# Copyright Notice

# Correlation-Based Device Energy-Efficient Dynamic Multi-Task Offloading for Mobile Edge Computing

Siqi Zhang, Na Yi and Yi Ma
Institute for Communication Systems, University of Surrey, UK, GU2 7XH
e-mails: {s.zhang, n.yi, y.ma}@surrey.ac.uk

***Abstract*—Task offloading to mobile edge computing (MEC) has emerged as a key technology to alleviate the computation workloads of mobile devices and decrease service latency for the computation-intensive applications. Device battery consumption is one of the limiting factors needs to be considered during task offloading. In this paper, multi-task offloading strategies have been investigated to improve device energy efficiency. Correlations among tasks in time domain as well as task domain are proposed to be employed to reduce the number of tasks to be transmitted to MEC. Furthermore, a binary decision tree based algorithm is investigated to jointly optimize the mobile device clock frequency, transmission power, structure and number of tasks to be transmitted. MATLAB based simulation is employed to demonstrate the performance of our proposed algorithm. It is observed that the proposed dynamic multi-task offloading strategies can reduce the total energy consumption at device along various transmit power versus noise power point compared with the conventional one.**

***Index Terms*—Task offloading, device energy efficiency, MEC, correlation, task splitting.**

## I. Introduction

With the continuous development of mobile communication technology and the rapid development of mobile Internet, mobile terminals represented by smart phones, tablet computers, laptop, and smart assistants have been widely used. But the mobile terminal receives limiting factors such as volume, weight, performance, power, etc. Its working ability is still in a serious and tedious state, which cannot meet the increasing demand of people. Although the mobile terminal has made great progress in hardware technology (for example, the continuous replacement of CPU/GPU, the continuous improvement of chip manufacturing process from 28nm to 14nm to the current 7nm, 5nm [1], etc.), but it is still far from what people need. Moreover, with the emergence of new concepts such as autonomous driving, telemedicine, and Industry 4.0 which need ultra reliability, low latency [2], ordinary equipment is even more unable to support their operations. Meanwhile, with the emergence of machine learning, artificial intelligence and other emerging technologies [3], the rapid development of image recognition, speech recognition and other applications, virtual reality and augmented reality game applications are emerging in endlessly. The operation of these applications requires a large amount of computing resources and storage resources, and they are all computationally intensive applications at a time. Due to the limitation of mobile terminals or some other devices, when computationally intensive applications [3] are running on smart terminals, the endurance of the terminal and the performance of the application are very problematic. How to solve this problem of resource limitation and energy consumption has become a huge challenge today.

Most of the literature suggests to change the task allocation method, transmission power, clock frequency to optimize the task offloading algorithm. In [4]– [5], authors proposed task splitting, which is a way to change the task structure to reduce the latency and improve the local device energy efficiency. However, they only split the task, but did not consider the redundancy of sources. In this paper, we propose to split task to the smallest executable task, named as unit, and then select the unit by using the correlation between them. Furthermore, both time and task domain correlation is considered in our work to selected the necessary tasks to be processed.

## II. Problem Formulation and System Models

In this section, the definition of task and unit will be introduced first to help to understand the proposed models. Both task and unit are the process of collecting data, processing data, and sending instructions, but unit is the smallest section that can form a task, and unit cannot be divided anymore. That means task can be composed of one or more units, and a task can be split into one or more units.

Consider $N$ devices (users), denoted by a set of $\mathbb{N} = \{1, 2, ...., N\}$, and device $i$ has $M_i$ tasks at the same time, $i \in \mathbb{N}$, denoted by a set of $\mathbb{M} = \{M_1, M_2, ...., M_N\}$, each task in the $M_i$ tasks is composed of $M_s$ different unit, denoted by a set of $\mathbb{K} = \{k_{M_1}, k_{M_2}, ...., k_{M_s}\}$and $i \in \mathbb{N}$. After some tasks are split, some identical units may be generated, so the correlation of task domain came into being. In our work, it is proposed to use the correlation between units to improve the energy-efficient of local device and reduce the latency. Moreover, it is assumed here that devices communicate with MEC server orthogonally. The main target here is to improve the energy efficiency of local devices when fulfil the extremely severe latency requirements of each unit and each device.

