Offloading Energy Efficiency with Delay Constraint for Cooperative Mobile Edge Computing Networks

Thai T. Vu, Nguyen Van Huynh, Dinh Thai Hoang, Diep N. Nguyen, and Eryk Dutkiewicz
School of Electrical and Data Engineering, University of Technology Sydney, Australia

***Abstract*—In this paper, we introduce a cooperative edge network model which enables edge nodes to cooperate in sharing computing and radio resources with the aim of minimizing the total energy consumption for mobile users while meeting their delay requirements. To find the optimal task offloading decisions for mobile users, we first formulate the joint task offloading and resource allocation optimization problem. However, this optimization problem is a mixed integer non-linear programming (MINLP) with both binary and real variables, i.e., offloading decisions and resource allocations, respectively, that is an NP-hard problem and intractable to find the optimal solution. Thus, we introduce a relaxing approach which transforms the MINLP to a relaxed optimization problem with real variables. After proving that the relaxed problem is a convex problem, we propose two solutions, namely ROP which is adopted from the interior point method and IBBA which is developed from the branch and bound algorithm. Through the numerical results, we show that our proposed approaches allow minimizing the total energy consumption and meeting all delay requirements for mobile users.**

Keywords- Task offloading, mobile edge computing, resource allocation, MINLP, and branch-and-bound algorithm.

I. Introduction

The development of mobile applications and Internet-of-Things (IoT) networks has brought a great deal of benefits for human lives, but it also faces many challenges. In particular, mobile and IoT applications have been developed recently often require computations with high complexity, e.g., 3D rendering and image processing, and/or low delay constraints, e.g., interactive games and online object recognitions. However, mobile and IoT devices are usually limited by computing resources, battery life, and network connections, and thus advanced applications may not be able to implement on these devices in practice. Thus, mobile edge computing has been introduced as an effective solution to address this problem.

Mobile Edge Computing (MEC) is an emerging network architecture that "move" the cloud computing capabilities closer to the mobile users [1]. Specifically, in an MEC network, powerful computing devices, e.g., servers, are deployed at the edges of the mobile network to support hardware resource-constrained devices, e.g., mobile and IoT devices, to perform high complexity computational tasks. The deployment of MEC networks can save energy consumption, increase operation time, and reduce performance delay for smart devices through utilizing powerful resources of the edge nodes. Furthermore, this can reduce operation costs for mobile network operators up to 67% by reducing the total throughput and peak backhaul bandwidth consumption [2]. As a result, technical standards for MEC are being developed by the European Telecommunications Standards Institute to promote the development of MEC in future mobile networks [3].

However, an MEC node does not possess abundant computing resource as that of the public cloud, e.g., Amazon Web Services and Microsoft Azure. Additionally, although computation offloading demand from mobile users is usually high, not all computational tasks benefit by being offloaded to the edge node. Some tasks even consume more energy when being offloaded than processed locally due to the communication overhead, i.e., transmit requests and receive results. Consequently, joint task offloading and resource allocation to minimize energy consumption for mobile devices under the edge's resource constraints and delay requirements is the most important challenge in MEC networks [1].

In [4], the authors study an energy efficient computation offloading scheme in a multi-user MEC system. In particular, the authors first formulate an energy consumption optimization problem with explicit consideration of delay performance. Through analyzing the relationship between mobile users' demands and edge computing node's capacity, the authors then can derive the optimal offloading probability and transmit power for mobile users. Aiming to minimize the overall cost of energy, computation, and delay for all users, the authors in [5] introduce a joint offloading and resource allocation for computation and communication in an MEC network. Due to the NP-hard problem, the authors proposed a three-step algorithm including semidefinite relaxation, alternating optimization, and sequential turning. In addition, there are some other research works in the literature studying different approaches for jointly energy efficiency and delay management in MEC networks. For example, the authors in [6] present a computation offloading game model to address the distributed computation offloading decision problem for mobile users, and the authors in [7] introduce a computation offloading hierarchical model in which a task can be offloaded to an MEC node or a cloud server.

