Maximizing Mobiles Energy Saving Through Tasks Optimal Offloading Placement in two-tier Cloud

Houssemeddine Mazouzi  
University Paris 13, Sorbonne Paris  
Cité  
Villetaneuse, France  
mazouzi.houssemeddine@univ-paris13.fr

Nadjib Achir  
University Paris 13, Sorbonne Paris  
Cité  
Villetaneuse, France  
nadjib.achir@univ-paris13.fr

Khaled Boussetta  
University Paris 13, Sorbonne Paris  
Cité  
Villetaneuse, France  
khaled.boussetta@univ-paris13.fr

ABSTRACT
In this paper, we focus on tasks offloading over two tiered mobile cloud computing environment. We consider several users with energy constrained tasks that can be offloaded over cloudlets or on a remote cloud with differentiated system and network resources capacities. We investigate offloading policy that decides which tasks should be offloaded and determine the offloading location on the cloudlets or on the cloud. The objective is to minimize the total energy consumed by the users. We formulate this problem as a Non-Linear Binary Integer Programming. Since the centralized optimal solution is NP-hard, we propose a distributed linear relaxation heuristic based on Lagrangian decomposition approach. To solve the subproblems, we also propose a greedy heuristic that computes the best cloudlet selection and bandwidth allocation following tasks’ energy consumption. We compared our proposal against existing approaches under different system parameters (e.g. CPU resources), variable number of users and for six applications, each having specific traffic pattern, resource demands and time constraints. Numerical results show that our proposal outperforms existing approaches. We also analyze the performance of our proposal for each application.

CCS CONCEPTS
• Networks → Cloud computing; • Human-centered computing → Ubiquitous and mobile computing systems and tools; • Theory of computation → Network optimization;

KEYWORDS
computation offloading; mobile cloud computing; mobile edge computing; cloudlet; Lagrangian decomposition.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:
Houssemeddine Mazouzi, Nadjib Achir, and Khaled Boussetta. 2018. Maximizing Mobiles Energy Saving Through Tasks Optimal Offloading Placement in two-tier Cloud. In MSWiM ’18: 21st ACM Int’l Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems, Oct. 28–Nov. 2, 2018, Montréal, Québec, Canada. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3242102.3242133

1 INTRODUCTION
In recent years, mobile devices have undergone a major transformation, moving from small devices with limited capabilities to important everyday accessories with important capabilities. This recent advances in hardware and software mobile technology have led to an exponential growth in mobile application markets. Unfortunately, even as mobile applications become more and more intensive, the computing power of mobile devices remains limited compared to what we can find in datacenters and cloud. Furthermore, because the limited weight and size and therefore the life of the battery, a powerful approach to improving the performance of mobile applications and reducing the shortage of mobile device resources is required. One possible approach is to enable mobile devices to offload some of their intensive workloads to remote high-performance virtual machines. Unfortunately, even though clouds have rich computing and storage resources, they are generally geographically far away from users. In this case, this approach may suffer from significant and fluctuating delays on the Internet. This finding is particularly problematic for some mobile applications, such as augmented reality or cloud gaming, which require a reduced response time.

To reduce this long access delay, an emerging tendency is to push the cloud to the network edge, mainly located within existing wireless Access Points (APs), ADSL box or Base Stations (BSs). This proximity gives the opportunity to users to offload their tasks to this Edge cloud or Cloudlets. This new paradigm is known as ‘Mobile Edge Computing (MEC)’. A Cloudlet can be seen as small data center, and because of this geographical proximity between users and cloudlets, the access delay on the task offloading can be greatly reduced, compared to remote clouds, and thus significantly improving user experiences. In this paper, we focus on multi-user MEC, where users offload their applications to edge servers or cloudlets. In this case, both the wireless bandwidth and the computing resources must be shared among the users.

This paper presents a new computation offloading policy for multi-user multi-cloudlet environment, named Efficient Cloudlet Selection Offloading policy (ECESCO). Our objective is to determine which users should be offloaded and to which cloudlet in order to minimize the overall energy consumed by the users.

Basically, ECESCO assigns each user to the best cloudlet in order to reduce the energy consumption of all users, according to the
network and system resources. We formulate this problem as a Binary Integer Programming (BIP) and propose a distributed linear relaxation heuristic based on Lagrangian decomposition approach. Our policy consists of two decision levels. The local offloading decision level that concerns the users associated with the same access point, in order to solve the offloading subproblem of this access point. The global offloading decision level ensures that the offloading solution given by the local offloading decision level complies with the cloudlet resource constraints. A greedy heuristic was proposed to solve the local offloading decision subproblem and tries to offload each task to the best available cloudlet.

The rest of the paper is organized as follows: Section 2 presents related work, and section 3 describes the modeled system. Problem formulation and solving are detailed in Section 4. Performance evaluation of our proposed policy is discussed in section 5. Finally, a conclusion is drawn in Section 6.

