alternative to local execution of the task, a MU can decide for offloading to the cloudlet. If the MU decides for offloading, the task will be sent over a radio access channel to an AP and then forwarded via the backhaul link from the AP to the cloudlet. As available communication and computation resources in the network have to be shared among all MUs, a model for the transmission and remote computation is introduced in the following paragraphs.

Transmission: In the first step, MU k transmits its task to AP l. A MU can only connect to one AP simultaneously. It it assumed that each AP l has a separate transmission channel with given maximum bandwidth b_l^{max}. If multiple MUs decide to offload their tasks over the same AP l, MU k only receives a fraction $b_{k,l}$ of the total available bandwidth b_l^{max}. To avoid interference among multiple MUs transmitting to AP l, each AP is assumed to use an orthogonal transmission scheme. The uplink rate can be expressed by the Shannon channel capacity as

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

where the uplink channel gains $|h_{k,l}|^2$ and the white Gaussian noise power σ^2 are assumed to be known at the AP. The channel gains $|h_{k,l}|^2$ gains are determined by the distance law with distance $d_{k,l}$ and the environmental parameter α. The transmission time of MU k to AP l can be expressed as

$$T_{k,l}^{\text{trans}} = \frac{t_k}{r_{k,l}^{\text{uplink}}}. \tag{4}$$

Each MU has two more powers that have to be considered. The transmit power p_k^{trans} is the equivalent power of the radio access hardware that is activated during the transmission. The static power p_k^{static} shall account for the power of all other hardware components of the device while communicating or being idle until the result of the task is computed at the cloudlet. With these powers, the transmission energy over the radio access channel is

$$E_{k,l}^{\text{trans}} = \left(p_k^{\text{trans}} + p_k^{\text{static}}\right) \frac{t_k}{r_{k,l}^{\text{uplink}}}. \tag{5}$$

After AP l received the task of MU k, it forwards the task to the cloudlet over the backhaul link with a maximum transmission rate $r_l^{\text{backhaul max}}$. Similar to the shared bandwidth, MUs have to share the maximum available rate when multiple MUs use the same AP l for offloading. In this case, MU k receives a fractional rate $r_{k,l}^{\text{backhaul}}$. Accordingly, the backhaul transmission time of the task of MU k from AP l to the cloudlet server can be defined as

$$T_{k,l}^{\text{backhaul}} = \frac{t_k}{r_{k,l}^{\text{backhaul}}} \tag{6}$$

and the energy MU k consumes during backhaul transmission as

$$E_{k,l}^{\text{backhaul}} = p_k^{\text{static}} \frac{t_k}{r_{k,l}^{\text{backhaul}}}. \tag{7}$$

A transmission of the computation results back from the cloudlet to the MUs is neglected, since the data size of the result can in many applications be assumed to be significantly smaller that the original task size, similar to [4], [14].

Cloudlet Computation: After the cloudlet received the task of MU k, it can start the computation. The cloudlet is assumed to have a higher CPU frequency $f^{\text{AP max}}$ than any of the MUs, i.e. $\max\left\{f_k^{\text{MU}}\right\} < f^{\text{AP max}}$. Multiple offloading MUs have to share the computation resources at the cloud and in this case, MU k receives an equivalent computation frequency f_k^{AP} at the cloudlet. The computation time of the task of MU k at the cloudlet is expressed by

$$T_k^{\text{comp}} = \frac{c_k t_k}{f_k^{\text{AP}}} \tag{8}$$

and the energy the MU consumes during the computation by

$$E_k^{\text{comp}} = p_k^{\text{static}} \frac{c_k t_k}{f_k^{\text{AP}}}. \tag{9}$$

D. Overall Energy and Time Definitions

After introducing all individual times and energies to be considered in the computation offloading scenario, the overall model can be formulated. Each MU k can decide for local computation or for offloading its task via AP l. Therefore, an offloading decision vector $\mathbf{x}_k = [x_k^{\text{MU}}, x_{k,1}^{\text{AP}}, \ldots, x_{k,L}^{\text{AP}}]$ is used to store the offloading decision of MU k. This binary vector has a 1 at the position of the offloading decision and is 0 otherwise. x_k^{MU} represents the local computation, while $x_{k,l}^{\text{AP}}$ is used for the decision to offload to AP l.

A MU deciding for offloading also requires at least a fraction of the shared communication and computation resources to be able to transmit and compute its task at the cloudlet. The fractions of the communication resources of MU k are stored in two vectors, one vector containing the allocated bandwidths $\mathbf{b}_k = [b_{k,1}, \ldots, b_{k,L}]$ and the other vector containing the allocated backhaul transmission rates $\mathbf{r}_k = [r_{k,1}, \ldots, r_{k,L}]$. Finally, the allocated fraction of the computation frequency f_k^{AP} for offloading MU k is stored in a scalar value.