The energy cost minimization problem is formulated as:

$$OPT-1 \qquad \min_{\mathcal{A},\ \mathcal{F},\ \mathcal{P}} \sum_{n=1}^{n=N} E_n^L \tag{1}$$

Subject to

$$\begin{aligned} C1: LT^m_{n,j} &\le T_{n,j\ max}, \quad j \in A, n \in \mathbb{N} \\ C2: LT^l_{n,k} &\le T_{n,k\ max}, \quad k \in B, n \in \mathbb{N} \\ C3: LT_n &\le T_{n\ max}, \quad n \in \mathbb{N} \\ C4: f_{n,l} &\le f_{n,l\ max}, \quad n \in \mathbb{N} \\ C5: p^t_n &\le p^t_{n\ max}, \quad n \in \mathbb{N} \end{aligned} \tag{2}$$

$E^L_n$ represents the local energy consumption of user $n$, $LT^m_{n,j}$ represents the latency of unit $j$ of user n processed on MEC, $LT^l_{n,k}$ represents the latency of unit k of user $n$ processed on local device, $T_{n,j\ max}$ represents the latency requirement of unit $j$, $A$ represents the collection of all tasks processed on the MEC, and $B$ represents the collection of all units processed on the local device, $LT_n$ represents the latency of user $n$, $T_{n\ max}$ represents the latency requirement of user $n$, $f_{n,l}$ represents the computation capability of user $n$, such as the number of CPU cycles per second, $p^t_n$ represents the transmission power of user $n$, and $f_{n,l\ max}$, $p^t_{n\ max}$ indicate the maximum value of $f_{n,l}$ and $p^t_n$ respectively.

$\mathcal{F} = \{f_{n,l} | n \in \mathbb{N}\}$, $\mathcal{P} = \{p^t_n | n \in \mathbb{N}\}$, $\mathcal{A} = \{A_{n,j} | n \in \mathbb{N}, j \in \mathbb{M}2\}$, C1 and C2 are to limit the latency of each task to not exceed the requirements, C3 is to limit the user's overall latency does not exceed the requirements, C4 is to limit the processing power of the local device not to exceed its maximum processing power, C5 is to limit the transmission power of the local device not to exceed its maximum transmission power.

### A. Transmission Model

When the interference is not considered, the signal-to-noise ratio (SNR) of user n is

$$SNR_n = \frac{p^t_n h^2_{n,m}}{B_w \mathcal{N}_0} \tag{3}$$

and then, the transmission rate (uplink) of user $n$ is calculated as:

$$r_n = Blog_2(1 + SNR_n). \tag{4}$$

In (3) and (4), $B_w$ represents the bandwidth of this channel, $h_{n,m}$ represent the channel gain of user n to MEC, and $\mathcal{N}_0$ represents the noise spectral density, $p^t_n$ represents the transmission power of user $n$. Because the amount of data that needs to be offloaded is much larger than the amount of data that needs to be downloaded, so the downlink is ignored in this paper [6], [7].

### B. Computation Model

Let $w_{n,j}$ represent the CPU cycle required to calculate the unit (unit $j$ of user $n$), $d_{n,j}$ represents the computation input data (in bits) of the $j$ th unit of $n$ th user of local devices.

*Local Computing:* when user chooses to process locally, use $f_{n,l}$ to represent the CPU clock speed of user's device, so the latency in local for computing can be represented as:

$$t^l_{n,j} = \frac{w_{n,j}}{f_{n,l}}. \tag{5}$$

where $t^l_{n,j}$ denotes the time required to complete unit $j$ of user $n$ in local device.

The energy required to complete unit $j$ in local device can be expressed by the following formula

$$E^l_{n,j} = \kappa w_{n,j} f^2_{n,l} \tag{6}$$

In this case, $\kappa$ is the effective switched capacitance depending on the chip architecture [8].

*1) MEC Computing:* when a user decides to offload tasks to MEC, and transmit a unit through a wireless network, the corresponding transmission latency and energy consumption will be generated. According to the communication model, the uplink transmission latency when user $n$ offload the task is:

$$t^t_{n,j} = \frac{d_{n,j}}{r_n}, \tag{7}$$

and the energy required to transmit unit $j$ can be expressed by the following formula:

$$E^t_{n,j} = P^t_{n,j} t^t_{n,j}. \tag{8}$$

where $t^t_{n,j}$ represents the time required to transmit the $j$ th unit of $n$ th user. After the computing unit is offloaded to the MEC, the MEC will allocate certain computing resources to this unit, considering that the computing resources allocated by the MEC to each user are fixed. Let $f_{MEC}$ denote the computing resources allocated by the MEC, and the latency for the MEC to perform the $j$ th unit of $n$ th user is

$$t^m_{n,j} = \frac{w_{n,j}}{f_{MEC}}. \tag{9}$$

Because MEC has a constant energy supply, MEC's energy consumption need not be considered in this paper.