2 RELATED WORK

Several works were proposed to explore computation offloading in order to improve the performance of the mobile devices. Some work focused on the wireless bandwidth allocation in order to take offloading decision, such as Meng-Hsi Chen et al. [3], Xu Chen et al. [4], Songtao Guo et al. [7], and Keke Gai et al. [5]. The work presented by Meng-Hsi et al. is one of the first works supporting multi-user computation offloading in mobile cloud computing. It decides which task must be performed in the remote cloud and which task must be performed locally. Then, it allocates the wireless bandwidth to each offloaded task in order to reduce the energy consumption of the mobile device. Xu Chen et al. policy was designed to a single cloudlet mobile-edge environment. Each user tries to offload its tasks, accordingly with the available wireless bandwidth to reduce the energy consumption. Another offloading approach for multi-users was presented by Songtao Guo et al. The offloading policy decides which tasks should be offloaded and allocates the wireless bandwidth to each offloaded task. Then, it allocates the local processor frequency. Lastly, Keke Gai et al. propose a scheduler to assign the tasks between the local mobile device and the remote cloud in order to save energy consumption. In a multi-cloudlet scenario, the computational capacity of each cloudlet is limited, unlike these heuristics, the computation offloading policy must select the best cloudlet to each user with the purpose of achieving high performance.

More recently, many works focus on cloudlets placement heuristics in the MEC environment. The main goal is to find how many cloudlets are needed and where place them, such as Mike Jia et al. [8, 18], Hong Yao et al. [19], and Longjie Ma et al. [11]. Mike Jia et al. heuristic tries to find the best cloudlets placement in a large network, then select a cloudlet to perform the computation tasks of each access point. The K-median clustering based on users density is used to place the cloudlets. Then each access point is statically assigned to a cloudlet. Similarly, Hong Yao et al. was designed to support heterogeneous cloudlets environment. Finally, Longjie Ma et al. was introduced to find the minimal number of cloudlets that must be placed to improve the user experience quality. However, the density of mobile-users are dynamic and changes over time. So, static assignment of the access points to cloudlets may decrease the performance of the computation offloading. To confirm this assumption, our heuristic ECESO consider dynamic cloudlet selection and the wireless bandwidth allocation with the aim of minimizing energy consumption of mobile devices.

Anwesha Mukherjee et al. [12, 14] and Mike Jia et al. [9] focus on the dynamic cloudlet selection in order to reduce the offloading cost. Anwesha Mukherjee et al. designed a multi-level offloading policy to optimize the energy consumption. The users offload to the nearest cloudlet in the first step. According to the resource availability in this cloudlet, it can perform the tasks or offload the task to another cloudlet, and so on. Mike Jia et al. introduced a heuristic to balance the load between the cloudlet. Its main goal is to migrate some tasks from overloaded cloudlets to underloaded cloudlets to reduce the execution time. These works propose dynamic cloudlet selection heuristics, but they do not consider the wireless bandwidth in a multi-user environment.

The presented policies focus on reducing the offloading cost. They try to offload the tasks to a predetermined offloading server, the remote cloud or a cloudlet. Consequently, the performance of computation offloading can be decreased due to the dynamic density of users in such environment. Therefore, designing a new offloading policy is mandatory. The new policy must consider dynamically many offloading servers for whom a user can offload its tasks, and compute optimal task placement in two-tier MEC environment.

3 SYSTEM DESCRIPTION AND MODELING

3.1 MEC Environment

We consider the mobile edge computing environment (MEC) illustrated in Fig. 1. The infrastructure is composed of M access points, K deployed cloudlets and one remote cloud. In the remainder, we will refer to the cloud as the (K + 1)th offloading server. Similarly to [18], we assume that the number of cloudlets is less than the number of access points (K < M).

Let M denote the set of access points. The remote cloud and the cloudlets constitute a set, denoted K, of offloading servers. All these servers offer computation resources to perform offloaded tasks.
tasks. In each server $k \in \mathcal{K}$, the resources are characterized by a fixed capacity, denoted $F_k$, of computational resource units. A computational resource unit is expressed in Ghz and is defined as the number of cycles per second allocated to perform a task. We denote $c_k$ the number of cycles per second allocated by server $k$ to perform any given offloaded task. Similarly to [4, 7], we consider that $c_k$, $\forall k \in \mathcal{K}$, is fixed and does not change during the computation. We also assume that the number of computation resources units, $F_k$, is limited in cloudlets while it is infinite in the cloud. Formally, $\forall k \in \{1, 2, ..., K\}, F_k \ll F_{K+1} = \infty$.

3.2 Tasks requirements

The system is observed at a given time. Extension to continuous observation time is possible by discretizing the time into contiguous observation intervals. At observation time, we assume that each access point $m \in M$ is associated to a set of users, denoted $N_m$. Let $N_m$ be the size of the above set.

Let $u_{m,n}$ denote the $n^{th}$ user associated with the $m^{th}$ access point. At observation time, we assume that each user $u_{m,n}, \forall (m,n) \in (M,N_m)$, have exactly one task, denoted $r_{m,n}$. This task is characterized by the number, denoted $Y_{m,n}$, of CPU cycles needed for its computation. The purpose of our work is to decide if a task should be performed locally, on the user’s terminal or remotely in one of the $\mathcal{K}$ servers. To this end, and similarly to [6], we distinguish two offloading decision tasks:

1. The static offloading decision task
2. The dynamic offloading decision task

In the first category, the tasks are always offloaded. The offloading decision is taken at the application design and these tasks must be performed remotely on the cloudlets/cloud either because they require some specific hardware/software environment (e.g. GPU, operating systems) that is not available on the user’s terminal or in order to fulfill some application’s constraints (e.g. security issues). Regarding this category, the purpose of our proposal is to decide in which server $k \in \mathcal{K}$ a task must be offloaded.