In the following, the times and energies are defined as functions dependent on the offloading decisions and shared resources.

The combination of (4), (6) and (8) leads to the overall offloading time of MU k as

$$T_{k,l}^{\text{AP}}\left(b_{k,l}, r_{k,l}, f_k^{\text{AP}}\right) = T_{k,l}^{\text{trans}} + T_{k,l}^{\text{backhaul}} + T_{k,l}^{\text{comp}}. \tag{10}$$

In a similar way, (5), (7) and (9) can be combined to the overall offloading energy as

$$E_{k,l}^{\text{AP}}\left(b_{k,l}, r_{k,l}, f_k^{\text{AP}}\right) = E_{k,l}^{\text{trans}} + E_{k,l}^{\text{backhaul}} + E_{k,l}^{\text{comp}}. \tag{11}$$

As the offloading decision vector $\mathbf{x}_k$ has only one non-zero element, the energy of MU k required for the computation of its task based on the offloading decision can be written as

$$E_k\left(\boldsymbol{x}_k, \boldsymbol{b}_k, \boldsymbol{r}_k, f_k^{\text{AP}}\right) = x_k^{\text{MU}} \cdot E_k^{\text{MU}} + \sum_{l=1}^{L} x_{k,l}^{\text{AP}} \cdot E_{k,l}^{\text{AP}}\left(b_{k,l}, r_{k,l}, f_k^{\text{AP}}\right). \tag{12}$$

III. Energy Minimization Problem

After defining the offloading energy of one MU k in (12), the sum of the energies of all MUs is used as the objective of the global energy minimization problem

$$\underset{\substack{\boldsymbol{x}_k, \boldsymbol{b}_k, \boldsymbol{r}_k, \\ f_{k,l}^{\text{AP}}, \forall k \in \mathcal{K}}}{\arg\min} \quad \sum_{k=1}^{K} E_k\left(\boldsymbol{x}_k, \boldsymbol{b}_k, \boldsymbol{r}_k, f_{k,l}^{\text{AP}}\right), \tag{13}$$

$$\text{s.t.} \quad x_{k,l}^{\text{AP}} T_{k,l}^{\text{AP}}\left(b_{k,l}, r_{k,l}, f_{k,l}^{\text{AP}}\right) \leq T_k^{\text{MU}}, \quad \forall k \,\&\, \forall l, \tag{13a}$$

$$\sum_{k=1}^{K} b_{k,l} \leq b_l^{\max}, \qquad \forall l, \tag{13b}$$

$$\sum_{k=1}^{K} r_{k,l}^{\text{backhaul}} \leq r_l^{\text{backhaul max}}, \qquad \forall l, \tag{13c}$$

$$\sum_{k=1}^{K} f_k^{\text{AP}} \leq f^{\text{AP max}}, \tag{13d}$$

$$b_{k,l}, r_{k,l}, f_{k,l}^{\text{AP}} \geq 0, \qquad \forall k \,\&\, \forall l, \tag{13e}$$

$$x_k^{\text{MU}}, x_{k,l}^{\text{AP}} \in \{0, 1\}, \qquad \forall k \,\&\, \forall l, \tag{13f}$$

$$x_k^{\text{MU}} + \sum_{l=1}^{L} x_{k,l}^{\text{AP}} = 1, \qquad \forall k. \tag{13g}$$

The constraints can be categorized into three groups. The first constraint (13a) shall ensure a good quality of experience for the MU by restricting the maximum offloading time to an upper limit. In this case, the offloading of a task has to be at least as fast as the local computation on the CPU of the MU.

The next group of constrains (13b) to (13e) regulates the shared resources. Constraints (13b) to (13d) restrict the sum of the allocated fractions of the resources to their respective upper limit. Constraint (13e) prevents the case that the optimization could allocate negative resources.

The last two constraints (13f) and (13g) control the offloading decisions. Constraint (13f) is used to ensure the binarity of the offloading problem and constraint (13g) secures that each MU takes exactly one offloading decision.

Although the proposed offloading problem can be formulated in a compact form, it is a mixed-integer non-linear program (MINLP) that cannot be solved efficiently. Furthermore,

the number of constraints in (13) is rapidly increasing with the numbers K of MUs and L of APs. In the next section, an algorithm based on a game theoretic model is investigated to find an approximation of (13).

IV. Game Theoretic Algorithm

The presented energy minimization problem in Section III jointly optimizes the offloading decisions of the MUs and the resource allocation of the shared communication and computation resources. A global optimum point can only be computed when knowledge about all characteristics of the scenario is available at one entity in the network. Usually, this assumption cannot be made and therefore, a heuristic algorithm with distributed partial knowledge is an alternative to the central optimization. For this algorithm, we separate the offloading decisions and the resource allocation.

