DM2-ECOP: An Efficient Computation Offloading Policy for Multi-user Multi-cloudlet Mobile Edge Computing Environment

HOUSSEMEDDINE MAZOUZI and NADJIB ACHIR, L2TI, Institut Galilée, Université Paris 13, Sorbonne Paris Cité, France
KHALED BOUSSETTA, L2TI, Institut Galilée, Université Paris 13, Sorbonne Paris Cité, Agora/INRIA, France

Mobile Edge Computing is a promising paradigm that can provide cloud computing capabilities at the edge of the network to support low latency mobile services. The fundamental concept relies on bringing cloud computation closer to users by deploying cloudlets or edge servers, which are small clusters of servers that are mainly located on existing wireless Access Points (APs), set-top boxes, or Base Stations (BSs). In this article, we focus on computation offloading over a heterogeneous cloudlet environment. We consider several users with different energy—and latency-constrained tasks that can be offloaded over cloudlets with differentiated system and network resources capacities. We investigate offloading policies that decide which tasks should be offloaded and select the assigned cloudlet, accordingly with network and system resources. The objective is to minimize an offloading cost function, which we defined as a combination of tasks' execution time and mobiles' energy consumption. We formulate this problem as a Mixed-Binary Programming. Since the centralized optimal solution is NP-hard, we propose a distributed linear relaxation-based heuristic approach that relies on the Lagrangian decomposition method. To solve the subproblems, we also propose a greedy heuristic algorithm that computes the best cloudlet selection and bandwidth allocation following tasks' offloading costs. Numerical results show that our offloading policy achieves a good solution quickly. We also discuss the performances of our approach for large-scale scenarios and compare it to state-of-the-art approaches from the literature.

Q1 CCS Concepts: • **Networks → Cloud computing**; • **Human-centered computing → Ubiquitous and mobile computing systems and tools**; • **Theory of computation → Network optimization**;

Additional Key Words and Phrases: Computation offloading, mobile cloud computing, mobile edge computing, cloudlet, Lagrangian decomposition

ACM Reference format:
Houssemeddine Mazouzi, Nadjib Achir, and Khaled Boussetta. 2019. DM2-ECOP: An Efficient Computation Offloading Policy for Multi-user Multi-cloudlet Mobile Edge Computing Environment. *ACM Trans. Internet Technol.* 19, 2, Article 24 (March 2019), 24 pages.
https://doi.org/10.1145/3241666

Authors' addresses: H. Mazouzi and N. Achir, L2TI, Institut Galilée, Université Paris 13, Sorbonne Paris Cité, 99 Avenue J-B Clement, Villetaneuse, 93430, France; emails: {mazouzi.houssemeddine, nadjib.achir}@univ-paris13.fr; K. Boussetta, L2TI, Institut Galilée, Université Paris 13, Sorbonne Paris Cité, Agora/INRIA, 99 Avenue J-B Clement, Villetaneuse, 93430, France; email: khaled.boussetta@univ-paris13.fr.

1533-5399/2019/03-ART24 $15.00
https://doi.org/10.1145/3241666

1 INTRODUCTION

The Mobile Cloud Computing paradigm has been proposed to allow remote execution of resource-hungry mobile applications in the cloud. The application's computation is then transmitted to the remote cloud to be performed. The latter operation is known as **computation offloading** [6, 7]. Unfortunately, the geographical distance between the cloud and user can introduce large and variable latency. That can significantly degrade the quality of experience of delay-sensitive applications, such as mobile gaming, augmented-reality, and face and speech recognition [8, 30]. To overcome such problems, **Mobile Edge Computing (MEC)** has emerged as a main paradigm that aims to provide cloud computing capabilities at the edge of the network to support latency-sensitive mobile applications. The main concept relies on deploying small clusters of servers, called cloudlets, at the edge of the network [11, 18]. Users can then offload their computation to closer cloudlets.

In multi-user context, several mobile devices can compete to offload their computations to the cloudlets. Hence, the performances of offloading policies are strongly dependent on the cloudlets' computational resources sharing and on the wireless bandwidth allocation strategies [4, 5, 12]. In addition, in a multi-cloudlet MEC environment, where many cloudlets are available around users, the performance of the computation offloading depends on the cloudlet selection [14, 33, 34].

Many recent works have investigated cloudlet selection problems [21, 27, 34]. Most of the proposed offloading policies rely on user density to statically assign each region to a cloudlet [15, 27]. Therefore, as shown in Figure 1, users within a region will always offload to the same cloudlet. Nevertheless, the dynamic density of users may imbalance the load between the cloudlets, leading to suboptimal MEC capacities usage and longer offloading delays. Therefore, to achieve high performance, an offloading policy must jointly consider bandwidth allocation, computation resource allocation, and cloudlet selection.

To tackle this problem, we explore, in this article, computation offloading in multi-user, multi-cloudlet MEC. Our aim is to provide an efficient offloading policy, which determines the best offloading decision and cloudlet selection for each user, with the aim of reducing the total offloading cost.

This work presents a new computation offloading policy named Distributed Multi-user Multi-cloudlet Efficient Computation Offloading Policy (DM2-ECOP), which aims to improve the performance of offloading in an MEC environment. It extends the offloading strategies presented in References [4, 5, 12]. As in previous works, we assume that each user executes only one application at a time. However, unlike these works, we define two categories of applications that can be supported by the MEC: (1) applications that must be performed remotely and (2) applications that can be performed either locally or remotely, accordingly with the conditions at execution time. DM2-ECOP tries to select the best cloudlet according to network and system resource availability, while minimizing the offloading cost. The offloading cost is defined as a combination of the energy consumed by the mobile devices and the total applications' completion times.

We formulate this computation offloading problem as a Mixed Binary Programming. Then, we solve it using a distributed linear relaxation-based heuristic that follows the Lagrangian decomposition approach. DM2-ECOP is composed of two decision levels: (1) The local offloading manager handles the users associated within the same AP and solves the offloading subproblem related to this AP; the local offloading manager uses our proposed algorithm, named Greedy Best Cloudlet Selection First Heuristic (GBC-SFH), which selects the cloudlet to which each application will be offloaded to minimize the energy consumption and completion times. (2) At a second level, a global offloading manager ensures that the cloudlets' resources allocated by each local offloading manager satisfy the capacity constraints of each cloudlet.

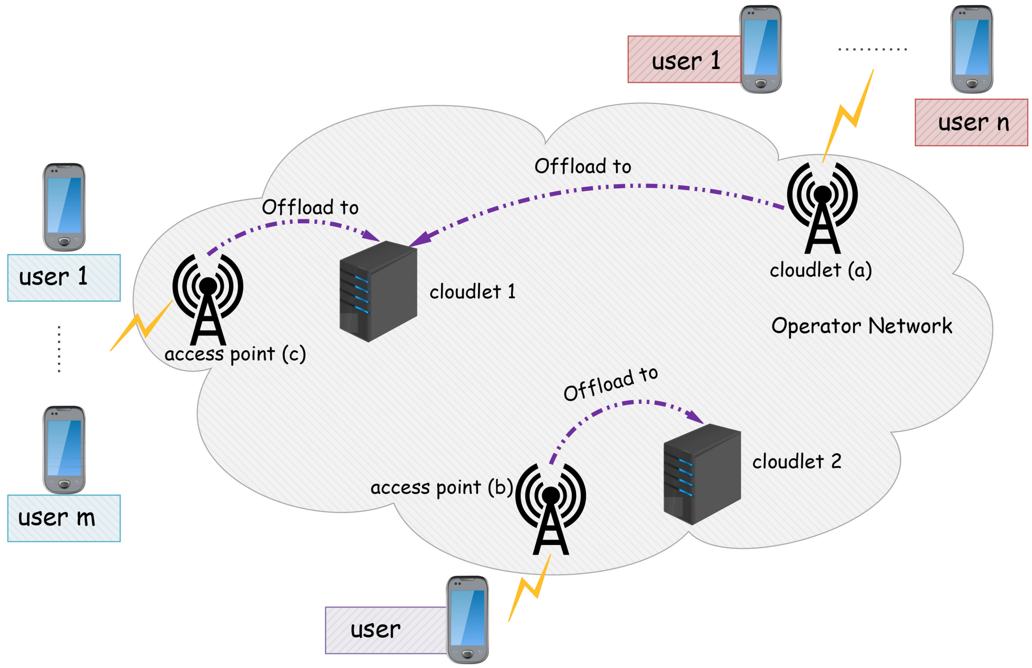


Fig. 1. Multiuser computation offloading in multi-cloudlet MEC environment.

The remainder of this article is organized as follows: Section 2 presents existing works in computation offloading. Section 3 introduces the system modeling. The multi-user, multi-cloudlet offloading problem is formulated in Section 4. Our offloading policy, named DM2-ECOP, is explained in Section 5. Performance evaluation is detailed and analyzed in Section 6. Finally, a conclusion is drawn in Section 7.

2 RELATED WORK

Many works were proposed to explore computation offloading to improve the performance of mobile devices. However, not all of the proposed offloading policies have the same goals. In the following, we distinguish between three main goals: (i) offloading decision, (ii) cloudlets placement, and (iii) cloudlet selection.

Some of the proposed works investigate the offloading decision to decide which computation should be offloaded to the remote cloud, such as: Meng-Hsi Chen et al. [4], Xu Chen et al. [5], Songtao Guo et al. [12], Keke Gai et al. [10], Yuyi Mao et al. [22], and Dong Huang et al. [13]. Meng-Hsi et al. are one of the first to work on multi-user computation offloading in mobile cloud computing. The proposed offloading policy determines which computation must be performed in the remote cloud and which one must be performed locally by the mobile device. Then, it allocates the wireless bandwidth to each user to reduce the energy consumption of the mobile device. The Xu Chen et al. offloading policy was designed for a single cloudlet MEC environment. Each user tries to offload its computation, accordingly with the available wireless bandwidth to reduce the energy consumption. Another offloading approach for multi-user was presented by Songtao Guo et al. Similar to DM2-ECOP, this work minimizes an offloading cost defined as a combination of energy consumption and processing time. The offloading policy decides which computation can be offloaded and allocates the wireless bandwidth and the processor frequency to each offloaded computation. In similar way, Keke Gai et al. proposed a scheduler to assign the tasks between the local mobile device and the remote cloud to save energy consumption. Yuyi Mao et al. presented an offloading policy that tries to offload the computation in a multi-user scenario to an edge server, cloudlet. The proposed policy allocates the CPU frequency and the bandwidth to each user to reduce the energy consumption of the mobile device. Lastly, Dong Huang et al. designed a computation offloading policy for a single-user scenario to reduce the energy consumption of the mobile device. They focused on partial offloading, where the offloading policy partitions the application at runtime to determine which computation must be performed locally and which must be offloaded to a remote server. Although all these policies improve the performances of the mobile device, they rely on an unlimited capacity of the cloud. Consequently, they need some enhancements to be applied for the MEC, where cloudlets have limited computing resources.