In the second category the offloading decision is taken at runtime. These tasks can be executed either locally or offloaded in one of the $k \in \mathcal{K}$ servers. Regarding this category, the purpose of our proposal is to provide a policy that decides if a dynamic offloading decision task has to be executed locally (on the user’s device) or if it is optimal to offload it to a specific server $k \in \mathcal{K}$.

To differentiate these two categories, we associate to each task $r_{m,n}$ a binary variable $y_{m,n}$, which is equal to 0 if $r_{m,n}$ is a dynamic offloading decision task and otherwise 1 if $r_{m,n}$ is a static offloading decision task.

3.3 Local Processing time

If task $r_{m,n}$ is performed locally, then the user’s terminal computational capabilities are employed. However, the transmission is not required. We assume that the user’s device can allocate at observation time a computational capacity, denoted $f_{m,n}$, to perform locally the task. This quantity is expressed as a number of cycles per seconds. We can thus derive the local processing time of task $r_{m,n}$ as follow:

$$T_{m,n}^l = \frac{Y_{m,n}}{f_{m,n}}$$  (1)

3.4 Remote Processing time

If task $r_{m,n}$ is offloaded on server $k$ then the following steps are executed:

1. In order to perform task $r_{m,n}$ remotely on server $k$ a certain amount of data is transmitted from the user’s terminal to server $k$. We denote $u_{up,m,n}$ the quantity of uploaded data. This quantity is expressed in bits. The transmission passes first through the wireless access link that connect the user’s terminal to the access point $m$ and then on the backhaul network that relay the access point $m$ to the server $k$. Let $B_{up,m,n}$ be the allocated bandwidth at observation time to transmit the input data of offloaded task $r_{m,n}$ from user $u_{m,n}$ to its access point $m$. We will detail in section 3.5 how to compute this quantity when a given number of users associated to a same access point must offload their tasks at observation time. We can then derive the transmission time to upload the data of an offloaded task $r_{m,n}$ from user $u_{m,n}$ terminal to its access point $m$ as follow:

$$T_{m,n}^{up} = \frac{u_{up,m,n}}{B_{up,m,n}}$$  (2)

Similarly, let $B_{bh,m,k}$ denotes the end to end backhaul bandwidth between the access point $m$ and the offloading server $k$. We can express the transmission time of the data associated to an offloaded task $r_{m,n}$ from access point $m$ to server $k$ as follow:

$$T_{m,n}^{bh,k} = \frac{u_{up,m,n}}{B_{bh,m,k}}$$  (3)

(2) This uploaded data is then used by the offloaded server $k$ to perform the task. The remote computation time of task $r_{m,n}$ can be expressed as the ratio of the CPU cycles required for the task’s computation ($Y_{m,n}$) to the number of cycles per second allocated at server $k$ ($c_k$):

$$T_{m,n}^{ce,k} = \frac{Y_{m,n}}{c_k}$$  (4)

(3) Finally, when the task’s computation is finished, the server $k$ returns back the results to the user’s terminal. We denote $d_{m,n}$ the amount of the results data that is returned back to the user $u_{m,n}$. This quantity is also expressed in bits. We can thus derive the transmission time of task $r_{m,n}$ remote computation result from server $k$ to access point $m$ as follow:

$$T_{m,n}^{bh} = \frac{d_{m,n}}{B_{bh,m,k}}$$  (5)

In the same way, We can also express the transmission time to download the remote computation result of the offloaded task $r_{m,n}$ from access point $m$ to user $u_{m,n}$ terminal as follow:

$$T_{m,n}^{bw} = \frac{d_{m,n}}{B_{bw,m,n}}$$  (6)
Here, $B_{m,n}^\text{up}$ denotes the allocated bandwidth to transmit the offloaded task results from the access point $m$ to user $u_{m,n}$.

We can then compute the completion of task $r_{m,n}$ when the latter is offloaded on server $k$ as follow:

$$T_{m,n}^k = T_{m,n}^\text{up} + T_{m,n}^\text{bh} + T_{m,n}^\text{c} + T_{m,n}^\text{d} + T_{m,n}$$

(7)

### 3.5 Shared wireless access bandwidth

At observation time, the number of transmissions to an access point $m$ depends on the number of tasks that are offloaded. Let $\pi_m$ denote the number of tasks that are offloaded through access point $m \in M$ to a server $k \in K$. Obviously, $0 \leq \pi_m \leq N_m$

The allocated bandwidth to upload and download the offloaded task data depends on the number of concurrent transmissions on the access point. In this paper, we choose to develop the expression of the upload and download allocated bandwidth of an offloaded task assuming that 802.11n technology is used at access point. Adaptation of this work to cellular technologies is possible using existing models from the literature. As we will emphasis in section 20, the latter is offloaded on server $k$ as follow:

$$B_{m,n}^\text{up} = \frac{B(\pi_m)}{\pi_m}$$

(10)

Here, $\tau$ denotes the stationary probability that one mobile device transmits a packet in a slot time. It can be expressed as:

$$\tau = \frac{1}{1 + \frac{1 - p}{2(1 - p)(CW - 1) - (1 - pR^k)}}$$

(11)

Fixed point method can be used to solve equations 10 and 11 and determine the value of $\tau$.