A. Resource Allocation Strategy

For the resource allocation, a simplified version of (13) is considered as

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

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

$$\sum_{k=1}^{K} r_{k,l}^{\text{backhaul}} \le r_l^{\text{backhaul max}}, \qquad \forall l, \quad (14b)$$

$$\sum_{k=1}^{K} f_k^{\text{AP}} \le f^{\text{AP max}}, \qquad (14c)$$

$$b_{k,l}, r_{k,l}, f_{k,l}^{\text{AP}} \ge 0, \qquad \forall k \,\&\, \forall l, . \quad (14d)$$

The first simplification of removing constraints (13f) and (13g) comes from the assumption that the offloading decisions are known, since they are solved in a second separate problem. Furthermore, the maximum time constraint (13a) is neglected as the resource controller does not need to be aware of the quality of experience of the individual MUs.

Although (14) is still a non-linear optimization problem, optimal solutions for the shared resources can be found analytically. As the radio access channel bandwidths, the backhaul rates and the computation frequency at the cloudlet are independent of each other, each can be treated separately. When reformulating the problem using Lagrange multipliers and solving the corresponding KKT-conditions, as shown in [10] and [11], closed-form solutions for the resource allocation can be found. The fraction of the bandwidth $b_l^{\max}$ for MU k offloading via AP l is given as

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

the fraction of the backhaul transmission rate $r_{k,l}^{backhaul*}$ for MU k offloading via AP l can be calculated as

$$r_{k,l}^{backhaul*} = r_l^{\text{backhaul max}} \cdot \frac{\sqrt{x_{k,l}^{\text{AP}} p_k^{\text{trans}} t_k}}{\sum_{k=1}^{K}\sqrt{x_{k,l}^{\text{AP}} p_k^{\text{trans}} t_k}} \tag{16}$$

and the fraction of the computation frequency at the shared cloudlet for MU k can be calculated as

$$f_k^{\text{AP}\,*} = f^{\text{AP max}} \cdot \frac{\sqrt{x_{k,l}^{\text{AP}} p_k^{\text{static}} c_k t_k}}{\sum_{k=1}^{K}\sqrt{x_{k,l}^{\text{AP}} p_k^{\text{static}} c_k t_k}}. \tag{17}$$

B. Offloading Decision Game

Eqs. (15) to (17) can now be used to allocate the resources for known offloading decision. In this section, the game theoretic algorithm is defined to find the offloading decisions.

Similar to [15] and [9], we define a strategic form game $\mathcal{G}$ as an ordered triplet

$$\mathcal{G} = \left(\mathcal{K}, \{\mathcal{S}_k\}_{k\in\mathcal{K}}, \{u_k\}_{k\in\mathcal{K}}\right), \tag{18}$$

The set $\mathcal{K}$ of players is equivalent to the set of MUs, the set of strategies corresponds to their possible offloading decisions

$$\mathcal{S}_k = \Bigg\{ s_k = (x_k^{\text{MU}}, x_{k,1}^{\text{AP}}, \ldots, x_{k,L}^{\text{AP}}) \;\Bigg|\; x_k^{\text{MU}}, x_{k,l}^{\text{AP}} \in \{0,1\};\; x_k^{\text{MU}} + \sum_{l=1}^{L} x_{k,l}^{\text{AP}} = 1 \Bigg\}. \tag{19}$$

A MU plays one of its strategies $s_k \in \mathcal{S}_k$ and the played strategies of all K MUs can be formulated as $\boldsymbol{s} = (s_k, s_{-k})$. This notation of the strategy profile separates the played strategy of MU k as s_k from the strategies of all other MUs $\mathcal{K}\backslash\{k\}$ as s_{-k}. Using the strategy profile $\boldsymbol{s}$, the utility function for each MU k can be defined as the required energy for the computation of its task from (12) as $u_k(\boldsymbol{s}) = E_k\left(\boldsymbol{x}_k,\boldsymbol{b}_k,\boldsymbol{r}_k,f_k^{\text{AP}}\right)$.

The strategy profile and utility function are used to define a state, when a game terminates: A Nash equilibrium (NE) is a strategy profile $\boldsymbol{s}^*$ where none of the MUs can decrease its utility function when deviating from their strategy s_k^*, i.e. $u_k(s_k^*, s_{-k}) \le u_k(s_k, s_{-k}), \forall s_k \in \mathcal{S}_k$. Although a NE does not always exist for general strategic form games, it has been proven by the authors of [16] that a potential game has at least one NE.

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

For the description of the following game theoretic algorithms, we define two sets of MUs: the set of offloading MUs to AP l, $O_l(\mathbf{s}) = \{k | x_{k,l}^{\text{AP}} = 1\}$ and the set of all offloading MUs $O = \cup_{l \in \mathcal{L}} O_l$. Furthermore, the number of MUs offloading to AP l in iteration step t is denoted by $n_l(t)$ and the number of all offloading MUs is denoted by $n_O(t)$.