In this work, we study a cooperative MEC network in which edge nodes are deployed in the same area to support high complexity computation tasks of the mobile users. The edge nodes have different radio and computing resources, meanwhile mobile users have distinct computation tasks with various delay requirements. To minimize the total energy consumption for mobile users in the network and meet all tasks' delay requirements, we first formulate the joint task offloading and resource allocation optimization problem for all mobile users and edge nodes. Since the optimization problem is a mixed integral non-linear programming (MINLP) which is NP-hard and intractable to solve, we introduce a relaxing solution which converts binary decision variables to real values. We then prove that the relaxed optimization problem is a convex problem which can be solved by some effective methods,

e.g., the interior point method (IPM). Although the IPM can find the optimal solution for relaxed problem, the obtained decision variables are real numbers which may not be practical in implementing in the MEC network. In addition, when converting decision variables to real values, the complexity of optimization problem becomes higher, which is inefficiency to implement in MEC networks, especially when the number of variables is large. Therefore, in this paper, we introduce IBBA, an improvement of branch and bound algorithm to address the MINLP. The proposed IBBA allows not only finding optimal binary variables for offloading decisions, but also utilizing the characteristics of binary variables to reduce the complexity in finding the optimal solution. The extensive numerical results are then performed to demonstrate the efficiency of proposed solutions in terms of minimizing the total energy consumption for mobile devices and meeting delay requirements for offloading tasks.

II. System Model

A. Network Model

We consider a mobile edge computing network (MEC) with N mobile users, M cooperative edge nodes, and one cloud server as shown in Fig. 1. The set of mobile users and MEC nodes in the network are denoted by $\mathbb{N} = \{1, 2, \ldots, N\}$ and $\mathbb{M} = \{1, 2, \ldots, M\}$, respectively. Each mobile user has computing tasks which can be processed locally or offloaded to MEC nodes to execute. The time is slotted, and in each time slot, it is assumed that each mobile user can send one computing task to one of the edge nodes in the network. If a task is decided to be executed at an MEC node, the mobile user will send the requested task to the target edge node. After the task is performed at the edge node, the result will be sent back to the user. Note that if the edge node does not have sufficient computing resources or it cannot meet the delay constraint of the task, the edge node will send the task to the cloud server for processing.

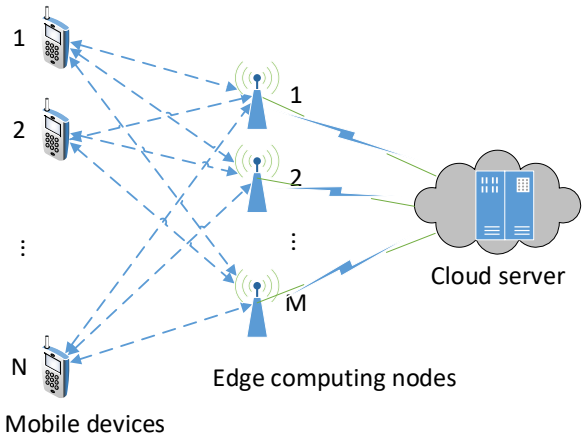

Fig. 1: Cooperative mobile edge computing network.

B. Mobile Devices

At each time slot, mobile user i has a task which needs to be executed. The task is defined by a tuple $I_i \left(D_i^i, D_i^o, C_i, T_i^r\right)$, in which D_i^i is the input data size (including input data and execution code), D_i^o is the output data size, C_i is the number of CPU cycles that is required to execute the task, and T_i^r is the delay requirement of task I_i. In this paper, we set T_i^r as the maximum delay requirement of the task. Each mobile device has a processing rate defined by f_i^l which expresses the hardware capability of the mobile device.

C. MEC Nodes

We assume that each MEC node j has a resource capability denoted by a tuple (R_j^u, R_j^d, F_j^f) in which R_j^u, R_j^d and F_j^f are total uplink rate, total downlink rate, and CPU cycle rate, respectively. These resources can be allocated partially to perform mobile users' offloading tasks.

D. Cloud Server

All MEC nodes are assumed to be able to connect with a public cloud server. If a task is sent to an MEC node, but the MEC node cannot perform due to the resource or delay constraint, the edge node will forward the task to the cloud server for processing. We denote the data rate between an MEC node and the cloud server as r^{fc}, and the the processing rate assigned to each task on the cloud server as f^c.