- $E$ is the average time to transmit the packet payload information of size $d$. It can be computed as follow:

$$E = \frac{d}{W} + \frac{CW}{CW - 1}$$

(12)

- $p_b$ denotes the probability that the channel is busy. It can be computed as follow:

$$p_b = 1 - (1 - \tau)^{\pi_m}$$

(13)

- $T_s$ is the average time the channel is sensed busy because of a successful transmission. Let $T_{MPDU}$, $T_{ACK}$, SIFS and DIFS denote the time to transmit the MPDU (including MAC header, PHY header), the time to transmit an ACK, the SIFS time, and the DIFS time, respectively.

$$T_s = (T_{MPDU} + SIFS + T_{ACK} + DIFS) \cdot \frac{CW}{CW - 1} + \phi$$

Here, $T_{MPDU}$, $T_{ACK}$ and DIFS denote the time to transmit the MPDU (including MAC header, PHY header), the time to transmit an ACK, and the DIFS time, respectively.

- $T_c$ is the average time the channel is sensed busy by each station during a collision.

$$T_c = T_{MPDU} + SIFS + T_{ACK} + DIFS + \phi$$

Using equation 9 we can compute the allocated bandwidth to transmit the input data of offloaded task $r_{m,n}$ from user $u_{m,n}$ to its access point $m$ as follows:

$$B_{m,n}^\text{up} = \frac{B(\pi_m)}{\pi_m}$$

(14)

In the same way, we can obtain the allocated bandwidth to transmit the offloaded task results from the access point $m$ to user $u_{m,n}$ as follow:

$$B_{m,n}^\text{d} = \frac{B(\pi_m)}{\pi_m}$$

(15)

### 3.6 Completion time constraint

Let $T_{m,n}$ denotes the total processing time of task $r_{m,n}$. From the above system description and modeling section, one can see that $T_{m,n}$ depends on the processing time and eventually, in case of offloading, upload and download transmission times. Precisely, if a task $r_{m,n}$ is performed locally, then $T_{m,n} = T_{m,n}^k$. Otherwise, if a task $r_{m,n}$ is offloaded on server $k$ the $T_{m,n} = T_{m,n}^k$.
Hence, to integrate applications’ Quality of Services requirements, we associate to each offloadable task a time constraint threshold. Precisely, we define a maximum completion time threshold, denoted \( t_{m,n} \) to any task \( r_{m,n} \), \( \forall (m,n) \in (M, N_m) \).

The completion time is a hard constraint for any task, \( r_{m,n} \), \( \forall (m,n) \in (M, N_m) \). In other words, the optimal offloading policy must satisfy the following constraint:

\[
T_{m,n} \leq t_{m,n}
\]  

(16)

As stated before, the purpose of our offloading policy is to determine which tasks should be offloaded and to which server in order to satisfy the completion time constraint. In addition to this constraint, our purpose is to determine the optimal offloading policy which minimize the overall energy consumed by the users terminals. In the following sections we will detail energy consumption models for both, local and remote tasks processing.

3.7 Energy consumption: local processing

Following the model detailed in [2] of power consumption due to tasks processing on portable devices, we can then derive the total amount of energy consumed to process task \( r_{m,n} \) locally as follows:

\[
Z_{m,n}^l = \kappa \cdot (F_{m,n})^3 + T_{m,n}^l = \kappa \cdot (F_{m,n})^2 + \gamma_{m,n}
\]

(17)

where \( \kappa \) is the effective switched capacitance, which depends on the chip architecture, and is used to adjust the processor frequency. Similarly to [2], we set \( \kappa = 10^{-9} \).

3.8 Energy consumption: Remote Processing

The total amount of energy consumed by the user’s device to perform the task remotely is equal to the energy used when the device 1) turns the radio in the transmission mode to send the data to the remote server, 2) turn the radio in idle mode to wait the task completion and finally 3) turn the radio in the reception mode in order to receive the result data from the remote server. The consumed energy can thus be expressed as follow:

\[
Z_{m,n}^r = P_{m,n}^{tx}x_{m,n}^{\text{up}}(T_{m,n}^r + T_{m,n}^{idle}) + P_{m,n}^{rx}x_{m,n}^{\text{down}}
\]

(18)

where \( P_{m,n}^{tx} \) is the power consumption when the radio interface is in transmission mode, \( P_{m,n}^{rx} \) is the power consumption when the radio interface is in reception mode and \( P_{m,n}^{idle} \) is the power consumption when the radio interface in idle mode. As is commonly assumed [2], we suppose that

\[
P_{m,n}^{idle} \leq P_{m,n}^{rx} \leq P_{m,n}^{tx}
\]

(19)