In this paper, it is assumed that MEC can only process one unit at the same time, communication capacity can only support the transmission of a unit simultaneously, as shown in Fig. 1 and Fig. 2. Blue blocks represent transmission time, green ones represent computing time, black ones and yellow ones represent queuing for transmission, and queuing for computing respectively At the beginning of this section, it is introduced that each user generates multiple tasks and then they are split into multiple units, and needs to be processed. However, due to channel and processor limitations, these units cannot be processed at the same time. There are two definitions of waiting time for the units which will be processed in local and waiting time for the units which will be processed in MEC. Suppose A is the set of units which will be processed in MEC, and A=$\{$1,2, ...,a$\}$, B is the set of units which will be processed in local device, and B=$\{$1,2, ...,b$\}$,

**Definition 1** (waiting time for the units which will be processed in local): The sum of all time, before the unit is

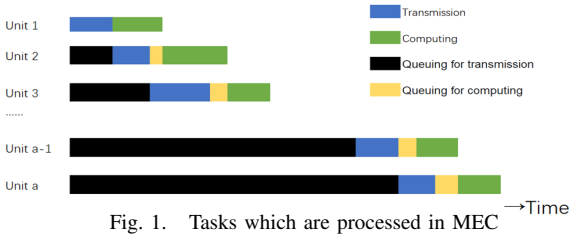



Fig. 1. Tasks which are processed in MEC

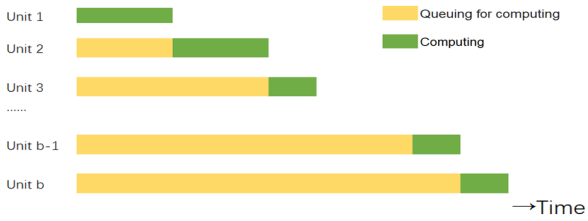



Fig. 2. Tasks which are processed in local

processed by local device. Use $WT^l_{n,k}$ to present the waiting time of unit $k$ of device $n$, which will be processed in local.

**Definition 2** (waiting time for the units which will be processed in MEC): The sum of all time, before the unit is processed in MEC. Use $WT^m_{n,j}$ to present the waiting time of unit $j$ of device $n$, which will be processed in MEC.

According to the Definition 2, waiting time for the units which will be processed in MEC is divided into two parts, time of waiting for transmission plus time of transmission ($WT3^m_{n,j}$), and time before processing in MEC and after MEC receives the unit($WT4^m_{n,j}$), that means $WT^m_{n,j}$ can be divided into $WT3^m_{n,j} + WT4^m_{n,j}$.

Assume that $unit_j \in A$
$when\ j = 1$

$$WT3^m_{n,j} = t^t_{n,j}, \qquad WT4^m_{n,j} = 0 \tag{10}$$

$when\ j > 1$

$$\begin{aligned} WT3^m_{n,j} &= WT3^m_{n,j-1} + t^t_{n,j} \\ WT4^m_{n,j} &= max(WT3^m_{n,j}, LT^m_{n,j-1}) - WT3^m_{n,j} \end{aligned} \tag{11}$$

In (11) , max means the maximum value between these two value.

$$\begin{aligned} WT^m_{n,j} &= WT3^m_{n,j} + WT4^m_{n,j} \\ LT^m_{n,j} &= WT_{n,j} + t^m_{n,j} \end{aligned} \tag{12}$$

$LT^m_{n,j}$ means the latency of $unit_j$, $unit_j \in A$

Assume that $unit_k \in B$
so, in this case

$$WT^l_{n,k} = \sum_{c=1}^{c=j-1} t^l_{n,c} \tag{13}$$

so that, the latency of unit $k$ can be expressed as ($unit_k \in B$):

$$LT^l_{n,k} = \sum_{c=1}^{c=k-1} t^l_{n,c} + t^l_{n,k} = \sum_{c=1}^{c=k} t^l_{n,c} \tag{14}$$

So the latency of the whole system of user $n$ can be represent as:

$$Ts_n = max(LT^l_{n,b}, LT^m_{n,a}) \tag{15}$$

Use $E^L_n$ to represent the total energy consumption of local device $n$:

$$E^L_n = \sum_{a=1}^{a=j} E^t_{n,a} + \sum_{b=1}^{b=k} E^l_{n,b} \tag{16}$$

## III. Proposed Task Offloading Algorithm

In this section, first, several preprocessing methods will be proposed, and perform some operations before task offloading, without affecting the reliability of the task, reduce the amount of task calculation and transmission, and then introduce the algorithm proposed for task offloading.