III. Problem Formulation

In this paper, we consider a joint offloading and resource allocation problem in which the total energy consumption of mobile devices is minimized. We denote the computation offloading decision variable for task I_i by $x_i = (x_i^l, x_i^f, x_i^c)$, in which x_i^l, x_i^f and x_i^c respectively indicates whether task I_i is processed locally at the mobile device, an MEC node, or the cloud server. Here, the variable $x_i^f = (x_{i1}^f, x_{i2}^f, ..., x_{iM}^f)$ is to determine which MEC node will execute the task I_i. Similarly, the variable $x_i^c = (x_{i1}^c, x_{i2}^c, ..., x_{iM}^c)$ is to determine which MEC node will forward the task to the cloud server.

In this paper, we solve the offloading decision and resource allocation problem jointly aiming at minimizing the total consumed energy for mobile devices in the network under the delay constraints of computation tasks. In the following, we will present analysis for energy consumption and expected delay for computing tasks under three scenarios, i.e., local, MEC node, and cloud server processing.

A. Local Processing

For the local computing approach, the offloading decision for task I_i is defined by $x_i = (1, 0, 0)$. In this case, the consumed energy E_i^l of the mobile device is proportional to the CPU cycles required for task I_i and the expected delay T_i^l is the execution time of the task. We have:

$$E_i^l = v_i C_i, \tag{1}$$

and

$$T_i^l = \frac{C_i}{f_i^l}, \tag{2}$$

where v_i denotes the consumed energy per CPU cycle [8].

B. MEC Node Processing

For the MEC node processing approach, the offloading decision for task I_i is defined by $x_i = (0, 1, 0)$. If task I_i is processed at MEC node j ($x_{ij}^f = 1$), the MEC node will allocate spectrum and computation resources for mobile device i, defined by a tuple $r_{ij} = (r_{ij}^u, r_{ij}^d, f_{ij}^f)$, in which r_{ij}^u, r_{ij}^d respectively are uplink rate, downlink rate for input and output transmissions, and f_{ij}^f is CPU cycle rate for the task being processed at MEC node j. In this case, the energy consumption at the mobile user is for both transferring data to and receiving data from the MEC node j, and the delay includes time for transmitting input data, receiving output data and task processing at the MEC node.

Let e_{ij}^u and e_{ij}^d denote the energy consumption for transmitting and receiving a unit of data, respectively. The consumed energy of mobile device E_{ij}^f and the delay T_{ij}^f are given by:

$$E_{ij}^f = E_{ij}^u + E_{ij}^d, \tag{3}$$

and

$$T_{ij}^f = \frac{D_i^i}{r_{ij}^u} + \frac{D_i^o}{r_{ij}^d} + \frac{C_i}{f_{ij}^f}, \tag{4}$$

where $E_{ij}^u = e_{ij}^u D_i^i$ and $E_{ij}^d = e_{ij}^d D_i^o$.

Additionally, task I_i can be processed at only one MEC node, and thus the consumed energy E_i^f of mobile device and the delay T_i^f since task I_i is processed at MEC node j is defined as follows:

$$E_i^f = \sum_{j=1}^{M} x_{ij}^f E_{ij}^f, \tag{5}$$

and

$$T_i^f = \sum_{j=1}^{M} x_{ij}^f T_{ij}^f, \tag{6}$$

s.t.

$$\begin{cases} x_i^f = \sum_{j=1}^{M} x_{ij}^f = 1, \\ x_{ij}^f \in \{0, 1\}, \forall j \in \mathbb{M}. \end{cases} \tag{7}$$