JPBR Algorithm: Due to space limitations, the original JPBR algorithm is only described shortly and no pseudo-code is presented. A detailed version can be found in [9]. The JPBR algorithm is iterative and one MU joins the game after another. It consists of two main functions:

- *ImproveAP*: The shared resources of the APs are congested by multiple MUs that decided for offloading and at least one MU wants to switch the AP it uses for offloading. In decreasing order of their possible improvement by switching, the unsatisfied MUs play their best response and switch from AP l to AP l', which minimizes their utility function.
- *Update Phase*: In the current time step t, one MU joined the game and decided for offloading to AP l. Now one AP l has one more offloading MU, i.e. $n_l(t) = n_l(t-1)+1$. In this case, it has to be evaluated whether another MU offloading via AP l can now improve its utility function by computing locally instead of offloading its task.
  Otherwise, if no MU decided for offloading due to congested APs in time step t, all APs are evaluated whether one MU should stop offloading. If one MU at AP l' stops offloading, the locally computing MU who can improve its utility the most by switching its decision to offloading via AP l' is added to the set of offloading MUs. Otherwise, the remaining $n_O(t)-1$ offloading MUs change their offloading decisions using *ImproveAP*.

The *ImproveAP* function in [9] is proven to posses a potential function and to terminate in a finite number of improvement steps in a NE, as proven by [16]. Therefore, the JPBR algorithm also terminates when no more MU can join the set of offloading MUs and all offloading MUs are satisfied with their strategy after the execution of the *ImproveAP* function. The JPBR algorithm terminates after all MUs are added to the game and no MU wants to deviate from its strategy any more.

Adaptation of JPBR Algorithm to JPBRTC Algorithm: The JPBR algorithm has some drawbacks: The available resources are distributed equally among all offloading MUs and the algorithm can only be used for the joint objective function where energy and time are combined by weighting factors. In the next paragraphs, it is explained how the JPBR algorithm is modified to integrate the proposed resource allocation strategy of (15) to (17) and the maximum offloading time constraint of (13a). The proposed algorithm is called *Join and Play Best Replies Under a Maximum Time Constraint* (JPBRTC).

When implementing the proposed resource allocation strategy into the JPBR and JPBRTC algorithms, the offloading decisions of the MUs are unknown during the execution the *ImproveAP* and *Update Phase* functions. Therefore, we had to combine the equal distribution of the JPBR algorithm with averaged versions of (15) to (17). By this, we get an estimate of the resources based on the characteristics of all MUs. The adapted fraction bandwidth for MU k offloading to AP l is estimated as

$$\hat{b}_{k,l}^{*} = \frac{b_l^{\max}}{n_l} \cdot \frac{\sqrt{\frac{x_{k,l}^{\text{AP}}(p_k^{\text{trans}}+p_k^{\text{static}})t_k}{\log_2\left(1+\frac{p_k^{\text{trans}}|h_{k,l}|^2}{\sigma^2}\right)}}}{\frac{1}{K}\sum_{k=1}^{K}\sqrt{\frac{(p_k^{\text{trans}}+p_k^{\text{static}})t_k}{\log_2\left(1+\frac{p_k^{\text{trans}}|h_{k,l}|^2}{\sigma^2}\right)}}}, \tag{20}$$

the estimated fraction of the backhaul transmission rate for MU k offloading via AP l as

$$\hat{r}_{k,l}^{backhaul*} = \frac{r_l^{\text{backhaul max}}}{n_l} \cdot \frac{\sqrt{x_{k,l}^{\text{AP}} p_k^{\text{trans}} t_k}}{\frac{1}{K}\sum_{k=1}^{K}\sqrt{p_k^{\text{trans}} t_k}} \tag{21}$$

and the estimated fraction of the computation frequency of MU k at the cloudlet as

$$\hat{f}_k^{\text{AP}\,*} = \frac{f^{\text{AP max}}}{n_O} \cdot \frac{\sqrt{x_{k,l}^{\text{AP}} p_k^{\text{static}} c_k t_k}}{\frac{1}{K}\sum_{k=1}^{K}\sqrt{p_k^{\text{static}} c_k t_k}}. \tag{22}$$

The estimates (20) to (22) are used during the execution of the the functions, while the proportional fair resource allocation from (15) to (17) is calculated after the execution of each function. The use of the estimated resources also has the benefit that a potential function can still be found and the *ImproveAP* function converges to a NE with the adapted resource allocation strategy.