Yucen Nan et al. [25, 26] and Chongyu Zhou et al. [35] proposed computation offloading policies to reduce the energy consumption of fog nodes. They introduced an offloading policy where the fog nodes try to offload their computation to the remote cloud. For each fog node, the policy decides which computation must be offloaded to the remote cloud and which one must be performed locally by the node. In Reference [26], the offloading policy has been extended to reduce the completion time of the applications. Similarly, Chongyu Zhou et al. introduced an online offloading policy. It can select the computations that should be performed by the nearest cloudlet in order to minimize a system-wide utility, which is the execution time. Contrary to these policies, which reduce the energy consumption of the fog server, DM2-ECOP focuses on the reduction of the offloading cost on the mobile device side. In addition, the IoT device has a tiny computing capacity that cannot perform any application. However, the mobile device has considerable computing capacity that performs complex applications.

Cloudlets placement is also a challenging issue for MEC, and many recent works propose some cloudlets placement heuristics in MEC environment. Mike Jia et al. [14, 33], Hong Yao et al. [34], and Longjie Ma et al. [21] introduced cloudlets placement and selection algorithms in a multi-user, multi-cloudlet MEC environment. The Mike Jia et al. offloading approach is one of the first heuristics on cloudlets placement in a large-scale environment. Its main goal is to find the best cloudlets placement in a large network, then select a cloudlet to perform the computation of each AP. The K-median clustering based on user density is used to place the cloudlets. Then each AP is statically assigned to a cloudlet. Similarly, Hong Yao et al. have been designing heuristics to support heterogeneous cloudlets environment. Finally, Longjie Ma et al. have been introducing a heuristic to find the minimal number of cloudlets that must be placed to improve the user experience quality in a large-scale network. In a multi-user MEC environment, the density of mobile users is dynamic and changes over time. So, static assignment of the APs to cloudlets may decrease the performance of the computation offloading. To avoid this problem, our DM2-ECOP approach considers dynamic cloudlet selection and wireless bandwidth allocation with the aim of minimizing energy consumption and improving the performance of mobile devices.

Other works try to find a dynamic cloudlet selection in a multi-cloudlet MEC environment. Anwesha Mukherjee et al. [24, 27], Mike Jia et al. [15], Qiliang Zhu et al. [36], and Arash Bozorgchenani et al. [2] have proposed to support the dynamic cloudlet selection to reduce the offloading cost. Anwesha Mukherjee et al. designed a multilevel offloading policy to optimize energy consumption. The users offload to the nearest cloudlet in the first step. According to the amount of resources available in this cloudlet, it can perform the task or offload it to another cloudlet. Mike Jia et al. introduced a heuristic to balance the load between the cloudlet. Its main goal is to migrate some computations from overloaded cloudlets to underloaded cloudlets to reduce the execution time. Similarly, Qiliang Zhu et al. developed a two-tier offloading policy, where the mobile device offloads its computation to an offloading server based on the resource availability. They used an agent that decides to perform the computation in the local cloudlet or to offload it to the remote cloud. Arash Bozorgchenani et al. offloading policy tries to select a nearby fog node to offload some computation of a busy fog node to save energy consumption and completion time. Even these works proposed dynamic cloudlet selection heuristics, the tasks still always offloaded to the nearest cloudlet that decides to perform them locally or transmit them to other cloudlets. Thus, an additional offloading cost is induced; consequently, the performance of offloading will decrease. In our proposal, the cloudlet that performs each offloaded task will be determined at the offloading decision time without any additional cost.

All the offloading policies presented above have been focusing on reducing the offloading cost. They offload computations to a predetermined remote server (the remote cloud or a local cloudlet). The selection of the remote server is done statically at the development time based on metrics such as user density, despite the fact that the density of users can change dynamically over the time. In addition, the computing capacity of the cloudlet is limited and cannot perform all the offloaded computation. To avoid this situation, the most adopted strategy in the literature was to consider a two-tier approach. Basically, the tasks are offloaded to the nearest cloudlet, and this cloudlet offloads some computation to another cloudlet or to the remote cloud when it is overloaded. Although two-tier offloading policies can improve the performance of the offloading approach, they engender an additional offloading cost. Moreover, in a multi-cloudlet scenario, where many cloudlets are available around the user, selecting the same cloudlet always is not the best strategy. Therefore, in this article, we propose a new offloading policy to improve the efficiency of computation offloading in MEC. The new policy must consider many cloudlets for which a user can offload its computation and compute optimal computation placements to optimize the offloading cost.

3 SYSTEM DESCRIPTION

In this section, we describe our system modeling. We first introduce the MEC model, then we present the communication and computation offloading models. Finally, an offloading cost is proposed as an objective function for our optimization problem. Table 1 presents variables and notations used, in this article, to model our multi-user, multi-cloudlet computation offloading problem.

3.1 MEC Environment Model

Let us consider an MEC environment composed of M APs and K cloudlets, as illustrated in Figure 2. We suppose that the number of cloudlets is less than the number of APs ($K \leq M$). In this article, we assume that the cloudlets have already been deployed and are co-located with the APs. We also consider that the users can communicate with the cloudlets through their APs. We denote in the following the set of APs by $\mathcal{M} = \{1, 2, \ldots, M\}$, and we assume that each AP i is associated with N_i users. Let us consider $\mathcal{N}_m = \{1, 2, \ldots, N_m\}$ as the set of users associated with the m^{th} AP and $\mathcal{K} = \{1, 2, \ldots, K\}$ as the set of the cloudlets. We also define $u_{m,n}$ as to n^{th} user of the m^{th} AP. Similar to existing works [14, 22, 33], every user runs one application on his mobile device. The application is characterized by its: (i) computational resource requirement in terms of CPU cycles, denoted by $\gamma_{u_{m,n}}$, (ii) the amount of data uploaded to MEC, denoted by $up_{u_{m,n}}$, (iii) the amount of data that must be downloaded by the user from MEC at the end of execution on the MEC, denoted by $dw_{u_{m,n}}$, and (iv) finally, the maximum tolerated delay according to the Quality of Service (QoS) required by the application, denoted by $t_{u_{m,n}}$.

As considered in previous works [11, 19], we distinguish between two categories of computation offloading applications: (1) *static offloading decision task* and (2) *dynamic offloading decision task*. In the first category, the application is partitioned in advance at the design time between: a local part (task) that should always be executed in the mobile and a remote part (task) that should always

Table 1. Notation

Symbol	Description
$\gamma_{u_{m,n}}$	the computational resource required by the task of the user $u_{m,n}$.
λ_m	the Lagrangian multiplier of the subproblem m.
c_k	the computing resource allocation on cloudlet k.
$dw_{u_{m,n}}$	the amount of data to download by the user $u_{m,n}$ from the MEC.
$E^l_{u_{m,n}}$	the local energy consumption for user $u_{m,n}$.
$E^e_{u_{m,n},k}$	the remote energy consumption for user $u_{m,n}$ in cloudlet k.
F_k	the computing capacity of the cloudlet k.
$f_{u_{m,n}}$	the local computing capacity of the user $u_{m,n}$.
K	the number of cloudlets available in the network.
M	the number of APs in network.
N_m	the number of users associated to the AP m.
$P^{tx/rx}_{u_{m,n}}$	power consumption when the Wi-Fi interface is transforming or receiving data.
$P^{Idle}_{u_{m,n}}$	power consumption when the Wi-Fi interface is not transforming or receiving data.
$r_{u_{m,n}}$	the task of the user $u_{m,n}$.
$t_{u_{m,n}}$	the maximum tolerated delay according the QoS of the task of the user $u_{m,n}$.
$T^t_{u_{m,n},k}$	the communication time when user $u_{m,n}$ offload to cloudlet k.
$T^l_{u_{m,n}}$	the local processing time for user $u_{m,n}$.
$T^e_{u_{m,n},k}$	the remote processing time for user $u_{m,n}$ in cloudlet k.
$u_{m,n}$	the n^{th} user for the m^{th} AP.
$up_{u_{m,n}}$	the amount of data uploaded to the MEC from the user $u_{m,n}$.
W_m	the wireless data rate at the AP m.
$w_{u_{m,n}}$	the allocated data rate for the user $u_{m,n}$.
$x_{u_{m,n},k}$	the offloading decision variable for the task of user $u_{m,n}$ in the cloudlet k.
$y_{u_{m,n}}$	the category belong the tasks (1 static offloading decision task and 0 otherwise).
$\mathcal{Z}^l_{u_{m,n}}$	the local offloading cost for user $u_{m,n}$.
$\mathcal{Z}^e_{u_{m,n},k}$	the remote offloading cost for user $u_{m,n}$ on cloudlet k.

be executed remotely. As illustrated in Figure 3(a), the task's source code is already in the remote server, so the mobile device needs to transmit only the input data to the remote server. A typical example of static offloading decision task is the FLUID application on Android [16] that is used for particle simulations. The thin client side of FLUID is executed on the mobile, while the server part must be performed remotely in MEC, because it requires high-performance GPU computing processors that are not commonly available in mobile devices.

In the second category, the application needs to be partitioned at runtime accordingly with the network and MEC resource availability. Basically, the mobile terminal needs to decide if it is useful to execute the task on the mobile or to offload all or part of the task. In this case, as illustrated in Figure 3(b), the mobile device must transmit its source code and the input data when the task is offloaded. An example of this kind of application is the Linpack benchmarks [32] for Android, which aims to measure the performances of Android devices. This application can be either totally executed on the mobile terminal or partially offloaded to a cloudlet.