C. Cloud Server Processing

For the cloud computing approach, the offloading decision for task I_i is defined by $x_i = (0, 0, 1)$. If MEC node j forwards task I_i to the cloud server (i.e., $x_{ij}^c = 1$), the MEC node will allocate communication resource for mobile device i, defined by a tuple $r_{ij} = (r_{ij}^u, r_{ij}^d, f_{ij}^f)$, in which r_{ij}^u, r_{ij}^d are uplink rate, downlink rate for input and output transmissions, and $f_{ij}^f = 0$. After receiving the task, the MEC node j sends the input data to the cloud server for processing, then receives and sends the result back to the mobile device. In this case, the total consumed energy E_{ij}^c at the mobile user is the same as in the case of the MEC processing, while the delay T_{ij}^c includes the time for transmitting the input from mobile user to the MEC node, time from the MEC node to the cloud server, time for receiving the output from the cloud server to mobile user via the edge node, and task-execution time at the cloud server. These performance metrics are as follows:

$$E_{ij}^c = E_{ij}^f = E_{ij}^u + E_{ij}^d, \tag{8}$$

and

$$T_{ij}^c = \frac{D_i^i}{r_{ij}^u} + \frac{D_i^o}{r_{ij}^d} + \frac{(D_i^i + D_i^o)}{r^{fc}} + \frac{C_i}{f^c}. \tag{9}$$

Similarly, because only one MEC node can forward task I_i to the cloud server, the consumed energy E_i^c of mobile device and the delay T_i^c since task I_i is processed at the MEC node are defined as follows:

$$E_i^c = \sum_{j=1}^{M} x_{ij}^c E_{ij}^c, \tag{10}$$

and

$$T_i^c = \sum_{j=1}^{M} x_{ij}^c T_{ij}^c, \tag{11}$$

s.t.

$$\begin{cases} x_i^c = \sum_{j=1}^{M} x_{ij}^c = 1, \\ x_{ij}^c \in \{0, 1\}, \forall j \in \mathbb{M}. \end{cases} \tag{12}$$

Let E_i and T_i, respectively, be the consumed energy of mobile device and the delay when task I_i is processed. Note that a task can be executed at either the mobile device, an MEC node, or the cloud server. Thus, we have:

$$E_i = x_i^l E_i^l + x_i^f E_i^f + x_i^c E_i^c, \tag{13}$$

and

$$T_i = x_i^l T_i^l + x_i^f T_i^f + x_i^c T_i^c, \tag{14}$$

s.t.

$$\begin{cases} x_i^l + x_i^f + x_i^c = 1, \\ x_i^l, x_i^f, x_i^c \in \{0, 1\}. \end{cases} \tag{15}$$

From (7), (12) and (15), we derive the following offloading constraints:

$$\begin{cases} x_i^l + x_i^f + x_i^c = x_i^l + \sum_{j=1}^{M} x_{ij}^f + \sum_{j=1}^{M} x_{ij}^c = 1, \\ x_i^l, x_i^f, x_i^c \in \{0, 1\}, \\ x_{ij}^f, x_{ij}^c \in \{0, 1\}, \forall (i, j) \in \mathbb{N} \times \mathbb{M}. \end{cases} \tag{16}$$

The expressions (13) and (14) can be rewritten as follows:

$$\begin{aligned} E_i &= x_i^l E_i^l + \sum_{j=1}^{M} x_{ij}^f E_{ij}^f + \sum_{j=1}^{M} x_{ij}^c E_{ij}^c \\ &= x_i^l v_i C_i + \sum_{j=1}^{M} x_{ij}^f \left(e_{ij}^u D_i^i + e_{ij}^d D_i^o \right) \\ &\quad + \sum_{j=1}^{M} x_{ij}^c \left(e_{ij}^u D_i^i + e_{ij}^d D_i^o \right), \end{aligned} \tag{17}$$

and

$$\begin{aligned} T_i &= x_i^l T_i^l + \sum_{j=1}^{M} x_{ij}^f T_{ij}^f + \sum_{j=1}^{M} x_{ij}^c T_{ij}^c \\ &= x_i^l \frac{C_i}{f_i^l} + \sum_{j=1}^{M} x_{ij}^f \left(\frac{D_i^i}{r_{ij}^u} + \frac{D_i^o}{r_{ij}^d} + \frac{C_i}{f_{ij}^f} \right) \\ &\quad + \sum_{j=1}^{M} x_{ij}^c \left(\frac{D_i^i}{r_{ij}^u} + \frac{D_i^o}{r_{ij}^d} + \frac{(D_i^i + D_i^o)}{r^{fc}} + \frac{C_i}{f^c} \right), \end{aligned} \tag{18}$$

s.t.