### A. Correlation in Time Domain of Tasks

With the change of time, the task constantly updates its own information source to process the task. In the traditional task offloading, only consider how to change the task allocation mode, the processing frequency, transmission power to reduce latency and improve the energy-efficient, but in their algorithm, the data size does not change, which to a large extent, restricts the development of task offloading, this paper provide a new way to filter some unnecessary information to reduce the data size, so as to further reduce the latency and energy consumption. Correlation coefficient is a good way to be considered.

The correlation coefficient introduced in [9] is defined as:

$$r(x,y) \triangleq \frac{\mathsf{cov}(x,y)}{\sqrt{\mathsf{var}(x)\mathsf{var}(y)}}\ ,\ r(x,y) \in [-1,1]. \tag{17}$$

In this formula, x, y are the sources of information which need to be compared. Suppose there is a task, denoted as $Task\ A$, and denote the names of $Task\ A$ at different moments as $Task\ A1$, $Task\ A2$, $Task\ A3$, $Task\ A4$ in the order of time, $A1$ is the first and $A4$ is the last one, to reduce the number of processing, correlation coefficient should be used between them to decide whether to process these tasks, or process part of them. There are two ways to use the correlation coefficient.

The first way, set only one threshold to decide whether to process this task: set a threshold $\alpha$, and then process the $Task\ A1$, and then calculate then correlation coefficient of $Task\ A1$ and $Task\ A2$. If the correlation coefficient between them is greater than $\alpha$, the $Task\ A2$ will not be processed, and keep the $Task\ A1$ in memory, and the correlation coefficient between $Task\ A1$ and $Task\ A3$ should be calculated, if it is greater than the $\alpha$, $Task\ A3$ will not be processed too, and continue to calculate the correlation coefficient between $Taks\ A1$ and $Task\ A4$, if not, $Task\ A3$ will be processed, and keep $Task\ A3$ in memory, and then calculate the correlation coefficient between $Task\ A3$ and $Task\ A4$, and repeat with following Tasks.

The second way, set multi-threshold to decide how to process this task: two thresholds as an example. First of all, set two thresholds, $\alpha$ and $\beta$, and $\alpha > \beta$, and process the first task. When the correlation coefficient between two adjacent tasks is greater than $\beta$ and smaller than $\alpha$, just need to process the different part of this two tasks, when the correlation coefficient

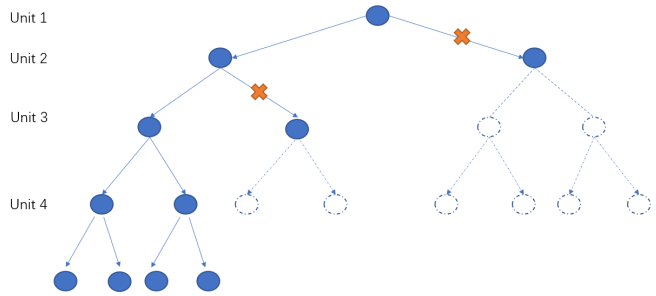



Fig. 3. Binary Tree

between two tasks is smaller than $\beta$, it can be considered the information of these two tasks are totally different, so, the new task need to be processed, when the correlation coefficient between two tasks is greater than $\alpha$, it can be considered the information of these two tasks are totally same, the new task need not to be processed again

### B. Correlation in Task Domain of Tasks

The correlation coefficient is considered in time domain, so as to reduce the amount of data, then the next step is to do the task splitting to get the units of these necessary task. As said before, different tasks may split into the same unit, so the correlation of task domain came into being. How to use these correlation to further optimize the algorithm is the main content of this part, and use $C_{o,p}$ to represent the correlation between unit o and unit p. $C_{o,p} \in \{0, 0.5, 1\}$. when $C_{o,p} = 0$, that means that there is no relationship between unit o and unit p, both units need to be processed, when $C_{o,p} = 1$, that means unit o and unit p are total same, and only need to process one of them and share the result, when $C_{o,p} = 0.5$, that means unit o and unit p are different units, but use the same source information.