The full pseudo-code of the JPBRTC algorithm can be found in Algorithm 1. We added a comment next to most functions to explain the working principle during the execution of the code. The algorithm is iterative and one MU is added to the game at each iteration step. Before the next MU joins the game in time step t, the MUs already participating in the game have reached a NE in the previous time step $t-1$. The integration of the maximum time constraint into the algorithm is done by excluding the MU k' with the highest time violation for the round and add it again back to the game after all remaining MUs have reached a NE with strategies $s_{-k'}(t)$. After all K MUs are taking part in the game, no MU violates its time constraint and no MU wants to deviate from the found strategy profile $\boldsymbol{s}^*$, the NE is reached and the algorithm terminates.

V. Numerical Results

A. Simulation Parameters

To model the scenario for the numerical simulations, we assume the MUs and APs to be in a $1\,\text{km} \times 1\,\text{km}$ square area. For most simulations, 30 MUs and 5 APs are placed in uniform random distribution. Each MU k has a task with a size t_k in the range of $[2, 100]$ MB, drawn from a uniform random distribution. The channel gains between MU k and AP l are dependent on the Euclidean distance $d_{k,l}$ and are calculated as $|h_{k,l}|^2 = 1/d_{k,l}^{\alpha}$ with a path loss coefficient $\alpha = 4$ for urban areas. The white Gaussian noise power is assumed to be $\sigma^2 = 10^{-13}$ W. Each available bandwidth $b_l^{\max}$ of the radio access channels is drawn from a uniform distribution in the

Algorithm 1 JPBRTC Algorithm

Initialize $t = 0$, $k = 1$
Initial MU decision:
$s_1(t=0) = \boldsymbol{s}^*(0) = \underset{\{s_1 \in \mathcal{S}_1 | T_1(s_1) < T_1^{\text{MU}}\}}{\arg\min} u_1(s_1)$
$t = t + 1$
for $k = 2 : K$ **do**
  % Already added MUs have reached a NE $\boldsymbol{s}^*(t-1)$
  $s_k(t) = \underset{\{s_k \in \mathcal{S}_k \,|\, T_k(s_k, s^*_{-k}(t-1)) < T_k^{\text{MU}}\}}{\arg\min} u_k(s_k, s^*_{-k}(t-1))$
  Obtain: $\boldsymbol{s}(t) = (s_k(t), s^*_{-k}(t-1))$
  Perform *Update Phase* function to obtain the NE $\boldsymbol{s}^*(t)$
  % Check if a mobile user violates the time constraint:
  while $O_{T^{\text{MU}} < T}(\boldsymbol{s}(t)) \neq \emptyset$ **do**
    % Find MU k' most exceeding time constraint:
    $k' = \underset{\{k \in O(\boldsymbol{s}(t)) \,|\, T_k(\boldsymbol{s}(t)) > T_k^{\text{MU}}\}}{\arg\max} \frac{T_k(\boldsymbol{s}(t))}{T_k^{\text{MU}}}$
    % Remove played strategy $x^{\text{AP}}_{k,l}$ from the set of strategies
    % for k' and remove MU k' from the game:
    $\mathcal{S}_{k'} = \{x^{\text{MU}}_{k'}, x^{\text{AP}}_{k,1}, \ldots, x^{\text{AP}}_{k,l-1}, x^{\text{AP}}_{k,l+1}, \ldots, x^{\text{AP}}_{k,L}\}$
    $\boldsymbol{s}'(t) = s_{-k'}(t)$
    % Check if the MU k' can be replaced by a MU that is
    computing locally:
    if $\exists k'' \in \mathcal{K} \backslash O(\boldsymbol{s}'(t))$ such that
    $u_{k''}(x^{\text{MU}}_{k''}, s_{-k''}(t)) > u_{k''}(x^{\text{AP}}_{k'',l}, s'_{-k''}(t))$ **then**
      % Let the MU with lowest reluctance start offloading:
      $k'' = \underset{\{k \in \mathcal{K} \backslash O(\boldsymbol{s}'(t)) \,|\, E^{\text{MU}}_{k''}(\boldsymbol{s}(t)) > E^{\text{AP}}_{k,l}(x^{\text{AP}}_{k,l}, s'_{-k}(t))\}}{\arg\min} \frac{E^{\text{AP}}_{k,l}(x^{\text{AP}}_{k,l}, s'_{-k}(t))}{E^{\text{MU}}_{k}(\boldsymbol{s}(t))}$
      $\boldsymbol{s}''(t) = (x^{\text{AP}}_{k'',l}, s'_{-k''}(t))$
    else
      % If no MU wants to start offloading, perform *ImproveAP*,
      % since AP l has one offloading MU less than before:
      $\boldsymbol{s}'(t) = ImproveAP(\boldsymbol{s}'(t))$
      % Some MU computing locally may now want
      % to start offloading via AP $m \in \mathcal{L} \backslash \{l\}$:
      if $\exists k'' \in \mathcal{K} \backslash O(\boldsymbol{s}'(t)) \quad \exists m \in \mathcal{L} \backslash \{l\}$ such that
      $u_{k''}(x^{\text{MU}}_{k''}, s_{-k''}(t)) > u_{k''}(x^{\text{AP}}_{k'',l}, s'_{-k''}(t))$ **then**
        % Find MU that profits the most when switching
        % to offloading and the corresponding AP:
        $(k'', m'') = \underset{\{(k,m) \in \mathcal{K} \backslash O(\boldsymbol{s}'(t)) \times \mathcal{L} \backslash \{l\}\}}{\arg\min} \frac{E^{\text{AP}}_{k,m}(x^{\text{AP}}_{k,m}, s'_{-k}(t))}{E^{\text{MU}}_{k}(\boldsymbol{s}(t))}$
        $\boldsymbol{s}''(t) = (x^{\text{AP}}_{k'',m''}, s_{-k''}(t))$
        % Some offloading MUs may want to deviate from
        % their decision after MU k'' starts offloading:
        $\boldsymbol{s}''(t) = ImproveAP(\boldsymbol{s}''(t))$
      end if
    end if
    % Add MU k' back to the game and let it
    % take an initial offloading decision:
    $s_{-k'}(t) = s_{k''}(t)$
    $s_{k'}(t) = \underset{\{s_{k'} \in \mathcal{S}_{k'} \,|\, T_{k'}(s_{k'}, s_{-k'}(t)) < T_{k',local}\}}{\arg\min} u_{k'}(s_{k'}, s_{-k'}(t))$
    Obtain: $\boldsymbol{s}'(t) = (s_{k'}(t), s_{-k'}(t))$
    Perform the *Update Phase* function to obtain the NE $\boldsymbol{s}^*(t)$
    $t = t + 1$
  end while
end for