$$\begin{cases} x_i^l + \sum_{j=1}^{M} x_{ij}^f + \sum_{j=1}^{M} x_{ij}^c = 1, \\ x_i^l, x_{ij}^f, x_{ij}^c \in \{0,1\}, \forall (i,j) \in \mathbb{N} \times \mathbb{M}. \end{cases} \tag{19}$$

In this paper, we address the joint offloading decision $\{x_i\}$ and resource allocation $\{r_i\}$ problem in which the objective is to minimize the total energy consumption of all mobile devices and all delay constraints must be satisfied, i.e.,

$$(\mathbf{P}_1) \quad \min_{\{x_i\},\{r_i\}} \sum_{i=1}^{N} E_i, \tag{20}$$

s.t.

$$\begin{cases} x_i^l + \sum_{j=1}^{M} x_{ij}^f + \sum_{j=1}^{M} x_{ij}^c = 1, \\ x_i^l, x_{ij}^f, x_{ij}^c \in \{0,1\}, \forall (i,j) \in \mathbb{N} \times \mathbb{M}, \\ T_i \le T_i^r, \forall i \in \mathbb{N}, \\ \sum_{i=1}^{N} f_{ij}^f \le F_j^f, \forall j \in \mathbb{M}, \\ \sum_{i=1}^{N} r_{ij}^u \le R_j^u, \forall j \in \mathbb{M}, \\ \sum_{i=1}^{N} r_{ij}^d \le R_j^d, \forall j \in \mathbb{M}, \\ r_{ij}^u, r_{ij}^d, r_{ij}^f \ge 0, \forall (i,j) \in \mathbb{N} \times \mathbb{M}. \end{cases} \tag{21}$$

The optimization problem $(\mathbf{P}_1)$ is an NP-hard. Hence, standard optimization techniques cannot be applied directly and the globally optimal solution is unfeasible. Thus, in the following, we introduce two effective approaches to address this problem.

IV. Proposed Optimal Solutions

A. Relaxing Optimization Solution

In this section, we introduce a relaxing approach which allows to find the optimal solution through converting binary decision variables, i.e., $\{x_i\}$, to real variables. By relaxing binary variables to real numbers, we then can reformulate the optimization problem $(\mathbf{P}_1)$ as follows:

$$(\mathbf{P}_2) \quad \min_{\{x_i\},\{r_i\}} \sum_{i=1}^{N} E_i, \tag{22}$$

s.t.

$$\begin{cases} x_i^l + \sum_{j=1}^{M} x_{ij}^f + \sum_{j=1}^{M} x_{ij}^c = 1, \\ x_i^l, x_{ij}^f, x_{ij}^c \in [0,1], \forall (i,j) \in \mathbb{N} \times \mathbb{M}, \\ T_i \le T_i^r, \forall i \in \mathbb{N}, \\ \sum_{i=1}^{N} f_{ij}^f \le F_j^f, \forall j \in \mathbb{M}, \\ \sum_{i=1}^{N} r_{ij}^u \le R_j^u, \forall j \in \mathbb{M}, \\ \sum_{i=1}^{N} r_{ij}^d \le R_j^d, \forall j \in \mathbb{M}, \\ r_{ij}^u, r_{ij}^d, r_{ij}^f \ge 0, \forall (i,j) \in \mathbb{N} \times \mathbb{M}, \end{cases} \tag{23}$$

To find the optimal solution for $(\mathbf{P}_2)$, we will prove that the relaxed problem is a convex optimization problem.

Theorem 1. *The relaxed problem $(\mathbf{P}_2)$ is a convex optimization problem.*

Proof. From (17), the energy consumption of task i, E_i, is a linear function of decision variable x_i. Consequently, the objective function $\sum_1^N E_i$ is a linear function with respect to real decision variables $\{x_i\}$.

From (18), the delay T_i is the sum of linear and linear-fractional functions: x_i^l, $\frac{x_{ij}^f}{r_{ij}^u}$, $\frac{x_{ij}^f}{r_{ij}^d}$, $\frac{x_{ij}^f}{f_{ij}^f}$, x_{ij}^c, $\frac{x_{ij}^c}{r_{ij}^u}$ and $\frac{x_{ij}^c}{r_{ij}^d}$ for all j in $\mathbb{M}$, These functions have positive coefficients: C_i, D_i^i, D_i^o, $\left(\frac{D_i^i + D_i^o}{f^{fc}} + \frac{C_i}{f^c}\right)$, D_i^i and D_i^o, respectively. Thus, $T_i(x_i, r_i)$ is a concave function with respect to x_i and r_i [9].