### C. Units Allocation

After arranging units in the order required by the latency, determine whether this unit needs to be assigned to the MEC for processing from the first unit. At this time, the tree diagram needs to be drawn, as shown in Fig. 3, use four units as an example. Starting from the vertex first unit, there are two paths, one (unit will be processed in MEC Server) and zero (unit will be processed in local device). If the unit latency caused by the path meets the requirements of this unit, this path will be maintained and the next level of judgment will be made. If the latency does not meet the requirements, then drop this path. Until finish the last task, and judgment whether the system latency is meet the requirement, if the answer is YES, keep this node, if not, drop this node, then all the feasible solutions can be obtained. Then, for getting the optimal solution, some other optimization to transmission power and clock frequency need to be done, and they will be introduced in the next few parts. In the above situation, there is no correlation between units, the next section will introduce the situation with there are correlation between units. when $C_{o,p} = 1$, the duplicate units should be deleted, and then allocate the remaining units in the same way as said before. when $C_{o,p} = 0.5$, these units with a correlation degree of 0.5 need to be merged, that means merge them into a large task to carry out the process of task allocation(this allocation process is same as said before). If this large task is allocated to MEC server for processing, it will reduce the latency and energy consumption caused by communication, that because they use the same source information. If this large task is allocated to local device for processing, the energy consumption and the latency will not change.

### D. Clock Frequency Allocation

According to (6), if $w_{n,j}$ (the CPU cycle required to calculate the unit) not change, the smaller the value of $f_n^t$, the lower energy consumption will take.

### E. Transmission Power Allocation

According to (3), (4), and (8) and use $D$ to represent the total transmission volume of all units that need to be transmitted to MEC, so the energy consumed for transmission becomes E1

$$E1 = \frac{D}{B_w} \times log_\eta 2^{p_n^t} \tag{18}$$

$\eta = 1 + \frac{p_n^t h_{n,m}^2}{B_w \mathcal{N}_0}$. So, the monotonic interval of E1 with respect to $p_n^t$, and take the $p_n^t$ value that minimizes E1 under conditions C1 to C5.

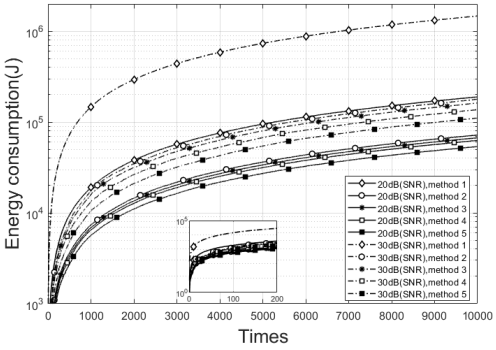
### F. Algorithm Flow

Use the maximum $p_n^t$ of the local device and maximum $f_n^t$ of local device to find all combinations that satisfy the conditions of C1-C5, and record them as $\mathcal{V}$, and then use the algorithms proposed before find the smallest value of energy consumption that can be achieved after optimization in $\mathcal{V}$ , take the combination with the least energy consumption as the final result. Therefore, the optimal clock frequency, transmission power, and unit allocation method can be obtained.

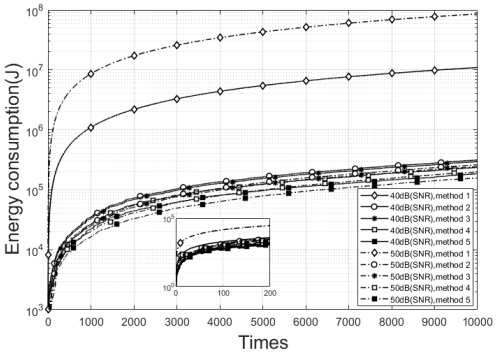
## IV. Simulation Results

In this section, we evaluate our proposed task-offloading algorithm with three state-of-the-art baselines through MATLAB based simulations. It is considered in our simulation, that system contains four users and a single MEC. Each user has different number of tasks, and different type of task has different size. Each user has its dedicated orthogonal channel to communicate to MEC.