range of $[50, 150]$ MHz and the available backhaul link rates $r_l^{\text{backhaul max}}$ in the range of $[50, 150]$ Mbit/s.

The CPU frequencies f_k^{MU} are also continuously uniformly distributed in the range of $[0.5, 1.5]$ GHz, while the cloudlet server has a fixed effective computation frequency $f^{\text{AP max}} = 100$ GHz. Each MUs is assigned with three powers drawn from uniform random distributions: the calculation power p_k^{calc} in the range of $[1.5, 2.5]$ W, the gross transmit power p_k^{trans} in the range of $[0.5, 1]$ W and the static power p_k^{static} in the range of $[0.1, 0.5]$ W. The assumed gross transmit power is higher than the maximum allowed transmit power of the LTE or 802.11 standards in order to include the power required for the radio hardware of the MUs, as measurements in [17] have shown.

B. Performance of the Proposed Resource Allocation Strategy

To be able to compare the new resource allocation strategy (RAS) from (15) to (16) as well as (20) to (22) to the equal distribution (ED) from [9], the original JPBR algorithm is used for the first simulation results. The ED is obtained by dividing the available resources by the number n_l of MUs offloading via AP l. As JPBR is used, the newly introduced maximum time constraint (13a) is neglected and the original joint objective function of energy and time combined by weighting factors γ_k^E for energy and γ_k^T for time is reintroduced.

500 Monte-Carlo runs with new randomly drawn parameters per run are used to calculate each data point of the following simulations. In Figure 2, we compare ED against the proposed RAS for different choices of the energy and time weights. The original simulations in [9] use a random choice of the weighting factors from a continuous uniform distribution on $[0, 1]$. In this case, most MUs will not completely neglect the maximum time that offloading their computation task requires. This result is shown as dashed blue line in Figure 2. The two solid curves with $\gamma_k^E = 1$ represent the results when the algorithm only considers the energy weights. It is visible that the proposed RAS outperforms the ED from [9]. The dashed red curve in the plot shows the result if only the time $\gamma_k^T = 1$ is considered as objective of the JPBR algorithm. It is clearly visible that the algorithm will perform worse in terms of the sum energy.

Since the JPBR algorithm does not consider a maximum time constraint, we are interested to see the performance in the same simulation scenario as in Figure 2 in terms of the computation delay for offloading MUs. The results in Figure 3 show the percentage of offloading MUs whose offloading time is longer than a local computation would have been. If the objective lies only on the computation time, i.e. $\gamma_k^T = 1$, which is again represented by the dashed red curve, about 10% of the offloading MUs would exceed the maximum time constraint. The other results are much worse and for between 38.6% and 98.5% of the offloading MUs, offloading of the task takes longer than a local computation. When comparing the solid curves in Figure 3, the proposed RAS still performs slightly better than the ED from [9].

Both results show that the JPBR algorithm in its original form can optimize a mixture of energy and time as optimization objectives, but offloading may be a slower experience than local computation for many MUs.