To simplify the analysis, we model in this article both *static offloading decision tasks* and *dynamic offloading decision tasks* as tasks with an offloading computation ratio noted as $a_{u_{m,n}}$. Basically, $a_{u_{m,n}}$ denotes the computation ratio of the application that should be executed remotely. To

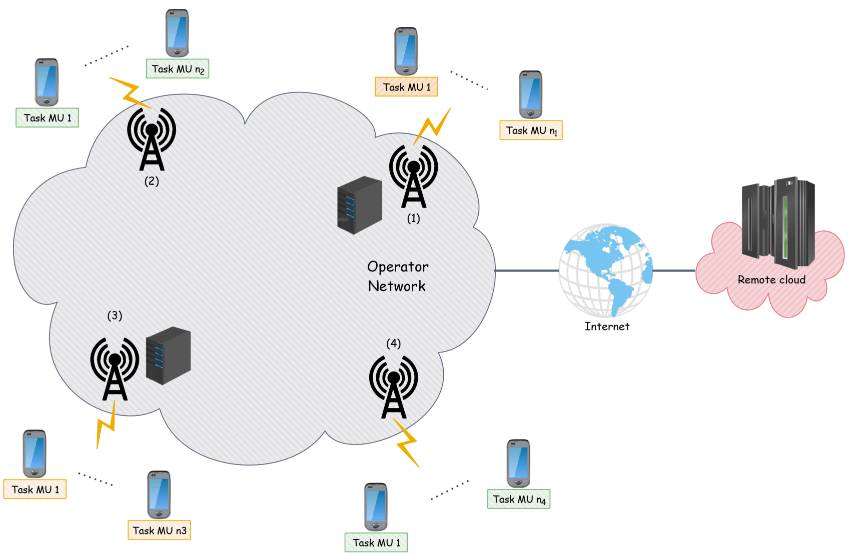
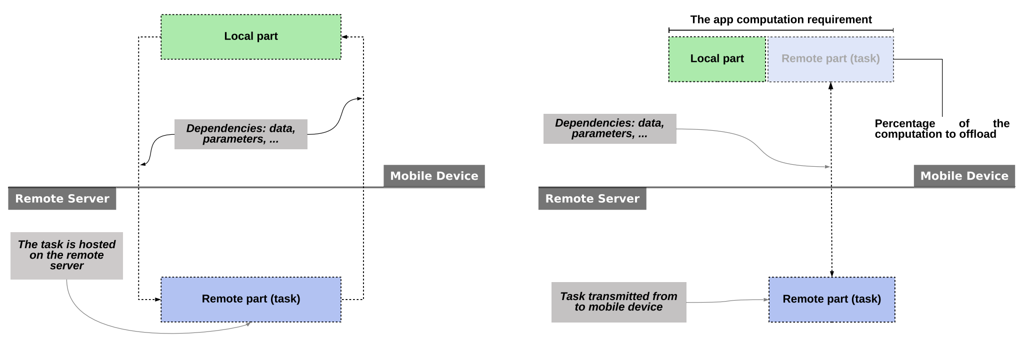


Fig. 2. Example of a multi-user (MU), multi-cloudlet MEC environment with 4 APs and 2 cloudlets ($M = 4$, $K = 2$).



(a) Static offloading decision task

(b) Dynamic offloading decision task

Fig. 3. Illustration of the two categories of tasks.

distinguish between *static offloading decision task*s and *dynamic offloading decision tasks*, we consider that for *static offloading decision tasks* the offloading ratio is always equal to 1, which means that the tasks that belong to that category are always offloaded. The local part is considered equal to zero, because it will not affect the performance of the system, since it should always be executed at the terminal. However, for *dynamic offloading decision tasks,* this offloading computation ratio can take any value between 0 and 1 (i.e., $a_{u_{m,n}} \in [0, 1]$). In addition, for simplicity (but without the loss of generality), we also assume that when a task is offloaded with a given offloading computation ratio, then the amount of data that should be transmitted are also proportional to that ratio. However, the output of the task, noted as $dw_{u_{m,n}}$, does not change whatever the value of $a_{u_{m,n}}$.

To indicate which category the application of user $u_{m,n}$ belongs, we introduce the binary variable $y_{u_{m,n}}$, which is equal to 1 for *static offloading decision task*s and 0 for *dynamic offloading decision tasks.*

Finally, due to hardware and software constraints required by the task, we assume that some cloudlets cannot perform some tasks. In this case, we define a second binary variable, $g_{u_{m,n},k}$, to indicate if the cloudlet k can perform the task. Thus, $g_{u_{m,n},k}$ is equal to 1 if the k^{th} cloudlet can execute the task, 0 otherwise.

3.2 Communication Model

Let $G = (V, E)$ be a weighted graph, where V $(\mathcal{M} \cup \mathcal{K})$ is a finite set of vertices corresponding to the sets of APs and cloudlets and E is a set of connections (edges) denoting a possible communication between any two vertices. We also consider a weight of the edge, noted as $e_{i,j}$, that represents the network delay between each two vertices i and j. Thus, using the Dijkstra algorithm, we can compute a delay matrix, noted as $\mathcal{D}_{m,k}$, that represents the delay between the m^{th} AP and the k^{th} cloudlet. According to the last considerations, we can estimate the bandwidth allocated to each user as follows:

$$w_{u_{m,n}} = \frac{W_m(\pi_m)}{\pi_m}, \tag{1}$$

where π_m is the **offloading capacity** that represents the number of users that offload their applications to MEC, as defined in Definition 5.1. $W_m(\pi_m)$ is the bandwidth shared by the m^{th} AP between the π_m users associated with it. To estimate this bandwidth, we consider the Bianchi model [1, 20].

According to the last assumptions, we can compute the communication time of any task. This time, noted as $T^t_{u_{m,n},k}$, is composed of time needed to upload the data from the user terminal to the cloudlet, plus the time needed to download the results from the cloudlet once the task is completed. In this case, $T^t_{u_{m,n},k}$ can be written as:

$$\begin{aligned} T^t_{u_{m,n},k} &= a_{u_{m,n}} \cdot \frac{up_{u_{m,n}}}{w_{u_{m,n}}} + \mathcal{D}_{m,k} + \frac{dw_{u_{m,n}}}{w_{u_{m,n}}} + \mathcal{D}_{m,k} \\ &= \frac{a_{u_{m,n}} \cdot up_{u_{m,n}} + dw_{u_{m,n}}}{w_{u_{m,n}}} + 2\mathcal{D}_{m,k}. \end{aligned} \tag{2}$$

3.3 Computation Processing Model

a) *Local processing:* We assume that the user's device has a local computational capability of $f_{u_{m,n}}$ used for the task computation. When a user offloads a percentage $a_{u_{m,n}}$ of its computation, the remaining part must be performed locally. So, the local processing time for the local part can be estimated by:

$$T^l_{u_{m,n}} = (1 - a_{u_{m,n}}) \cdot \frac{\gamma_{u_{m,n}}}{f_{u_{m,n}}}. \tag{3}$$

From the above equation, we can notice that the local processing time of a Static offload decision task is equal to zero ($T^l_{u_{m,n}} = 0$), since $a_{u_{m,n}}$ for static offload decision task is always equal to 1.

b) *Remote processing:* For remote computation, we consider that each cloudlet has a computational capability of F_k. Every cloudlet allocates some of its computational resource to perform the offloaded tasks. So, the remote execution time of the user's $u_{m,n}$ task can be estimated as the ratio of the offloaded computational resources and the allocated computation resource, as illustrated by the following equation:

$$T^e_{u_{m,n},k} = a_{u_{m,n}} \cdot \frac{\gamma_{u_{m,n}}}{c_k}, \tag{4}$$

where c_k is the amount of computational resource allocated to perform the task at the k^{th} cloudlet. To simplify our model, we consider that the allocated resource at each cloudlet is fixed and does not change during the computation [4, 5, 12].

3.4 Offloading Cost Model

We define the offloading cost as a combination of the energy consumption and the execution time of the application. In the following, we present both local offloading cost and remote offloading cost.

a) *Local offloading cost:* According to the forgoing considerations, the local offloading cost is expressed as a combination of the energy consumption and the execution time of the tasks, as in the following:

$$\mathcal{Z}^l_{u_{m,n}} = \beta_{u_{m,n}} \cdot E^l_{u_{m,n}} + (1 - \beta_{u_{m,n}}) \cdot T^l_{u_{m,n}}, \tag{5}$$

where $E^l_{u_{m,n}}$ and $T^l_{u_{m,n}}$ are, respectively, the total amount of energy and processing time of the local part of the application of the user $u_{m,n}$. $\beta_{u_{m,n}}$ denotes the weighting parameter of execution time and energy consumption of the user's offloading decision. When the battery of the user's device is at a low state, and the user needs to reduce the energy consumption, then it can set $\beta_{u_{m,n}} = 1$. However, when a delay-sensitive task is running, the user can set $\beta_{u_{m,n}} = 0$ to give more priority to the execution time. To get a more flexible cost model, we allow a multi-criteria offloading policy by considering energy consumption, execution time, or a combination of both of them.

Considering the energy-consumption model, we use in this article the model proposed in References [3, 23]. Using this model, we can compute $E^l_{u_{m,n}}$ as following:

$$E^l_{u_{m,n}} = \kappa \cdot (f_{u_{m,n}})^3 \cdot T^l_{u_{m,n}} \tag{6}$$

where κ is the effective switched capacitance, which depends on the chip architecture and is used to adjust the processor frequency. In the following, we set $\kappa = 10^{-9}$ as shown in [3, 23].

b) *Remote offloading costs:* The total amount of energy consumed by the user's terminal to perform the task remotely is equal to the energy used when the device turns the radio in the transmission mode to send the data to the remote server, plus the energy used when the device turns the radio in idle mode to wait the task completion plus the energy used by the device when it turn again the radio in the reception mode to receive the result data from the remote server. This consumed energy can be expressed as follows:

$$E^e_{u_{m,n},k} = P^{tx/rx}_{u_{m,n}} \cdot \left(T^t_{u_{m,n},k} - 2\mathcal{D}_{m,k}\right) + P^{idle}_{u_{m,n}} \cdot \left(T^e_{u_{m,n},k} + 2\mathcal{D}_{m,k}\right), \tag{7}$$

where $P^{tx/rx}_{u_{m,n}}$ is the power consumption when the radio interface is set to transmission or reception mode, and $P^{idle}_{u_{m,n}}$ is the power consumption in the case when the radio interface is set to idle mode [3, 23, 29].

Finally, we can define the remote offloading costs as follows:

$$\mathcal{Z}^e_{u_{m,n},k} = \beta_{u_{m,n}} \cdot E^e_{u_{m,n},k} + (1 - \beta_{u_{m,n}}) \cdot \left(T^t_{u_{m,n},k} + T^e_{u_{m,n},k}\right) \tag{8}$$

4 MULTI-USER, MULTI-CLOUDLET COMPUTATION OFFLOADING PROBLEM FORMULATION AND DECOMPOSITION

To propose an efficient offloading policy, we formulate the problem as an optimization problem. Then, we use Lagrangian relaxation to decompose the problem into subproblems and solve each one separately.