The channel bandwidth is set to 20MHZ, and the channel conforms to the Rayleigh distribution. The SNR of the user terminal is set to 10dB, 20dB, 30dB, 40dB and 50dB in our simulations. The maximum computing rate in user's device is $2 \times 10^9$cycles/s [10], and the maximum computing rate in MEC server is $20 \times 10^9$cycles/s [10], the size of each task is between 1-3M, and the number of processing revolutions they require is Between 1500-4500 cycles [10], and $\kappa = 10^{-11}$ [8]. There are two latency requirements, which are 50ms and 100ms.




(a) SNR=20,30dB




(b) SNR=40,50dB

Fig. 4. Energy consumption of five different algorithms

In the experiment, two proposed approaches simulated with the other three baselines. Method 1 (Baseline 1): the most primitive task offloading algorithm, without considering any transmission power and other optimizations. Method 2 (Baseline 2): Further optimize the processing frequency and transmission power based on the Method 1 [6]. Method 3 (Baseline 3): Based on Method 1 and Method 2, consider the impact of split tasks on task offloading [4]. Method 4: Consider the relevance of task on time domain to reduce the amount of data. method 5: Base on Method 3, consider the correlation in time domain and in different tasks to reduce the amount of data.

The Fig. 4(a) Fig. 4(b) show the energy consumption of the five different algorithms on the local device when the SNR on the user terminal are 20dB, 30dB, 40dB and 50dB respectively. In the same SNR, our proposed algorithms have better performs Fig. 5 shows the probability of task failure under different methods and SNR. From Fig. 4, and Fig. 5, with the same SNR, our proposed algorithms have better performs in energy saving and reliability, and as the increase of SNR, with the same method, the energy cost may increase too, this is because we record that the energy consumption of the failed tasks that can be known in the stage of decision making is zero. The larger the SNR, the more tasks that can be processed, so it will also cause more energy consumption.

## V. Conclusion

In this paper, it was studied how to improve the energy efficiency of local devices and reduce the latency in the process

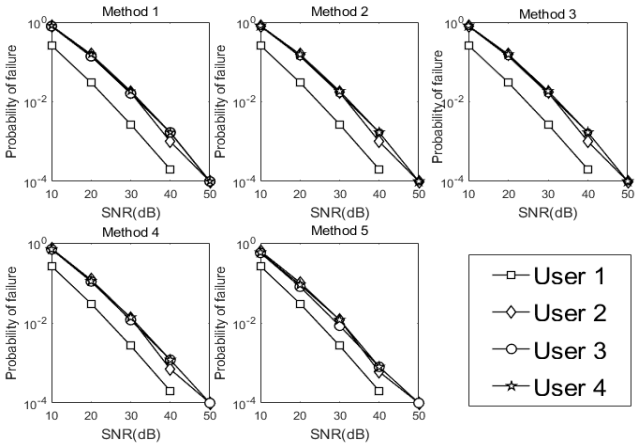



Fig. 5. Probability of task processing failure under different methods and SNR

of task offloading. A novel task offloading algorithm has been proposed with the consideration of task correlation in both time domain and task domain to reduce the number of tasks to be transmitted. Moreover, a joint optimization has been studied for task allocation, transmission power and clock frequency. MATLAB based simulation results have demonstrated that the proposed task offloading algorithm can reduce the device energy consumption along various transmit power versus noise variance setup compare with the conventional one.

## Acknowledgment

The work was supported in part by European Com- mission under the framework of the Horizon2020 5G-Drive project, and in part by 5G Innovation Centre (5GIC) HEFEC grant.

## References

[1] K. Choi et al., "Enhanced reliability of 7nm process technology featuring euv," in *Proc. 2019 Symposium on VLSI Technology*, 2019, pp. T16–T17.

[2] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, Feb. 2017.

[3] L. K. Nandhini and S. Udhayakumar, "Service offloading of computationally intensive processes using hadoop image processing interface in private cloud," in *Proc. 8th International Conference on Advanced Computing (ICoAC)*, 2017, pp. 201–205.

[4] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12 825–12 837, Feb. 2018.

[5] C. Liu et al., "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Feb. 2019.

[6] S. Guo et al., "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[7] K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, Aug. 2016.

[8] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," *HotCloud*, vol. 10, no. 4-4, p. 19, Jun. 2010.

[9] K. M. Bisheh, B. Zakeri, and S. M. H. Andargoli, "Correlation coefficient estimation for stochastic FDTD method," in *Proc. 7th International Symposium on Telecommunications (IST)*, 2014, pp. 234–238.

[10] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Apr. 2017.