4 PROBLEM FORMULATION AND SOLVING

As introduced earlier, the objective of this paper is to propose an efficient offloading policy that decides which tasks should be offloaded and to which offloading server (cloudlets or cloud), while minimizing the total energy consumed by the mobiles. Given our system description and according to the QoS and offloading servers’ resources capabilities constraints, our problem can be formulated as follows:

\[
\text{Minimize } \sum_{m} \sum_{n} Z_{m,n}
\]

Subject to:

\[
\begin{align*}
C1 & : \sum_{k=1}^{K} x_{m,n}^k \leq 1, \forall m \in M, u_{m,n} \in N_m \\
C2 & : y_{m,n} = -\sum_{k=1}^{K+1} x_{m,n}^k \leq 0, \forall m \in M, u_{m,n} \in N_m \\
C3 & : T_{m,n} \leq t_{m,n}, \forall m \in M, u_{m,n} \in N_m \\
C4 & : \sum_{m} \sum_{n} x_{m,n}^k \cdot c_k \leq F_k, \forall k \in K \\
C5 & : x_{m,n}^k \in \{0, 1\}, \forall m \in M, u_{m,n} \in N_m, k \in K
\end{align*}
\]

(20)

As indicated above, our objective is to minimize the total amount of energy consumed by the mobiles. Here \( x_{m,n}^k \) is the offloading decision of the task of the user \( u_{m,n} \) to the offloading server \( k \), which means that \( x_{m,n}^k = 1 \) if the user \( u_{m,n} \) offloads its task to the offloading server \( k \), and 0 otherwise. \( Z_{m,n} \) is the amount of energy consumed by the task of the user \( n \) on the access point \( m \), and can be computed as following:

\[
Z_{m,n} = (1 - \sum_{k=1}^{K+1} x_{m,n}^k) \cdot Z_{m,n}^l + \sum_{k=1}^{K+1} x_{m,n}^k \cdot Z_{m,n}^r
\]

(21)

Constraint (C1) ensures that each task is assigned at most to one offloading server. Constraint (C2) guarantee that any static offloading decision task must be assigned to exactly one offloading server, and a dynamic offloading decision task may be assigned to at most one offloading servers. The next constraint (C3) ensures that the QoS required by the task, in term of completion time, must be less than a given threshold. The processing time of task \( r_{m,n} \) can be expressed as following:

\[
T_{m,n} = (1 - \sum_{k=1}^{K+1} x_{m,n}^k) \cdot T_{m,n}^l + \sum_{k=1}^{K+1} x_{m,n}^k \cdot T_{m,n}^r
\]

(22)

The next constraint C4 shows that it is not possible to exceed the offloading capacity of the offloading server. Finally, constraint C5 ensures that the decision, \( x_{m,n}^k \), variable is a binary variable.

Theorem 1. The problem defined by equations 20 is a Non-Linear Binary Integer Problem (NLBIP) with an exponential function and constraints. It is an NP-hard problem.

Proof. Let us consider a special case where the same number of users are associated to each access point and all tasks are static offloading decision. So, all the tasks must be offloaded to the cloudlets and the bandwidth allocated to each user is known in advance. Thus, the special case is Linear Binary Integer Problem (LBIP). In fact, this special case can be easily reduced to the General Assignment Problem (GAP) with assignment restrictions, which is NP-hard as
shown in [10]. Since the special case is NP-hard, the problem 20 is also NP-hard.

One possible approach to resolve the above problem is to use some decomposition techniques such as Lagrangian relaxation. Thus, we introduce the Lagrangian multipliers \( \lambda \) on the constraint C5, since it is considered as a complicating constraint [15]. Here, \( \lambda_k \) denotes the price of all the tasks performed by the \( k \)-th offloading server. The Lagrangian function is given by:

\[
L(X, \lambda) = \sum_{m}^{M} \sum_{n}^{N_m} (Z_{m,n} \sum_{k}^{K+1} \lambda_k x_{m,n}^k + c_k) - \sum_{k}^{K+1} \lambda_k \pi_k
\]

In this case, the Lagrange dual problem for the primal problem (20) is then given by:

\[
\max \min_{X, \lambda} L(X, \lambda)
\]

We can see that the Lagrange dual problem is separable into two levels. The first level is the inner minimizing and consists of \( M \) subproblems for the \( M \) access points. The second level is the outer maximization and represents the master problem that consider the global variable and constraint C4.

4.1 Local Computation Offloading Decision Heuristic

As introduced in the last section, we decompose the Lagrange Dual problem into \( M \) subproblems. Each subproblem concerns one access point and aims to offload the task which belong to the users associated to that access point. This subproblem can be formulated as following:

\[
\text{Minimize} \sum_{m}^{M} \sum_{n}^{N_m} (Z_{m,n} \sum_{k}^{K+1} \lambda_k x_{m,n}^k + c_k)
\]

Subject to:

- constraints C1 - C3 and C5

From the last formulation, we can observe that we need to compute the bandwidth allocated to each user. Unfortunately, according to equation 8 this bandwidth depends on the number of users that offload their tasks (\( \pi_m \)). To overcome this dependency problem, we use a branching heuristic. Basically, it consists of finding the optimal value of \( \pi_m \) that gives the minimum offloading cost of the subproblem 21. We can easily derive a lower bound for \( \pi_m \), since the minimum number of tasks that should be offloaded by the users are the tasks that are considered as static offloading decision tasks. Similarly, we can also derive an upper bound for \( \pi_m \), which corresponds to the total number of tasks (\( N_m \)) belonging to the users of the access point \( m \). Consequently, we have to add one additional constraint (C6) to the subproblem formulation (21), as following:

\[
C6 : \sum_{m}^{M} \sum_{n}^{N_m} x_{m,n}^k = \pi_m
\]

To solve our subproblem, we propose a distributed greedy heuristic to select which user should offload its task and to which offloading server (Cloudlet). Our proposed heuristic is illustrated in the algorithm 1. Basically, we start our algorithm by finding the best offloading server for all static offloading tasks, that minimize the Lagrangian cost \( Z_{m,n}^k + \lambda_k c_k \) under the constraints C1 - C3 and C5 - C6. There after, since the wireless bandwidth at the access point maybe not enough to offload all the remaining dynamic offloading decision tasks, we propose to define an order to determine which task must be offloaded at first. To do this, we compute for each dynamic offloading decision task an offloading priority defined as following:

\[
a_{m,n} = Z_{m,n} - \min_{k \in K} Z_{m,n}^k
\]

This offloading priority depicts the potential gain, in terms of energy saving, between a local execution or an offloading of the task. The idea here is that when \( a_{m,n} \) increases, the offloading of the task is preferred since more energy is saved at the mobile. Finally, once the number of the offloading task is equal to the current offloading capacity (\( \pi_m \)), the remaining tasks are assigned to being performed locally by the users.

Algorithm 1 ECESO offloading heuristic

Output: output the offloading decisions \( X \) and cost \( Z \);
1: for each value of \( \pi_m \) do
2: allocate bandwidth using equation 14 and 15;
3: offload each static offloading decision task to the offloading server \( k \) that minimizes \( Z_{m,n}^k + \lambda_k c_k \) under constraints C1 - C3 and C5 - C6;
4: compute \( a_{m,n} \) for every dynamic offloading decision task;
5: sort dynamic offloading decision tasks in decreasing order of \( a_{m,n} \);
6: offload each dynamic offloading decision task to the offloading server \( k \) that minimizes \( Z_{m,n}^k + \lambda_k c_k \) under constraints C1 - C3 and C5 - C6. Otherwise, the task must be performed locally;
7: update the best offloading cost \( Z \) and decisions \( X \);
8: end for

4.2 Global Offloading Decision and Lagrangian Adjustment Heuristics

The outer level of the Lagrangian dual problem refers to the global offloading decision problem. It ensures a feasible offloading solution of the primal problem. As known, the optimal solution of the Lagrange dual requires an exhaustive search of all solutions’ space and Lagrange multiplier values which is a difficult task in general. Consequently, we need to adopt an alternative approach. In this work, we use the Subgradient-based heuristic proposed in [15]. This heuristic has three steps, first solve the subproblems, using our proposed heuristics, for the current value of \( \lambda \). Then, we check if the obtained a solution is not feasible. If so, we use a Lagrangian Adjustment Heuristic (LAH) to get a feasible solution using local searches. The idea of LAH is to check if every offloading server respect the constraint C4. When an offloading server does not respect this constraint, LAH tries to migrate some tasks offloaded from this offloading server to other offloading servers that respects all constraints. Finally, we update the Lagrange multipliers as following:
\[
\lambda_k(t + 1) = \lambda_k(t) + \theta(t) \star \left( \sum_{m} \sum_{n} x_{m,n}^k + c_k - F_k \right)
\]

where \( \theta(t) \) is the update step. In this work, we use Held and Karp formula [15] to update this step as following:

\[
\theta(t) = \eta(t) \times \frac{Z^* - Z(t)}{\sum_{k=1}^{K+1} \sum_{m} \sum_{n} x_{m,n}^k + c_k - F_k}^2
\]

where \( \eta(t) \) is a decreasing adaptation parameter \( 0 < \eta(0) \leq 2 \), \( Z^* \) is the best obtained solution of the problem 20 and \( Z(t) \) refers to the current solution of the Lagrangian Dual. We have:

\[
\eta(t + 1) = \begin{cases} 
\eta(t) & \text{if } Z(t) \text{ did not increase} \\
\eta(t) & \text{otherwise}
\end{cases}
\]

As suggested in [15] we set the values of \( \bar{\eta} = 0.9 \) and \( \eta(0) = 2 \). The master problem repeats these steps until the stop conditions, which are the maximum number of iterations \( N_{max} \) and the maximum error \( \epsilon \).