4.1 Problem Formulation

As introduced earlier, the objective of this article is to propose an efficient offloading policy. It decides which users should offload their tasks, determines the amount of computation to offload, and selects a cloudlet for each user, while minimizing the total offloading cost. Let us denote $x_{u_{m,n},k}$ to the offloading decision for the task of the user $u_{m,n}$ on the cloudlet k, which means that $x_{u_{m,n},k} = 1$ if the user $u_{m,n}$ offloads its task to the cloudlet k, 0 otherwise. Given the system description and according to the QoS and cloudlets' resource-capability constraints, the can be formulated as follows:

$$
\begin{aligned}
&\textbf{Minimize} \sum_{m}^{M} \sum_{n}^{N_m} \mathcal{Z}_{u_{m,n}} \\
&\textbf{Subject to:} \\
&C1: \sum_{k=1}^{K} x_{u_{m,n},k} \leq 1, \forall m \in \mathcal{M}, u_{m,n} \in \mathcal{N}_m \\
&C2: y_{u_{m,n}} - \sum_{k=1}^{K} x_{u_{m,n},k} \leq 0, \forall m \in \mathcal{M}, u_{m,n} \in \mathcal{N}_m \\
&C3: T_{u_{m,n}} \leq t_{u_{m,n}}, \forall m \in \mathcal{M}, u_{m,n} \in \mathcal{N}_m \\
&C4: x_{u_{m,n},k} \leq g_{u_{m,n},k}, \forall m \in \mathcal{M}, u_{m,n} \in \mathcal{N}_m, k \in \mathcal{K} \\
&C5: \sum_{m}^{M} \left(\sum_{n}^{N_m} x_{u_{m,n},k} \cdot c_k \right) \leq F_k, \forall k \in \mathcal{K} \\
&C6: x_{u_{m,n},k} \in \{0,1\}, \forall m \in \mathcal{M}, u_{m,n} \in \mathcal{N}_m, k \in \mathcal{K} \\
&C7: a_{u_{m,n}} \in [0,1], \forall m \in \mathcal{M}, u_{m,n} \in \mathcal{N}_m \\
&C8: a_{u_{m,n}} \geq y_{u_{m,n}}, \forall m \in \mathcal{M}, u_{m,n} \in \mathcal{N}_m,
\end{aligned} \tag{9}
$$

where $\mathcal{Z}_{u_{m,n}}$ is the offloading cost of the user's $u_{m,n}$ task. $\mathcal{Z}_{u_{m,n}}$ can be expressed by the following formula:

$$\mathcal{Z}_{u_{m,n}} = \mathcal{Z}^{l}_{u_{m,n}} + \sum_{k=1}^{K} x_{u_{m,n},k} \cdot \mathcal{Z}^{e}_{u_{m,n},k} \tag{10}$$

As indicated in the problem formulation, our objective is to minimize the total offloading cost of the users of the network. The first constraint ($C1$) ensures that each task is assigned to one cloudlet at most. Constraints ($C2$) guarantee that any static offloading decision task must be assigned to exactly one cloudlet and a dynamic offloading decision task may be assigned to one cloudlet at most. The next constraint ($C3$) shows that the QoS required by the task, in terms of completion time, must be less than a given threshold. The threshold is obtained based on the characteristics of the mobile application [5, 7]. For example, for an interactive application, the user's perception—the duration of the submission of the task until receiving the response—is a well-used technique to determine the threshold [8]. Completion time can be expressed as follows:

$$T_{u_{m,n}} = T^{l}_{u_{m,n}} + \sum_{k=1}^{K} x_{u_{m,n},k} \cdot \left(T^{t}_{u_{m,n},k} + T^{e}_{u_{m,n},k}\right) \tag{11}$$

The next constraint ($C4$) ensures that every offloaded task must be performed by a cloudlet that meets the hardware and software required by the task. The constraint $C5$ shows that it is not possible to exceed the computing capacity of each cloudlet. Constraint $C6$ ensures that any decision variable is a binary variable.

Finally, constraints ($C7$) indicate that the ratio of the offloaded computation must be a real value between 0 and 1. And, ($C8$) ensures that each static offloading decision task must always be entirely offloaded.

Theorem 4.1. *Equation (9) is a Non-Linear Mixed Binary Problem (NLMBP) 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 AP and all tasks are static offloading decisions. 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 a 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 Reference [17]. Since the special case is NP-hard, Equation (9) is also NP-hard. □

Considering the NP-hardness of the problem, it is difficult to achieve an optimal solution. Next, we propose a simplification version of Equation (9) using Lagrangian relaxation and decomposition approaches.

4.2 Problem Decomposition

To solve the above problem, we need a decomposition approach. Decomposing a complex optimization problem consists of breaking it up into smaller ones, called subproblems, and solving each of the smaller ones separately. Unfortunately, the constraint $C5$ is considered a complicating constraint [9, 31], since it involves the local variables of more than one subproblem. Consequently, the decomposition of Equation (9) does not work in one step and the subproblem cannot be solved independently. For these kinds of complex problems, there are advanced decomposition techniques that solve the problem by iteratively solving a sequence of subproblems. In this article, we use one of the most popular decomposition techniques, Lagrangian relaxation [9, 31]. The idea of Lagrangian relaxation comes up in the context of using Lagrangian multipliers to decompose the problem; thus, we introduce the Lagrangian multipliers $\lambda = [\lambda_k, k \in \mathcal{K}]^T$ on the constraint $C5$, where λ_k denotes the price of all the tasks performed by the k^{th} cloudlet. X and A are the set of the offloading decision variables and the set of the offloading ratio, respectively. The Lagrangian function is given by:

$$\begin{aligned} L(X, A, \lambda) &= \sum_{m}^{M} \sum_{n}^{N_m} \mathcal{Z}_{u_{m,n}} + \sum_{k}^{K} \lambda_k \sum_{m}^{M} \sum_{n}^{N_m} (x_{u_{m,n},k} \cdot c_k - F_k), \\ &= \sum_{m}^{M} \sum_{n}^{N_m} \mathcal{Z}_{u_{m,n}} + \sum_{m}^{M} \sum_{n}^{N_m} \sum_{k}^{K} \lambda_k x_{u_{m,n},k} \cdot c_k - \sum_{k}^{K} F_k, \\ &= \sum_{m}^{M} \sum_{n}^{N_m} \left(\mathcal{Z}_{u_{m,n}} + \sum_{k}^{K} \lambda_k x_{u_{m,n},k} \cdot c_k \right) - \sum_{k}^{K} \lambda_k F_k. \end{aligned}$$

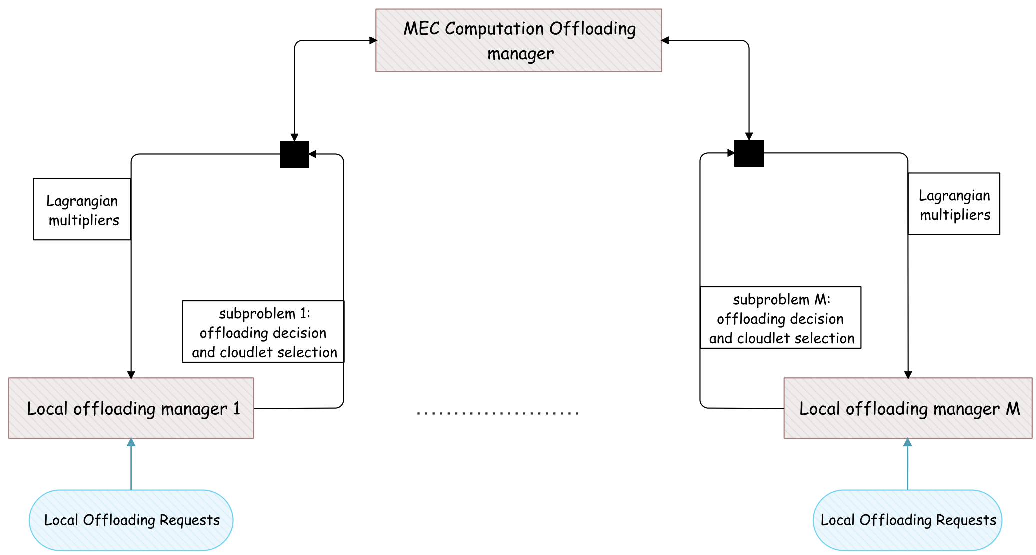


Fig. 4. DM2-ECOP architecture overview.

The Lagrangian dual problem for primal Equation (9) is then given by:

$$\max_{\lambda} \min_{X,A} L(X, A, \lambda).$$

We can see that the Lagrangian dual problem is separable into two levels: Level 1 is the inner minimizing, which consists of M subproblems, each one concerning only one AP. Level 2 is outer maximization, which is the master problem that considers the global variables and constraint $C5$.

Focusing on this observation, we introduce a new offloading policy named Distributed Multi-user Multi-cloudlet Efficient Computation Offloading Policy (DM2-ECOP). In the following, we describe the proposed computation offloading policy.

5 DM2-ECOP: DISTRIBUTED MULTI-USER, MULTI-CLOUDLET EFFICIENT COMPUTATION OFFLOADING POLICY

As introduced in the last section, the Lagrangian dual problem is decomposable into M subproblems. Each subproblem tries to find an optimal offloading decision and cloudlet selection to the users associated with one AP. Considering this characteristic, we design a new offloading policy, DM2-ECOP. As shown in Figure 4, DM2-ECOP has two levels of decision. The `local offloading manager` is responsible for the offloading decision and cloudlet selection of an AP; it solves the associated subproblem, then sends the offloading decision and the cloudlet selection to the centralized decision level. The `MEC computation offloading manager` receives the solution of all subproblems, then it ensures that the obtained offloading solution is feasible and respects all the constraints. After, it updates the Lagrangian multipliers and transmits the new values to every `local offloading manager` to improve the local solutions.

5.1 Local Offloading Manager: A Greedy Best Cloudlet Selection First Heuristic(GBC-SFH)

The local offloading manager tries to solve the subproblem of one AP to decide which user can offload. Then, it selects the appropriate cloudlet to perform the task of each user. According to the previous considerations, we can formulate the subproblem of a local offloading manager as follows:

$$\textbf{Minimize} \sum_{n}^{N_m} \left(\mathcal{Z}_{u_{m,n}} + \sum_{k}^{K} \lambda_k x_{u_{m,n},k} \cdot c_k \right)$$
$$\textbf{Subject to:}$$
$$\text{constraints C1} - \text{C4 and C6–C8.} \tag{12}$$

To solve the subproblem, we need to know the bandwidth allocated to each user. Unfortunately, this bandwidth depends on the number of users that offload their tasks. Therefore, we need to know the bandwidth allocation to decide whether a user should offload its task or not. To overcome this dependency problem, we use a branching heuristic. The key idea is that for any AP m, the number of users that can offload their tasks is between an upper and a lower bound. The lower bound corresponds to the number of users with static offloading decision tasks, and the upper bound corresponds to the maximum number of users associated with the AP m, N_m.

Definition 5.1. The **offloading capacity** of the AP m is defined as the number of tasks that have been accepted being performed by the MEC environment. We note it by π_m, and it is given by:

$$\pi_m = \sum_{n}^{N_m} \sum_{k}^{K} x_{u_{m,n},k}.$$

The strategy of solving the subproblem is very simple: It consists of finding the optimal π_m that gives the minimal offloading cost. We add the constraint $C9$ to the subproblem (12):

$$C9 : \sum_{n}^{N_m} \sum_{k}^{K} x_{u_{m,n},k} = \pi_m. \tag{13}$$

To achieve a good and fast offloading decision, the local offloading manager uses greedy heuristics to solve the subproblem (12). The Greedy Best Cloudlet Selection First Heuristic (GBC-SFH) offers heuristics to determine which users offload, to determine the computation to offload, and selects the cloudlet to perform each offloaded task. GBC-SFH iterates all possible values of the offloading capacity, π_m, in an increasing order, as illustrated in Algorithm 1. In brief, the idea is to find the best cloudlet selection for all static offloading tasks in the first step by minimizing the Lagrangian cost $\mathcal{Z}^e_{u_{m,n},k} + \lambda_k c_k$ under the constraints $C1 - C4$ and $C6 - C8$. So, each static offloading task is offloaded to the cloudlet that minimizes the Lagrangian cost.