Since the objective function in (22) is a linear function, and the constraints in (23) are concave functions, the relaxed problem is a convex optimization problem [9]. □

To solve the relaxed optimization problem $(\mathbf{P}_2)$, we can apply some effective tools as mentioned in [9]. In this paper, we adopt the *interior-point method* [9] to find the optimal solution because this is a very effective tool to address the convex optimization problem with constraints. We assume that a central cloud server or an MEC node with powerful computing capability and energy will solve the optimization problem $(\mathbf{P}_2)$. Then, the results will be distributed to all mobile devices and edge nodes to perform.

B. Improved Branch and Bound Algorithm

Although the relaxing approach can address the join offloading and resource allocation $(\mathbf{P}_1)$, its obtained optimal decision variables are real numbers which are impractical to implement in MEC networks. Furthermore, the relaxing approach cannot utilize the advantage of binary variables in reducing the complexity and finding the optimal solution. In particular, binary variables have only two variables, i.e., either 0 or 1. In addition, when the value of a variable is zero, its product will be zero, which allows to reduce the computational complexity significantly. Thus, we introduce an improved branch and bound algorithm, namely IBBA, which allows not only addressing the MINLP, but also utilizing the characteristics of binary variables to reduce the complexity of optimization problem $(\mathbf{P}_1)$.

In this paper, we exploit the following properties of the optimization problem $(\mathbf{P}_1)$ to propose the IBBA.

- **Branching task** dictates that a task can be executed at only one place, i.e, at the mobile device, one of edge nodes, or the cloud server via an MEC node. Thus, for the offloading decisions $x_i = \{x_i^l, x_{i1}^f, ...x_{iM}^f, x_{i1}^c, ...x_{iM}^c\}$ there is only one variable that is equal to 1, and all others are equal to 0. Thus, at a node in the IBBA tree, we choose to branch the decisions of a task, forming a *(2M+1)*-tree with height N. Here, $(2M+1)$ is the number of offloading decision variables of a task, and N is the number of tasks.
- **Simplifying problem** dictates that when a task is executed at mobile device, an edge node, or the cloud server via an edge node, all other MEC nodes do not need to allocate resources toward that task. Thus, when $x_{ij}^f = 0$ or $x_{ij}^c = 0$, we can eliminate all sub-expressions of the forms $x_{ij}^f A$ and $x_{ij}^c B$, these decision variables, and related resource allocation variables f_{ij}^f, r_{ij}^u and r_{ij}^d in $(\mathbf{P}_1)$. Consequently, we have sub-problems with the reduced number of variables.
- **Preserving convexity** dictates that after fixing some binary variables, sub-problems are convex optimization problems. In particular, based on Theorem 1, it can be observed that if we fix one or multiple binary variables

in $(\mathbf{P}_1)$ and set all other variable to be real variables, the corresponding relaxed sub-problems are always convex.

Based on three aforementioned properties, we introduce Algorithm 1. This algorithm not only allows to find the optimal solution for the optimization problem $(\mathbf{P}_1)$ faster, but also provides optimal binary offloading decision variables which can be efficiently implemented in MEC networks in practice.

Algorithm 1: IBBA Algorithm

Input : Set of tasks $\{I_i \left(D_i^i, D_i^o, C_i, T_i^r\right)\}$
Set of MEC nodes $\{Node_j(R_j^u, R_j^d, F_j^f)\}$
Cloud server r^{fc}, f^c
Output: Optimal variables of problem $(\mathbf{P}_1)$

```
begin
    Solution ← ∅; optVal ← +∞
    Stack.empty(); Stack.push((P_1))
    while Stack.isNotEmpty() do
        curProb ← Stack.pop()
        tempSol, tempVal ← Solve relaxing problem of curProb
        if tempVal > optVal or curProb is infeasible then
            Prune curProb
        end
        if tempVal < optVal then
            if tempSol satisfies all integer constraints of {x_i} then
                Solution ← tempSol
                optVal ← tempVal
                Prune curProb
            end
            else
                subProblems ← Branch curProb by fixing the decisions of the first task in the set {I_i}, which is not fixed so far, based on Branching task property.
                for each subProb in subProblems do
                    Simplify subProb based on Simplifying problem property.
                    Stack.push(subPob)
                end
            end
        end
    end
    Return Solution and optVal
end
```