### 5 NUMERICAL RESULTS

In this section, we evaluate the performance of our proposed offloading policy by evaluating several performance metrics, e.g. the average number of offloaded tasks, and the energy saving from the offloading under different settings. The MEC environment is consisting of a metropolitan area, which is composed of 20 access points and four cloudlets already deployed among the network. In addition, two network topologies are considered. The ring topology, in which the cloudlets are equidistantly deployed in the access point, i.e. cloudlet 1 is collocated with the access point 1, cloudlets 2 with the access point 6, cloudlet 3 with the access point 11 and cloudlet 4 with the access point 16. The second topology is generated by GT-ITM [16] tool, where the cloudlets are randomly deployed. In order to get a better understanding of the offloading policies performance, we consider real mobile applications. Table 2 illustrates the characteristics of the used applications, where \( \gamma \) indicates the computing resources required to perform the applications, \( ap \) represents the data that must be transmitted to the remote server, \( dw \) the data that should be received from the remote server and \( t \) the maximum tolerated delay according to the QoS required by the application. The first three applications are static offloading decision tasks, and the remaining applications are dynamic offloading decision tasks [6].

| Application | \( \gamma \) (G cycles) | \( ap \) (KB) | \( dw \) (byte) | \( t \) (sec) |
|-------------|-------------------------|--------------|----------------|----------|
| FACE        | 12.3                    | 62           | 60             | 5        |
| SPEECH      | 15                      | 243          | 50             | 5.1      |
| OBJECT      | 44.6                    | 73           | 50             | 13       |
| Linpack     | 50                      | 10240        | 120            | 62.5     |
| CPU BENCH   | 3.36                    | 80           | 80             | 4.21     |
| PI BENCH    | 130                     | 10240        | 200            | 163      |

In addition, we consider two cloudlets configurations. In the first configuration, each cloudlet has a computing capacity of 1000 Giga cycles/s, and allocates 10 Giga cycles to perform every offloaded task (\( F_k = 1000 \) and \( c_k = 10 \)). In the second configuration, we consider heterogeneous cloudlets, where cloudlets 1 and 2 have a computing capacity of 500 Giga cycles/s and cloudlets 3 and 4 have a computing capacity of 1500 Giga cycles/s. Both upload and download bandwidths of each access point are set to 150 mbps and the bandwidth allocated to each user is estimated using the parameter settings used in Bianchi model [1]. Regarding the backhaul network, we use the parameters presented in [13]. Moreover, as in [18], we assume a homogeneous users distribution in the network. As in [2], we also consider that the power consumed in transmission mode is equal to the power consumed in the reception mode and is equal to ten times the power consumed in idle mode. We set \( P_{idle} \) to 100 mWatts. The local computing capability of each user was randomly chosen from \( F_{m,n} \in [0.8, 1, 1.2] \) Giga cycles/s. Finally, we consider that each user randomly chooses an application from those described in the table 2.

In order to evaluate the performances of our proposal, we propose to compare it with the following offloading policies:

- **DOTA** [11, 18]: In DOTA, each access point is associated with the nearest cloudlet. In this case, all users connected to this access point offload their tasks to the same cloudlet. When a cloudlet is overloaded, the tasks are migrated to the remote cloud.
- **CBL** [8, 9]: Using CBL, we also associate each access point to the nearest cloudlet. Thus, all users connected to that access point have to offload their tasks into that same cloudlet. However, unlike DOTA, when the cloudlet is overloaded, the tasks are migrated to another cloudlet.
- **FCO** [3, 4]: In this policy, all users offload their tasks to the cloud.

In order to compare these offloading policies, we also define comparison metrics depicting the gain of a given offloading policy \( P \). This gain represents the benefit of the offloading policy \( P \) compared to case where the task is offloaded to the cloud (i.e. FCO policy). We formulate the gain as following:

\[
\text{gain of } P = 100 \times \frac{\text{cost of } FCO - \text{cost of } P}{\text{cost of } FCO}
\]

In Fig. 2, we plot the gain of our policy (ECSEO) compared to the gain of DOTA and CBL, when considering a network topology following the configuration 1 with homogeneous cloudlet characteristics. As expected, we can observe that the gain decreases when the number of users increases. This is mainly due to the fact that the backhaul cost increases when the number of offloaded task increases. We can also observe that the performances of ECSEO, DOTA and CBL are almost equivalent, except when the number of users exceeds 300, where we notice that our approach is slightly better. This is due to the fact that ECSEO tries to maximize the bandwidth allocated to each users and offload in priority the tasks with the greatest impact on the cost. Consequently, less tasks are offloading compared to the other offloading policies.

In Fig. 3, we compare the performances of ECSEO, BCL and DOTA in the case where heterogeneous cloudlets are deployed for both ring and GT-ITM topologies. As we can see, when few number
which reduce the additional offloading cost due to the migration and CBL assign statically the access point to the cloudlet. ECESO uses the topology to select the cloudlets. However, DOTA is more important than the DOTA and CBL. This is due to the fact that loaded tasks to the remote cloud, which adds more offloading costs. However, where the remote cloud has greater computing capacity than cloudlets ($c_{\text{cloud}} > c_{\text{cloudlets}}$) all the tasks are offloaded to the remote cloud. Fig. 4(b), 4(c) and 4(d) illustrate the performances of the ECESO policy for dynamic offloading decision tasks, CPUBENCH, PIBENCH and Linpack, respectively. The ratio of the offloading tasks decreases when the number of the users in the network increases, for example, for CPUBENCH application 100% of tasks are offloaded where the number of users is not greater than 200, but only 30% are offloaded where 1000 users are in the network. This decreasing of the ratio of offloaded tasks is due to the wireless bandwidth in the access point. We also note that the ratio of the offloaded tasks depends on the application characteristics when the applications require a lot of computing resources and transmit a huge amount of data (Linpack and PIBENCH) the ratio of offloaded tasks is lower. As a result, choose the placement of the tasks, remote cloud or cloudlet, depends on many factors, in the figures we noticed that the computing resource allocation in the remote cloud, the computing resource allocation in the cloudlets and the characteristics of the application affect directly the placement of the offloaded tasks.