For each dynamic offloading decision task, GBC-SFH tries to select the best cloudlet and compute the optimal ratio $a_{u_{m,n}}$ of the computation to offload. According to the resource availability, GBC-SFH can offload the task or perform it locally, $a_{u_{m,n}} = 0$, by the user's device. Since the wireless bandwidth at the AP may not be enough to offload all the dynamic offloading decision tasks, we need to define an order to determine which dynamic offloading decision task is preferred for the offloading. To this purpose, we define an offloading priority for each task according to the following formula:

$$\xi_{u_{m,n}} = \mathcal{Z}^l_{u_{m,n}} - \min_{k \in \mathcal{K}} (\mathcal{Z}^e_{u_{m,n},k}); \quad \text{under } a_{u_{m,n}} = 1.$$

Here, the offloading priority is the local cost minus the cost when all the computation is offloaded to the best cloudlet. The idea is that where $\xi_{u_{m,n}}$ is going higher, the user $u_{m,n}$ is more preferred to offload its task. Unlike the static offloading decision tasks, for dynamic offloading decision tasks, we need to compute the computation to offload ($a_{u_{m,n}}$). To this end, GBC-SFH uses a two-step method. In the first step, it selects the best cloudlet to offload the computation of the user $u_{m,n}$. At this step, GBC-SFH chooses the cloudlet that minimizes the Lagrangian cost $\mathcal{Z}^e_{u_{m,n},k} + \lambda_k c_k$ under the constraints $C1 - C4$, $C6 - C7$, and $a_{u_{m,n}} = 1$. After the selection of the best cloudlet, GBC-SFH computes the optimal value of $a_{u_{m,n}}$ for the current user. The optimal value of $a_{u_{m,n}}$ is the solution to the following problem:

$$\min\left(\mathcal{Z}^{e}_{u_{m,n},k} + \mathcal{Z}^{l}_{u_{m,n}}\right)$$
$$\textbf{Subject to: } \text{constraint } C7. \tag{14}$$

Equation (14) is a simple problem with one variable. Its optimal solution can be achieved by derivative sign rules [28]. Theorem 5.2 shows when the minimum of this problem will be achieved.

Theorem 5.2. *Let us define $\psi_{u_{m,n}}$ and $\mu_{u_{m,n}}$ the upload data-computing ratio of the dynamic offloading decision task, and the local-remote offloading cost ratio of the user $u_{m,n}$, respectively. They are given as follows:*

$$\psi_{u_{m,n}} = \frac{up_{u_{m,n}}}{\gamma_{u_{m,n}}},$$
$$\mu_{u_{m,n}} = \frac{w_{u_{m,n}} \cdot [\kappa \cdot f^3_{u_{m,n}} \cdot c_k \cdot \beta_{u_{m,n}} + (1-\beta_{u_{m,n}}) \cdot (c_k - f_{u_{m,n}}) - \beta_{u_{m,n}} \cdot P^{idle}_{u_{m,n}} \cdot f_{u_{m,n}}]}{c_k \cdot f_{u_{m,n}} \cdot (P^{tx/rx}_{u_{m,n}} \cdot \beta_{u_{m,n}} + 1 - \beta_{u_{m,n}})}$$

The minimum of Equation (14) is achieved when:

- *$a_{u_{m,n}} = 1$, if and only if:*
$$\psi_{u_{m,n}} < \mu_{u_{m,n}}$$
- *$a_{u_{m,n}} = 0$, if and only if:*
$$\psi_{u_{m,n}} > \mu_{u_{m,n}}$$
- *the problem is constant, if and only if:*
$$\psi_{u_{m,n}} = \mu_{u_{m,n}}.$$

Proof. The proof of Theorem 5.2 is detailed in the appendix. □

Using Theorem 5.2, we have three possible scenarios for dynamic offloading decision tasks: (1) when $a_{u_{m,n}} = 1$, the whole computation must be offloaded; (2) when $a_{u_{m,n}} = 0$, there is no offloading; and 3) when the Equation (14) is constant, all possible values of $a_{u_{m,n}}$ give the same performance. As the computation offloading is a solution to improve the performance, we choose to not offload, $a_{u_{m,n}} = 0$ when this case occurs. Once the number of the offloading tasks is equal to the current offloading capacity (π_m), the remaining tasks are assigned to be performed locally by the user's device.

Consequently, in the worst case, the GBC-SFH iterates N_m in the outer loop when there is no static offloading decision task, which means a complexity of $N_m \cdot log(N_m)$ to sort the tasks and $N_m \cdot K$ to assign to task at the inner loop. Thus, the maximum total number of iterations is $N^2_m \cdot K + N^2_m \cdot log(N_m)$. Therefore, the complexity of GBC-SFH is $O(N^2_m \cdot log(N_m))$, which is fast especially when the number of users associated to each AP is small [14, 21, 33] (≤20).

5.2 MEC Computation Offloading Manager

The outer level of the Lagrangian dual problem is the master problem. It ensures a feasible offloading solution of the primal Equation (9). Finding the optimal solution of the Lagrangian dual problem requires an exhaustive search of all solutions' space and Lagrangian multiplier values, which is a difficult task in general [9]. Consequently, we need to adopt a faster approach. In this work, we use the Subgradient-based heuristic [31]. The proposed heuristic used in `the MEC computation offloading manager` has three steps, as illustrated in Algorithm 2. First, it solves the subproblems

ALGORITHM 1: *The local offloading manager: GBC-SFH*

Input:
Π_m: Set of offloading capacity;
Output: Output the offloading decision $\mathcal{X}$, ratio $\mathcal{A}$, and cost $\mathcal{Z}$;
Sort Π_m in increasing order;
for $\pi_m \in \Pi_m$, **do**
    allocate bandwidth using Equation (1);
    offload each static offloading decision task to the cloudlet k that minimizes $\mathcal{Z}^e_{u_{m,n},k} + \lambda_k c_k$ under constraints $C1 - C4$ and $C6 - C8$;
    $nb_{\mathit{offloaded\_task}}$ = number of static offloading decision tasks;
    compute $\xi_{u_{m,n}}$ for every dynamic offloading decision task;
    Sort dynamic offloading decision tasks in decreasing order of ξ_{n_m};
    while $nb_{\mathit{offloaded\_task}} \leq \pi_m$, **do**
        Select the cloudlet k that minimizes $\mathcal{Z}^e_{u_{m,n},k} + \lambda_k c_k$ under constraints $C1 - C4$, $C6 - C7$, and $a_{u_{m,n}} = 1$;
        Compute the optimal value of $a_{u_{m,n}}$ using Theorem 5.2;
        if $a_{u_{m,n}} == 0$, **then**
            there is no offloading. This dynamic Offloading task must be performed locally;
        else
            Offload this dynamic Offloading task to the cloudlet k;
            $nb_{\mathit{offloaded\_task}} + +$;
        end if
        if (there is no more task) and ($nb_{\mathit{offloaded\_task}} < \pi_m$), **then**
            break the while-loop. There is no feasible solution for this value of π_m;
        end if
    end while
    all the remaining tasks must be performed locally;
    update the best offloading cost $\mathcal{Z}$, ratio $\mathcal{A}$ and decision $\mathcal{X}$;
end for

in the local offloading manager by the GBC-SFH for the current Lagrangian multipliers λ. Next, `the MEC computation offloading manager` checks if they obtained an offloading solution that is not feasible. If so, the Lagrangian Adjustment Heuristic (LAH) will be used the get a feasible solution using a local search. The idea of LAH heuristics is to check if every cloudlet respects the constraint $C5$. When a cloudlet does not respect this constraint, LAH heuristic reassigns some tasks offloaded from this cloudlet to another cloudlet that respects all constraints.

At the end, `the MEC computation offloading manager` updates the Lagrangian multipliers by the following formula:

$$\lambda_k(t+1) = \lambda_k(t) + \theta(t) \cdot \left(\sum_m^M \left(\sum_n^{N_m} x_{u_{m,n},k} \cdot c_k \right) - F_k \right), \tag{15}$$

where $\theta(t)$ is the update step. In this work, we use the Held and Karp formula [9, 31] to update this step as follows:

$$\theta(t) = \eta(t) \cdot \frac{\mathcal{Z}^* - \mathcal{Z}(t)}{\sum_{k=1}^K (\sum_m^M \sum_n^{N_m} x_{u_{m,n},k} \cdot c_k - F_k)^2}, \tag{16}$$

ALGORITHM 2: *MEC computation offloading manager*

Input:

It_{max}: maximum number of iterations;
ε: an infinitesimal number;

Output: offloading decision and cloudlet selection for all users;

```
Initialize λ_k randomly;
Z_max = −∞;
while (t < It_max and θ(t) > ε), do
    for (m ∈ M), do
        Z_m(t)= get the solution of subproblem m from the local offloading manager m;
    end for
    Z(t) = ∑_{m∈M} Z(t)_m − ∑_{k∈K} λ_k · F_k;
    if (Z(t) > Z_max), then
        Z_feasible = use Heuristic LAH to find a feasible solution;
        if (Z_feasible < Z*), then
            Z* = Z_feasible;
            update the best solution of the primal problem;
        end if
        Z_max = Z(t);
    end if
    update the Lagrangian multipliers and η using Equations (15), (16), and (17);
end while
```

where $\eta(t)$ is a decreasing adaptation parameter $0 < \eta(0) \leq 2$, $\mathcal{Z}^*$ is the best obtained solution of Equation (9), and $\mathcal{Z}(t)$ refers to the current solution of the Lagrangian dual problem. $\eta(t)$ can be expressed by the following formula:

$$\eta(t+1) = \begin{cases} \vartheta \cdot \eta(t) & \text{if } \mathcal{Z}(t) \text{ did not increase} \\ \eta(t) & \text{otherwise} \end{cases}. \tag{17}$$

As suggested in References [9, 31], we set the values of $\vartheta = 0.9$ and $\eta(0) = 2$. The master problem repeats these steps until the stop conditions, which are the maximum number of iterations It_{max} and the maximum tolerated error of the update step ε ($\theta \leq \varepsilon$).