C. Offloading Analysis

Before conducting experiments, we analyze when mobile users can benefit from offloading. A mobile user is said to be benefit from offloading if its total energy consumption when the task is offloaded is lower than processing locally. When the task is processed at the mobile device, the consumed energy depends on the required CPU cycles for the task. However, if the task is offloaded, the consumed energy at the mobile device is for both transferring input data D_i^i to and receiving output data D_i^o from an MEC node, thus the energy depends only on the input and output data sizes. If an MEC node does not have sufficient resource, it will forward the task to the cloud server for processing. In other words, in the offloading case, the energy required does not depend on whether the task processed at an MEC node or the cloud server. Thus, for the task i, offloading will benefit if $E_i^l > E_i^f$. While E_i^l is a function of required CPU cycles, E_i^f is a function of input/output data sizes. Therefore, we introduce parameter α as ratio between the number of required CPU cycles and input data size in order to quantify the likelihood of offloading tasks. Let α_i^* be the task complexity ratio at which $E_i^l = E_i^f$. We have:

$$\alpha_i^* = \frac{e_{ij}^u D_i^i + e_{ij}^d D_i^o}{v_i D_i^i}. \tag{24}$$

Let α_i be the ratio between the number of required CPU cycles C_i and input data size D_i^i. We have $C_i = \alpha_i \times D_i^i$. Thus, task i is likely to be offloaded if $E_i^l > E_i^f$ or $\alpha_i > \alpha_i^*$. This parameter is especially important in evaluating offloaded tasks as well as analyzing the performance of whole system.

V. Performance Evaluation

A. Experiment Setup

We use the configuration of a Nokia N900 mobile device described in [10], and set the number of devices as $N = 10$. Each mobile device has CPU rate $f_i^l = 0.5$ Giga cycles/s and the unit processing energy consumption $v_i = \frac{1000}{730}$ J/Giga cycle. We denote $U(a, b)$ as discrete uniform distribution between a and b. Here, we assume that each device has a task with the input and output data sizes following uniform distributions $U(10, 20)$MB and $U(1, 2)$MB, respectively. We also assume that each task has required C_i CPU processing cycles defined by $\alpha_i \times D_i^i$ Giga cycles, in which the parameter α_i Giga cycles/MB is the complexity ratio of the task. Additionally, we consider 4 MEC nodes. Both uplink and downlink transmission rates for each MEC node are 72 Mbps as in the capacity range of IEEE 802.11n. Besides, each MEC node has the total processing rate $f_j^f = 10$ Giga cycles/s. The uplink and downlink transmission rates between MEC nodes and the cloud server are constants set at $r^{fc} = 5$ Mbps. The cloud server can allocate a fixed CPU rate $f^c = 10$ Giga cycles/s for a task. All parameters are given in Table I.

Here, we refer the policy in which all tasks are processed locally as "Without Offloading" (WOP), and the policy in which all tasks are offloaded to the MEC nodes or the cloud server as the "All Offloading" (AOP). The results obtained by Algorithm 1 (IBBA) will be compared with the relaxing optimization policy (ROP), WOP, and AOP.

B. Numerical Results

1) Scenario 1 - Vary the Complexity of Tasks: In this scenario, we investigate the effect of task complexity on the offloading decisions and energy consumption of mobile devices by varying the complexity of all tasks. At first, we choose the complexity ratio of tasks α_i as $U(200, 500)$ cycles/byte, then