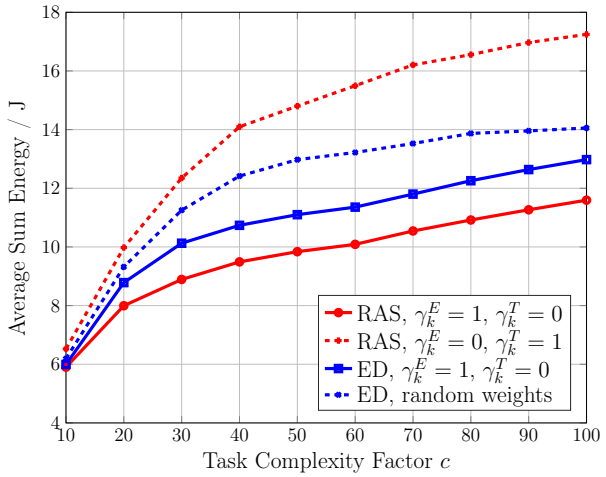

Fig. 2. Average energy comparison of JPBR with ED vs. proposed RAS

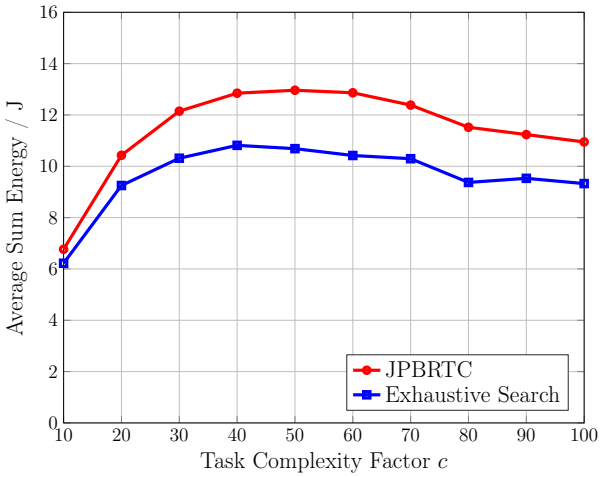

Fig. 4. Average energy comparison of optimal solutions by exhaustive search vs. the proposed JPBRTC algorithm

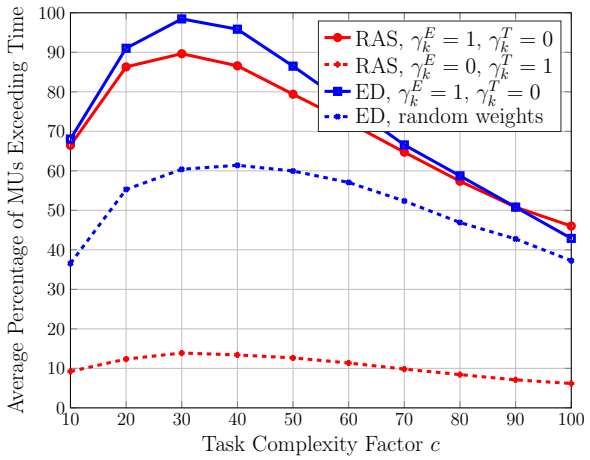

Fig. 3. Comparision of the avergage percentage of MUs that exceed the maximum offloading time

TABLE I
OFFLOADING GAIN FOR DIFFERENT COMPLEXITY FACTORS c_k

g of Complexity Factors c_k	10	40	70	100
g of Exhaustive Search	1.26	2.90	5.33	8.41
JPBRTC	1.16	2.44	4.43	7.16

C. Performance of the JPBRTC Algorithm

After discussing the original JPBR algorithm, the proposed JPBRTC algorithm is evaluated. As a direct comparison of JPBR and JPBRTC is not possible due to the different objective functions, the optimal solution of the energy minimization problem (13) is used as a benchmark. This solution is computed by finding the minimum sum energy through exhaustive search over all possible combinations of offloading decisions of all MUs. In the simulations for Figures 4 and 5, only 200 Monte-Carlo runs as well as 10 MUs and 3 APs are considered, because the number of possible combinations is growing rapidly by $(L+1)^K$, i.e. $4^{10} \approx 1.04$ million combinations.

Figure 4 shows the results obtained by the JPBRTC algorithm and the exhaustive search. The curves show a noticeable difference in the average energy consumption of all MUs, but the shape of both curves is comparable. The offloading gain $g = E_{\text{system without offloading}}/E_{\text{system with offloading}}$ for both curves is summarized in Table I for different task complexity factors c_k. Although there is also a gap between the JPBRTC and the optimal solution, JPBRTC still shows a great improvement over only local computation as shown in Table I.

Further possible modifications are also investigated. The first one is a relaxation of the tight maximum time constraint to 110% and 120% of the local computation time, given in Figure 5. The optimal solution is only computed for one maximum time constraint, as the computation of all offloading combinations requires much time. The red curves show the impact of the relaxed time constraint on the results of the JPBRTC algorithm. More MUs decide for offloading and the sum energy of all MUs is lowered by up to 7.97% for 110% T_k^{MU} and up to 14.75% for 120% T_k^{MU}. If the MUs are willing to lower their quality of offloading experience just by a small margin, it is possible to reduce the sum energy significantly.