6 NUMERICAL RESULTS

In this section, we evaluate the performance of DM2-ECOP using the characteristics of realistic system configuration. We use an MEC environment consisting of a metropolitan area, which is composed of 20 APs forming a ring topology. The delay between any two APs is 3*ms* and the delay between every AP and the remote cloud is 100*ms* [14, 33]. We suppose that four cloudlets are equidistantly deployed among the network, i.e., cloudlet 1 is collocated with the AP 1, cloudlets 2 with the AP 6, cloudlet 3 with the AP 11, and cloudlet 4 with the AP 16. To study the performance of our offloading policy, we consider four cloudlet configurations. Table 2 illustrates the list of the cloudlets' configurations considered for our tests. We consider real configurations used by public cloud providers, such as, Amazon Web Services (AWS) and Microsoft Azure [5, 12, 21], to simulate the behavior of DM2-ECOP policy for real-world scenarios.

The wireless bandwidth of each AP is 150Mbps. The bandwidth allocated to each user is estimated using the parameter settings used in the Bianchi model [1]. Similar to Reference [33], we

Table 2. List of the Cloudlets' Configurations Used for the Tests

Configuration	Computing capacity F_k and allocation c_k in Giga CPU cycles/s							
	cloudlet 1		cloudlet 2		cloudlet 3		cloudlet 4	
	c_1	F_1	c_2	F_2	c_3	F_3	c_4	F_4
configuration 1	10	1,000	10	1,000	10	1,000	10	1,000
configuration 2	15	1,000	10	1,000	15	1,000	10	1,000
configuration 3	10	500	10	500	10	1,500	10	1,500
configuration 4	15	500	10	500	15	1,500	10	1,500

Table 3. The Characteristic of the Real-world Applications Used for Our Tests

Application	$\gamma_{u_{m,n}}$ (Giga CPU cycles)	$up_{u_{m,n}}$ (Kilobyte)	$dw_{u_{m,n}}$ (Byte)	$t_{u_{m,n}}$ (Second)
	static offloading decision tasks			
FACE	12.3	62	60	5
SPEECH	15	243	50	5.1
OBJECT	44.6	73	50	13
	dynamic offloading decision tasks			
Linpack	50	10,240	120	62.5
CPUBENCH	3.36	80	80	4.21
PI BENCH	130	10,240	200	163

assume that the number of users connected to every AP, N_m is not greater than 20. Precisely, N_m takes values from $\{5, 10, 15, 20\}$. Each user runs one application from Table 3. The first three applications are static offloading decision tasks; the others are dynamic offloading decision tasks [11]. We assume that $P^{tx/rx}_{u_{m,n}} = 10 * P^{idle}_{u_{m,n}}$ and $P^{idle}_{u_{m,n}} = 100$mW, as shown in Reference [3]. The local computing capability of each user was randomly chosen from $f_{u_{m,n}} \in [0.8, 1, 1.2]$ gigacycles.

The performances of DM2-ECOP are compared to two offloading policies from the literature:

- Nearest Cloudlet Offloading (NCO) [14, 33]: in which each AP is associated with the nearest cloudlet. So, all the users connected to this AP offload their tasks to the same cloudlet. When a cloudlet is overloaded, the tasks are migrated to another cloudlet.
- Full Cloud Offloading (FCO) [4, 5]: In this case, the users offload their tasks to the remote cloud. To make sense of the performances comparison of the offloading policies DM2-ECOP, NCO, and FCO, we assume that the computing capacity allocated to perform each offloading task in the remote cloud is 10 gigacycles.

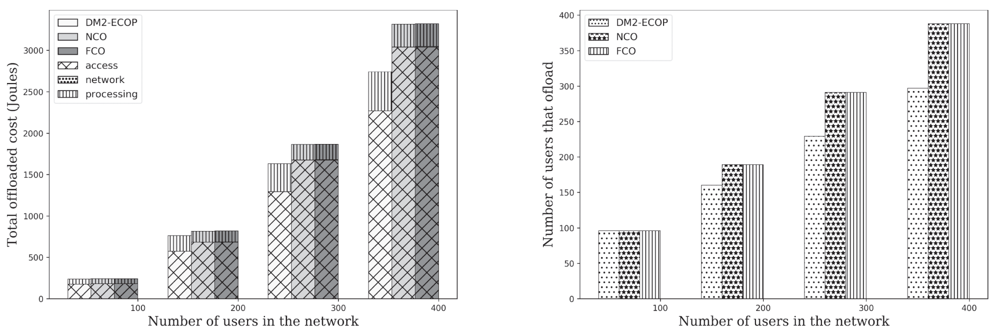
In the following, the default cloudlet configuration is the first configuration (configuration 1), and the density of users at each AP is considered as the same, i.e., the same number of users at each AP. Furthermore, the stop criteria of the MEC computation offloading manager for DM2-ECOP are $It_{max} = 100$ for the maximal number of iterations, and $\varepsilon = 10^{-20}$ for the maximum tolerated error of update steps.

6.1 Convergence of DM2-ECOP

To evaluate the performance of DM2-ECOP and its convergence to a feasible solution, Table 4 depicts the required number of iterations to get a feasible solution and the last value of update steps (θ). As expected, the required number of iterations increases as the number of users in the

Table 4. Number of Iterations and Update Step Taken by DM2-ECOP to Converge to a Feasible Solution

Number of users	Number of iterations	Update step θ
100	15	0.0
200	20	9.12×10^{-22}
300	29	5.09×10^{-21}
400	43	4.46×10^{-21}



(a) total offloading cost (b) number of users that offload

Fig. 5. Comparison of offloading policies DM2-ECOP, NCO, and FCO where the cost parameter $\beta = 1$.

network increases. Moreover, the update step converges to the maximum tolerated error ε with a few number of iterations; this convergence changes while the number of users increases. Thanks to the Held and Karp formula used in our work to update the Lagrangian update step, for the rapid convergence of DM2-ECOP offloading policy.

6.2 Offloading Performance Comparison

Figure 5 plots the offloading performances of DM2-ECOP, *NCO*, and *FCO* if we set the cost parameter $\beta = 1$. We also distinguish between the costs related to the network access, network backhaul, and processing. We note that DM2-ECOP reduces the total offloading cost compared to NCO and FCO. More precisely, we can see that the access cost of DM2-ECOP is the lowest, but the processing cost is the highest. This is due to the bandwidth allocation heuristic used by DM2-ECOP, which tries to maximize the bandwidth allocated to each user by minimizing the offloading capacity (π) of each AP. So, less users can offload their tasks to the MEC with DM2-ECOP compared to *NCO* and *FCO*. However, where the wireless bandwidth is enough to offload all tasks, DM2-ECOP and *NCO* are equivalent, as shown in Table 4 where 100 users are in the network.

To understand the effect of user density on offloading performances, we investigate in Figure 6 the offloading gain of DM2-ECOP compared to NCO under different user density. We consider four scenarios where the topology is divided into two regions, each one containing 10 APs. In Scenario 1, the regions have the same user density. In Scenario 2, user density in Region 1 is twice the user density in Region 2. In Scenario 3, user density in Region 1 is three times the user density in Region 2. Finally, in Scenario 4, user density in Region 1 is four times the user density in Region 2. In Figure 6, we note that the offloading gain of DM2-ECOP compared to NCO goes up when user density goes high. For example, where 200 users are in the network, the gain is 6.1% for Scenario 1 and 10.5% for Scenario 4. This is because the cloudlet selection in *NCO* is static, so this policy needs to migrate some tasks where the cloudlet is overloaded. Consequently, it adds an extra offloading

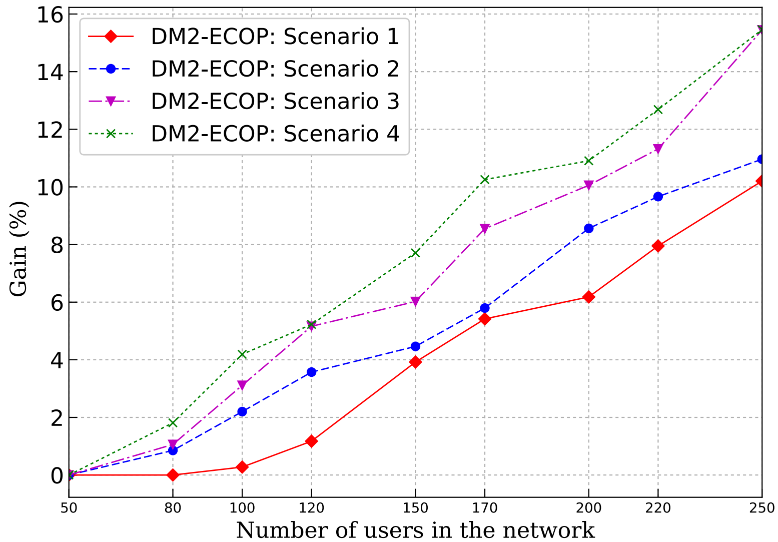
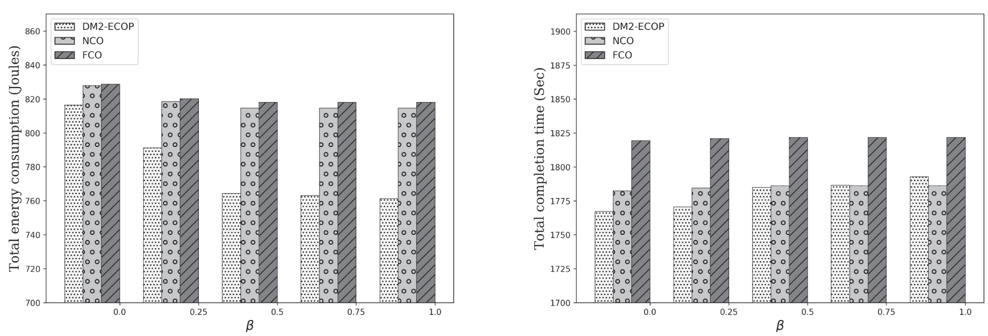


Fig. 6. Comparison of DM2-ECOP and NCO for different user density in the network.



(a) total energy consumption (b) total completion time

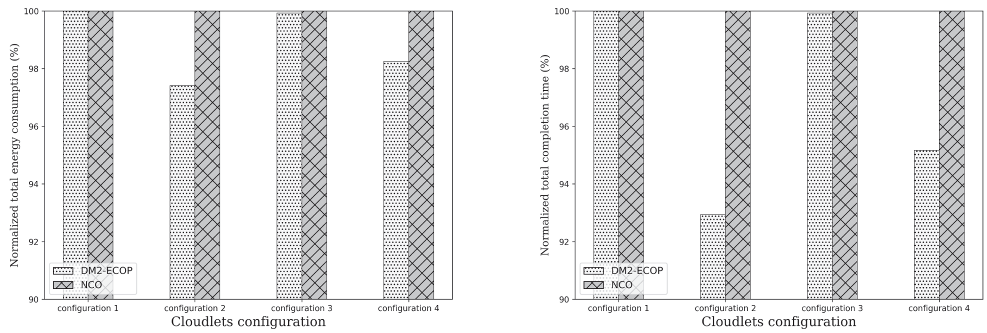
Fig. 7. Comparison of offloading policies DM2-ECOP, NCO, and FCO over the parameter β, where 200 users are in the mobile edge computing environment.

cost. However, DM2-ECOP tries to find the best cloudlet selections dynamically at the offloading decision, according to the system and network resource availability.