6 CONCLUSION

In this paper, we propose a new computation offloading policy in two-tier MEC environment. The proposed offloading policy decides which users should offload, to which offloading server and allocated the wireless bandwidth accordingly. First, we formulate the problem as a Non-Linear Binary Integer Program (NLBIP). Then, we propose an efficient heuristic to solve the problem using Lagrangian decomposition approach. The proposed heuristic uses a branching algorithm to maximize the bandwidth allocation and minimize the energy consumption. The numerical results show performance improvements in terms of the energy consumption compared to existing offloading policies under different scenarios.

REFERENCES

[1] Giuseppe Bianchi. 2000. Performance analysis of the IEEE 802.11 distributed coordination function. IEEE Journal on selected areas in communications 18, 3 (2000), 535–547.
[2] Aaron Carroll, Gernot Heiser, et al. 2010. An Analysis of Power Consumption in a Smartphone. In USENIX annual technical conference, Vol. 14. Boston, MA, 21–21.
[3] Meng-Hsi Chen, Ben Liang, and Min Dong. 2016. Joint offloading decision and resource allocation for multi-user multi-task mobile cloud. In 2016 IEEE International Conference on Communications (ICC). IEEE, 1–6.
[4] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2016. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. IEEE/ACM Transactions on Networking 24, 5 (2016), 2795–2808.
[5] Keke Gai, Meikang Qiu, and Hui Zhao. 2018. Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing. J. Parallel and Distrib. Comput. 111 (2018), 126–135.
[6] Ying Gao, Wenzhu Hu, Kiryong Ha, Brandon Amos, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2015. Are Cloudlets Necessary? October (2015).
Figure 4: Comparison of offloaded tasks to cloudlets and the remote cloud under different users and cloud computing capacity ($C_{\text{cloud}}$ in Giga cycles/s).

[7] Songtao Guo, Bin Xiao, Yuanyuan Yang, and Yang Yang. 2016. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In 35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016. Vol. 2016-July. IEEE, 1–9.

[8] Mike Jia, Jiannong Cao, and Weifa Liang. 2015. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. IEEE Transactions on Cloud Computing, PP, 99 (2015).

[9] Mike Jia, Weifa Liang, Zichuan Xu, and Meitian Huang. 2016. Cloudlet load balancing in wireless metropolitan area networks. In Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on, Vol. 2016-July. 1–9.

[10] Sven O Krumke and Clemens Thielen. 2013. The generalized assignment problem with minimum quantities. European Journal of Operational Research 228, 1 (2013), 46–55.

[11] Longjie Ma, Jigang Wu, and Long Chen. 2017. DOTA: Delay Bounded Optimal Cloudlet Deployment and User Association in WMANs. In Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE Press, 196–203.

[12] Anwesha Mukherjee, Debasis De, and Deepsubhra Guha Roy. 2016. A power and latency aware cloudlet selection strategy for multi-cloudlet environment. IEEE Transactions on Cloud Computing, PP, 99 (2016), 1–14.

[13] Jessica Oueis, Emilio Calvaneo-Strinati, Antonio De Domenico, and Sergio Barbarossa. 2014. On the impact of backhaul network on distributed cloud computing. In Wireless Communications and Networking Conference Workshops (WCNCW), 2014 IEEE, 12–17.

[14] Deepsubhra Guha Roy, Debasis De, Anwesha Mukherjee, and Rajkumar Buyya. 2016. Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. The Journal of Supercomputing (2016), 1–19.

[15] Jiufu Tang, Chongjun Yan, Xiaoging Wang, and Chengkuan Zeng. 2014. Using lagrangian relaxation decomposition with heuristic to integrate the decisions of cell formation and parts scheduling considering intercell moves. IEEE Transactions on Automation Science and Engineering, 11, 4 (2014), 1110–1121.

[16] Megan Thomas and Ellen W Zegura. 1994. Generation and analysis of random graphs to model internetworks. Technical Report, Georgia Institute of Technology.

[17] Ilene Timmirello, Giuseppe Bianchi, and Yang Xiao. 2010. Refinements on IEEE 802.11 distributed coordination function modeling approaches. IEEE Transactions on Vehicular Technology 59, 3 (2010), 1055–1067.

[18] Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, and Song Guo. 2016. Efficient Algorithms for Capacitated Cloudlet Placements. IEEE Transactions on Parallel and Distributed Systems 27, 10 (2016), 2866–2880.

[19] Hong Yao, Changmin Bai, Muzhou Xiong, Dazhi Zeng, and Zhangjie Fu. 2017. Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing. Concurrency and Computation: Practice and Experience 29, 16 (2017).