An even higher improvement can be achieved by sorting the MUs before the execution of the algorithm. The results of the sorting are shown in Figure 6 and Table II. A simple option is to sort the MUs in decreasing order of the task complexities c_k. This already leads to a decrease of the execution time up to 38.07%. Sorting in decreasing order by the their local computation energy E_k^{MU} can even lead to up to 59.58% improvement. Many MUs who enter the game in the beginning play their best offloading strategy, because the APs are not congested in the beginning, and will not change this offloading decision during the algorithm runtime any more. More sophisticated sorting metrics would be possible, e.g. considering the channel characteristics between the MUs and the APs, but these would require more knowledge about nearly

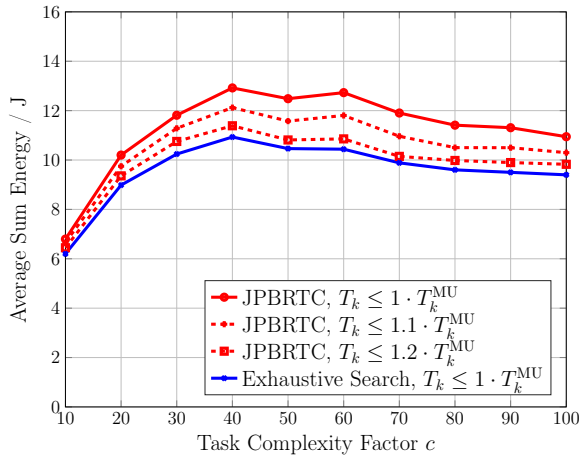

Fig. 5. Difference in the sum energy for relaxed maximum time constraint

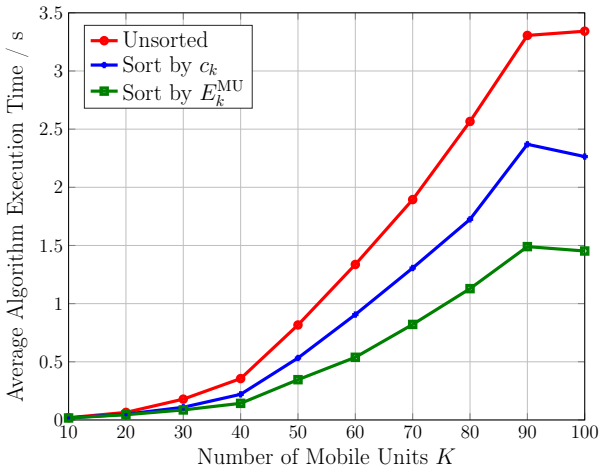

Fig. 6. Impact of initial sorting of the set of MUs

all parameters of the scenario at one central entity. Therefore, only these two simpler metrics are considered here.

TABLE II
EXECUTION TIME SAVING FOR DIFFERENT NUMBERS K OF MUS

Number K of MUs	10	40	70	100
Sort by Complexity c_k	6.58%	38.07%	31.05%	32.25%
Sort by E_k^{MU}	10.44%	59.58%	56.66%	56.54%

VI. CONCLUSION

A computation offloading problem in a MEC scenario with multiple APs is modeled. Since the proposed central energy minimization problem is a MINLP and not efficiently solvable by standard solvers, this formulation is used as a basis for a distributed game theoretic approach. By separating the offloading decisions and the resource allocation strategy, two subproblems are created and used in a fast iterative game theoretic algorithm. In numerical simulations, the proposed resource allocation strategy is tested on the JPBR algorithm and outperforms the reference allocation scheme of JPBR. The proposed resource allocation is then used for the in the developed energy minimization algorithm under a maximum offloading time constraint. This algorithm can achieve an offloading gain not far from the gain of the optimal solution by exhaustive search, while having a considerably lower computational complexity. The relaxation of the time constraint and the introduction of sorting strategies are further modifications to the algorithm with a significant impact on the results.

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] 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.
[2] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer Magazine*, vol. 43, no. 4, pp. 51–56, 2010.
[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] 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.
[5] 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.
[6] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," Tech. Rep. C11-738429-01, 2019.
[7] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. of the IEEE INFOCOM 2012*. IEEE, 2012, pp. 2716–2720.
[8] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Joint allocation of computation and communication resources in multiuser mobile cloud computing," in *Proc. of the IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2013, pp. 26–30.
[9] 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.
[10] 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.
[11] 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.
[12] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, "A double-auction mechanism for mobile data-offloading markets," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 5, pp. 1634–1647, 2015.
[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] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," in *Proc. of the IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.
[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.