6.3 Impact of the Cost Parameter β on the Offloading Performance

Figure 7 studies the effect of the offloading cost parameter β on the performance of the policies DM2-ECOP, *NCO*, and *FCO*. As we can see in Figure 7(a), the energy consumption of DM2-ECOP is better than *NCO* and *FCO* for all possible values of β. Indeed, even if we set β to 0, which means that we give a complete priority to the tasks' completion time, DM2-ECOP obtains better performances. Moreover, when we increase the value of β, the obtained performances are even better. Consequently, DM2-ECOP achieves better performance, in terms of energy consumption, whatever the offloading cost: energy consumption, completion time, or a combination of energy and time. This is because of the dynamic cloudlet selection adopted in DM2-ECOP.

In Figure 7(b), we investigate the effect of β on the performance in terms of completion time. We note that the completion time of DM2-ECOP and NCO are better than FCO, because the cloudlets



(a) total energy consumption (b) total completion time

Fig. 8. Offloading policies' performances over different cloudlet configurations, where $\beta = 1$ (the offloading cost is energy consumption) and 200 users are in the MEC environment.

are close to users. Moreover, the completion time of DM2-ECOP is the lowest where $\beta = 0$. However, NCO achieves better completion time than DM2-ECOP where β closes to 1. Consequently, NCO has the best performance in terms of completion time. In fact, when β is close to 1, the energy consumption becomes more important than the completion time in the expression of the offloading cost. DM2-ECOP reduces the offloading cost by offloading less tasks to the MEC, as shown in Figure 5, to minimize the energy consumed by the wireless access level. As a result, more tasks are executed locally, which increases the completion time.

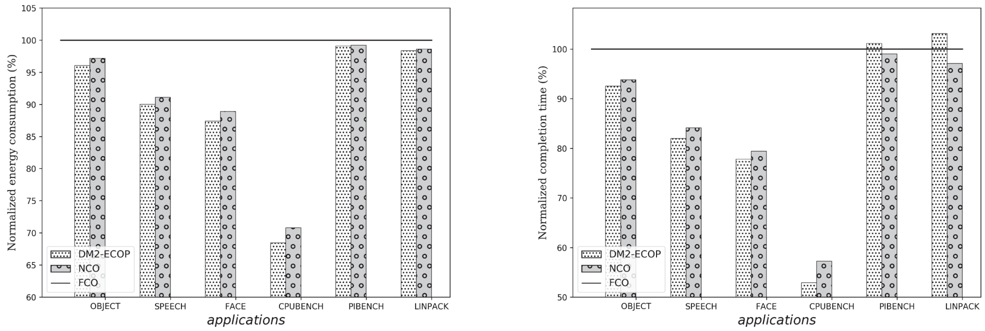
6.4 Impact of the Cloudlets' Configurations

In the following, we study the performance of the offloading policies over heterogeneous cloudlet configurations. In Figure 8, we investigate the performance of the offloading policies DM2-ECOP and *NCO* over four cloudlet configurations presented at the beginning of this section. We observe that the DM2-ECOP and *NCO* are equivalent when the cloudlets have exactly the same configurations, such as Configuration 1. However, where the computing resources allocated to each task are heterogeneous, which corresponds to a more realistic scenario, DM2-ECOP achieves better performance in terms of energy consumption and completion time. This can be explained by the fact that DM2-ECOP offloads to the best cloudlet, but *NCO* offloads to the nearest cloudlet. We also can notice that DM2-ECOP gets better performance for Configuration 2 than Configuration 4, even if the resource allocated for each task are the same in the two configurations. This is because the total computing capacity in Configuration 4 is not homogeneous. Thus, NCO needs to migrate some tasks from Cloudlets 1 and 2 to Cloudlets 3 and 4. To summarize, these results show that DM2-ECOP can achieve a good offloading performance under different cloudlet configurations.

6.5 Impact of the Applications Characteristics

As shown previously in our analytical model, the characteristics of the application have a crucial role in the efficiency of the offloading performance. To understand this role, we investigate the impact of the application on the offloading cost and on the amount of the offloaded computation, $a_{u_{m,n}}$.

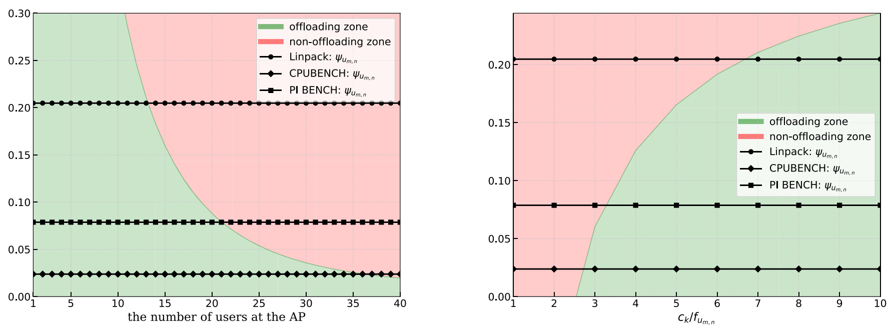
Figure 9 depicts the performances of the offloading policies for each application under the cloudlets' Configuration 3. We note that for static offloading decision tasks, DM2-ECOP has better energy consumption and completion time followed by *NCO*. Indeed, the static offloading decision tasks must be offloaded, and DM2-ECOP tries to find the best cloudlet selection at the decision



(a) total energy consumption

(b) total completion time

Fig. 9. Offloading policies' performances for each application for Configuration 3, where $\beta = 1$ (the offloading cost is energy consumption) and 200 users are in the MEC environment.



(a) Offloading decision vs number of users and local-remote offloading cost ratio

(b) Offloading decision vs number of users and local-remote offloading cost ratio

Fig. 10. The effect of the number of users, $w_{u_{m,n}}$, the considered offloading cost $\beta_{u_{m,n}}$, and the allocation computing resource c_k on the application partition decision, where the wireless bandwidth at AP is 150mbps, local cpu power is $f_{u_{m,n}} = 1$ gigacycle.

time. However, FCO offloads to the nearest cloudlet, which induces an additional offloading cost, since some tasks must be migrated to other cloudlets. However, for the dynamic offloading decision tasks, we note that DM2-ECOP has the lowest energy consumption and completion time where the application does not need to upload lots of data to the cloudlet, such as CPUBENCH. However, where the application requires lots of data, e.g., Linpack and PIBENCH, the completion time of *NCO* and *FCO* are better than that of DM2-ECOP. This is because DM2-ECOP tries to perform the application locally when it uploads a lot of data to minimize the access cost of the users.

Finally, to deeply analyze the performance of DM2-ECOP with the dynamic offloading decision tasks, we study the effect of the number of users and the ratio between the remote and local processing capacity on the offloading decision. In Figure 10(a), we plot the effect of the number of users per access point and, thus, the amount of bandwidth allocated to each user on the offloading decision. The red area corresponds to the case where we execute a task only locally (i.e., $a_{u_{m,n}} = 0$), and the green area corresponds to the case where a task is totally offloaded (i.e., $a_{u_{m,n}} = 1$). In addition to the offloading decision, we also plot the upload data-computing ratio $\psi_{u_{m,n}}$ for three dynamic offloading decision tasks, namely Linpack, CPUBENCH, and PI BENCH. As we can see, the three applications do not behave the same way when we increase the number of users per AP. Indeed, Linpack is the most sensitive application and stops offloading when the number of users is greater than 14. However, CPUBENCH is the less-sensitive application, since it stops offloading when the number of users reaches 37. This is due to the fact that Linpack sends much more data when it offloads compared to CPUBENCH. In Figure 10(b), we investigate the impact of the ratio between the local and remote processing capacity on the offloading decision. As in the last figure, we also plot the upload data-computing ratio $\psi_{u_{m,n}}$ for Linpack, CPUBENCH, and PI BENCH. As we can see, more the remote processing capacity is important compared to the local processing capacity more the decision is to offload task. However, as in the last figure, Linpack is less sensitive to that increase compared to CPUBENCH and PI BENCH. Indeed, since the total amount of data that should be offloaded for Linpack is important, the offloading becomes beneficial only if the remote processing capability is very important in comparison to the local processing capability.

7 CONCLUSION

Computation offloading in a multi-user, multi-cloudlet mobile edge computing environment is a challenging issue. In this article, we propose a new computation offloading policy to decide which users should offload and to which cloudlet. First, we formulate the problem as a Non-Linear Mixed Binary Integer Program. Then, we propose an efficient distributed heuristic to solve the problem using the Lagrangian decomposition approach. The proposed heuristic uses a branching algorithm to maximize the bandwidth allocation and minimize the offloading cost.

In addition, compared to other works, our proposal (DM2-ECOP) considers two categories of offloadable tasks: the static offloading decision tasks that must be performed remotely and the dynamic offloading decision tasks that can be performed both locally and remotely. We also add an offloading computation ratio associated with both static and dynamic decision tasks. This ratio denotes the portion of the application that is executed locally in the terminal and the portion of the application that should be offloaded to the cloudlet.

The obtained numerical results show performance improvements in terms of the offloading cost compared to existing offloading policies under different scenarios and cloudlet configurations. Moreover, because we consider that all the tasks have the same priority and they are not sharing resources (same CPU) at the cloudlet, we demonstrate that the best possible value of the offloading computation ratio is either 0 or 1.

For future work, we will consider an adaptive offloading policy, where the offloaded tasks must be determined at runtime. Moreover, in this article, we assume that each mobile is executing only one task at a time. In future work, we propose to explore the case where an application is characterized by a tasks dependency graph. In this case, more than one task can be offloaded at the same time to the remote cloudlet or cloud.

APPENDIX: PROOF OF THEOREM 5.2

Given the objective function of Equation (14), which is a function with the variable $a_{u_{m,n}}$, we can find the minimum following the derivative sign rules [28]. Let $\mathcal{F}_{u_{m,n}}$ be the derivative of the objective function, $\mathcal{Z}^e_{u_{m,n},k} + \mathcal{Z}^l_{u_{m,n}}$. $\mathcal{F}_{u_{m,n}}$ is given as follows:

$$\mathcal{F}_{u_{m,n}} = up_{u_{m,n}} \cdot \frac{P^{tx/rx}_{u_{m,n}} \cdot \beta_{u_{m,n}} + 1 - \beta_{u_{m,n}}}{w_{u_{m,n}}} + \gamma_{u_{m,n}} \cdot \left(\frac{\beta_{u_{m,n}} \cdot P^{idle}_{u_{m,n}}}{c_k} + \frac{1 - \beta_{u_{m,n}}}{c_k} - \frac{1 - \beta_{u_{m,n}}}{f_{u_{m,n}}} - \kappa \cdot f^2_{u_{m,n}} \cdot \beta_{u_{m,n}} \right).$$

The derivative function $\mathcal{F}_{u_{m,n}}$ is a constant and does not change when the variable $a_{u_{m,n}}$ does. According to the derivative sign rules [28], we have three cases:

Case 1: $\mathcal{F}_{u_{m,n}} < 0$, there the objective function of Equation (14) is monotonically decreasing. Consequently, its minimum is achieved at $a_{u_{m,n}} = 1$. This case occurs when:

$$\frac{up_{u_{m,n}}}{\gamma_{u_{m,n}}} < \frac{w_{u_{m,n}} \cdot \left[\kappa \cdot f_{u_{m,n}}^3 \cdot c_k \cdot \beta_{u_{m,n}} + (1-\beta_{u_{m,n}}) \cdot (c_k - f_{u_{m,n}}) - \beta_{u_{m,n}} \cdot P_{u_{m,n}}^{idle} \cdot f_{u_{m,n}}\right]}{c_k \cdot f_{u_{m,n}} \cdot \left(P_{u_{m,n}}^{tx/rx} \cdot \beta_{u_{m,n}} + 1 - \beta_{u_{m,n}}\right)}.$$

Case 2: $\mathcal{F}_{u_{m,n}} > 0$, there the objective function of Equation (14) is monotonically increasing. Consequently, its minimum is achieved at $a_{u_{m,n}} = 0$. This case occurs when:

$$\frac{up_{u_{m,n}}}{\gamma_{u_{m,n}}} > \frac{w_{u_{m,n}} \cdot \left[\kappa \cdot f_{u_{m,n}}^3 \cdot c_k \cdot \beta_{u_{m,n}} + (1-\beta_{u_{m,n}}) \cdot (c_k - f_{u_{m,n}}) - \beta_{u_{m,n}} \cdot P_{u_{m,n}}^{idle} \cdot f_{u_{m,n}}\right]}{c_k \cdot f_{u_{m,n}} \cdot \left(P_{u_{m,n}}^{tx/rx} \cdot \beta_{u_{m,n}} + 1 - \beta_{u_{m,n}}\right)}.$$

Case 3: $\mathcal{F}_{u_{m,n}} = 0$, there the objective function of Equation (14) is constant and does not change. So, at the values of $a_{u_{m,n}}$ give the same cost. This case occurs when:

$$\frac{up_{u_{m,n}}}{\gamma_{u_{m,n}}} = \frac{w_{u_{m,n}} \cdot \left[\kappa \cdot f_{u_{m,n}}^3 \cdot c_k \cdot \beta_{u_{m,n}} + (1-\beta_{u_{m,n}}) \cdot (c_k - f_{u_{m,n}}) - \beta_{u_{m,n}} \cdot P_{u_{m,n}}^{idle} \cdot f_{u_{m,n}}\right]}{c_k \cdot f_{u_{m,n}} \cdot \left(P_{u_{m,n}}^{tx/rx} \cdot \beta_{u_{m,n}} + 1 - \beta_{u_{m,n}}\right)}.$$

Ending of the proof. □

REFERENCES

[1] Giuseppe Bianchi. 2000. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE J. Select. Areas Comm.* 18, 3 (2000), 535–547.

[2] Arash Bozorgchenani, Daniele Tarchi, and Giovanni Emanuele Corazza. 2017. An energy and delay-efficient partial offloading technique for fog computing architectures. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'17)*. IEEE, 1–6.

[3] Aaron Carroll, Gernot Heiser. 2010. An analysis of power consumption in a smartphone. In *Proceedings of the USENIX Annual Technical Conference (USENIXATC'10)*, Vol. 14. Boston, MA, 21–21.

[4] Meng-Hsi Chen, Ben Liang, and Min Dong. 2016. Joint offloading decision and resource allocation for multi-user multi-task mobile cloud. In *Proceedings of the IEEE International Conference on Communications (ICC'16)*. IEEE, 1–6.

[5] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* 24, 5 (2016), 2795–2808.

[6] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. 2011. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the 6th European Conference on Computer Systems (EuroSys'11)*. ACM, 301–314.

[7] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. 2010. MAUI: Making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys'10)*. ACM, 49–62.

[8] Debessay Fesehaye, Yunlong Gao, Klara Nahrstedt, and Guijun Wang. 2012. Impact of cloudlets on interactive mobile cloud applications. In *Proceedings of the 16th IEEE International Enterprise Distributed Object Computing Conference (EDOC'12)*. IEEE, 123–132.

[9] Marshall L. Fisher. 2004. The Lagrangian relaxation method for solving integer programming problems. *Manag. Sci.* 50, 12-supplement (2004), 1861–1871.

[10] 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.

[11] Ying Gao, Wenlu Hu, Kiryong Ha, Brandon Amos, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2015. Are cloudlets necessary? October (2015).

[12] Songtao Guo, Bin Xiao, Yuanyuan Yang, and Yang Yang. 2016. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In *Proceedings of the 35th IEEE International Conference on Computer Communications (INFOCOM'16)*, Vol. 2016-July. IEEE, 1–9.

[13] Dong Huang, Ping Wang, and Dusit Niyato. 2012. A dynamic offloading algorithm for mobile computing. *IEEE Trans. Wireless Comm.* 11, 6 (2012). IEEE, 1991–1995.

[14] Mike Jia, Jiannong Cao, and Weifa Liang. 2015. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Trans. Cloud Comput.* 99 (2015).

[15] Mike Jia, Weifa Liang, Zichuan Xu, and Meitian Huang. 2016. Cloudlet load balancing in wireless metropolitan area networks. In *Proceedings of the 35th IEEE International Conference on Computer Communications (INFOCOM'16)*, Vol. 2016-July. IEEE, 1–9.

[16] Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F. O'Brien. 2013. Near-exhaustive precomputation of secondary cloth effects. *ACM Trans. Graph.* 32, 4 (July 2013), 87:1–87:8.

[17] Sven O. Krumke and Clemens Thielen. 2013. The generalized assignment problem with minimum quantities. *Euro. J. Op. Res.* 228, 1 (2013), 46–55.

[18] Grace Lewis, Sebastián Echeverría, Soumya Simanta, Ben Bradshaw, and James Root. 2014. Tactical cloudlets: Moving cloud computing to the edge. In *Proceedings of the IEEE Military Communications Conference.* IEEE, 1440–1446.

[19] Grace Alexandra Lewis. 2016. *Software Architecture Strategies for Cyber-foraging Systems.* Ph.D Dissertation, Carnegie Mellon University, Pittsburgh, PA.

[20] Jia-Liang Lu and Fabrice Valois. 2006. Performance evaluation of 802.11 WLAN in a real indoor environment. In *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking, and Communications (WiMob'06).* IEEE, 140–147.

[21] 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.

[22] Yuyi Mao, Jun Zhang, S. H. Song, and Khaled Ben Letaief. 2016. Power-delay tradeoff in multi-user mobile-edge computing systems. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'16).* IEEE, 1–6.

[23] Antti P. Miettinen and Jukka K. Nurminen. 2010. Energy efficiency of mobile clients in cloud computing. *HotCloud* 10 (2010), 4–4.

[24] Anwesha Mukherjee, Debashis De, and Deepsubhra Guha Roy. 2016. A power and latency aware cloudlet selection strategy for multi-cloudlet environment. *IEEE Trans. Cloud Comput.* 99 (2016), 1–14.

[25] Yucen Nan, Wei Li, Wei Bao, Flavia C. Delicato, Paulo F. Pires, Yong Dou, and Albert Y. Zomaya. 2017. Adaptive energy-aware computation offloading for cloud of things systems. *IEEE Access* 5 (2017), 23,947–23,957.

[26] Yucen Nan, Wei Li, Wei Bao, Flavia C. Delicato, Paulo F. Pires, and Albert Y. Zomaya. 2018. A dynamic tradeoff data processing framework for delay-sensitive applications in cloud of things systems. *J. Parallel and Distrib. Comput.* 112 (2018), 53–66.

[27] Deepsubhra Guha Roy, Debashis De, Anwesha Mukherjee, and Rajkumar Buyya. 2016. Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. *J. Supercomput.* (2016), 1–19.

[28] Mark Ryan. 2005. *Calculus Workbook for Dummies.* Wiley Publishing, Inc.

[29] Swetank Kumar Saha, Pratham Malik, Selvaganesh Dharmeswaran, and Dimitrios Koutsonikolas. 2016. Revisiting 802.11 power consumption modeling in smartphones. In *Proceedings of the 17th IEEE International Symposium on World of Wireless, Mobile, and Multimedia Networks (WoWMoM'16).* IEEE, 1–10.

[30] Mahadev Satyanarayanan, Grace Lewis, Edwin Morris, Soumya Simanta, Jeff Boleng, and Kiryong Ha. 2013. The role of cloudlets in hostile environments. *IEEE Pervas. Comput.* 12, 4 (2013), 40–49.

[31] Jiafu Tang, Chongjun Yan, Xiaoqing 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 Trans. Automat. Sci. Eng.* 11, 4 (2014), 1,110–1,121.

[32] Song Wu, Chao Niu, Jia Rao, Hai Jin, and Xiaohai Dai. 2017. Container-based cloud platform for mobile computation offloading. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS'17).* IEEE, 123–132.

[33] Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, and Song Guo. 2016. Efficient algorithms for capacitated cloudlet placements. *IEEE Trans. Parallel Distrib. Systems* 27, 10 (2016), 2,866–2,880.

[34] Hong Yao, Changmin Bai, Muzhou Xiong, Deze Zeng, and Zhangjie Fu. 2017. Heterogeneous cloudlet deployment and user-cloudlet association toward cost-effective fog computing. *Concurr. Comput.: Practice and Exper.* 29, 16 (2017).

[35] Chongyu Zhou, Chen-Khong Tham, and Mehul Motani. 2017. Online auction for truthful stochastic offloading in mobile cloud computing. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'17).* IEEE, 1–6.

[36] Qiliang Zhu, Baojiang Si, Feifan Yang, and You Ma. 2017. Task offloading decision in fog computing system. *China Comm.* 14, 11 (2017), 59–68.

Received December 2017; revised July 2018; accepted July 2018

Author Queries

Q1: AU: Please supply the CCS Concepts 2012 codes per the ACM style indicated on the ACM website. Please include the CCS Concepts XML coding as well.

Q2: AU: In Table 1, "the category belong the tasks" written as meant?

Q3: AU: Please specify figure number instead of "the last figure."

Q4: AU: Please clarify: "more the remote processing capacity is important compared to the local processing capacity more the decision is to offload task."

Q5: AU: Please specify figure number instead of "the last figure."

Q6: AU: Please supply publication source for Reference 11.