TABLE I: Experimental parameters

Parameters	Value
Number of mobile devices N	10
Number of MEC nodes M	4
CPU rate of mobile devices f_i^l	0.5 Giga cycles/s
Processing energy consumption rate v_i	$\frac{1000}{730}$ J/Giga cycles
Input data size D_i^i	$U(10, 20)$ MB
Output data size D_i^o	$U(1, 2)$ MB
Required CPU cycles C_i	$\alpha_i \times D_i^i$
Unit transmission energy consumption e_{ij}^u	0.142 J/Mb
Unit receiving energy consumption e_{ij}^d	0.142 J/Mb
Delay requirement T_i^r	$[30, 60]$s
Processing rate of each MEC node F_j^f	10 Giga cycles/s
Uplink data rate of each MEC node R_i^u	72 Mbps
Downlink data rate of each MEC node R_i^d	72 Mbps
CPU rate of the cloud server f^c	10 Giga cycles/s
Data rate between FNs and the cloud r^{fc}	5 Mbps

increase each task 100 cycles/byte for each experiment. The delay requirement is set at 40s for all tasks. Other parameters are set as in Table I.

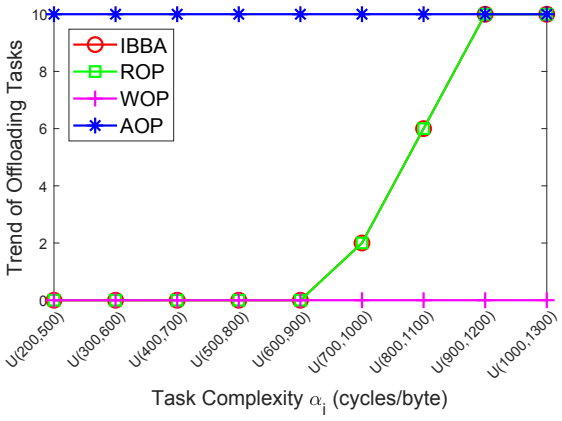

Fig. 2: Trend of offloading as the task complexity α_i increases.

Fig. 2 depicts the trend of offloading tasks when the task complexity ratio α_i is increased. While the trend of offloading tasks of the WOP and AOP are constants, i.e., 0 and 10, respectively, the offloading trends of both IBBA and ROP go up as α_i increases. Specifically, the numbers of offloaded tasks of the IBBA and ROP are equal 0 as the complexity ratio increases from $U(200, 500)$ to $U(600, 900)$. This is because α_i is less than α_i^*, which is equal to 911 cycles/byte according to Eq. (24) and parameters in Table I. Moreover, all tasks executed locally still can satisfy the delay constraints ($T_i^r = 40$s). Then, the numbers of offloading tasks increase dramatically from 0 to 10 since the task complexity ratio α_i increases from $U(600, 900)$ to $U(1000, 1300)$. This is because there is an increasing number of tasks with $\alpha_i > \alpha_i^*$. Noticeably, Fig. 2 also shows that all tasks get benefit from offloading when α_i is in the ranges from $U(900, 1200)$ to $U(1000, 1300)$.

Fig. 3 shows the average energy consumption of mobile devices for IBBA, ROP, WOP and AOP, when α_i increases from $U(200, 500)$ to $U(1000, 1300)$. Generally, while the average energy consumption is a constant (18.4J/task) for the AOP, it increases for other methods. This is because in the AOP, all tasks are offloaded and the consumed energy at mobile devices depends only on the data sizes of D_i^i and D_i^o. For the WOP, the consumed energy increases linearly according to the task complexity ratio. Similar to Fig. 2, the energy consumption trends of the IBBA and ROP are the same because their offloading decisions are impacted by the energy efficiency factor without constraints.

Fig. 3 shows that the optimal values of both IBBA and ROP are always lower than or equal to the minimum value of the WOP and AOP. Specifically, as α_i increases from $U(200, 500)$ to $U(600, 900)$, the consumed energy of the IBBA and ROP are equal to the case of the WOP because all tasks are processed locally due to no benefit from offloading. Then, the consumed energy of both IBBA and ROP will be reduced a bit as α_i increases from $U(700, 1000)$ to $U(800, 1100)$ because some tasks can be offloaded now. When the complexity ratio increases from $U(900, 1200)$ to $U(1000, 1300)$, all tasks get benefit from offloading, and thus for the IBBA and ROP, all the tasks are processed at either MEC nodes or the cloud server, leading to the equality in consumed energy of the three methods except WOP.

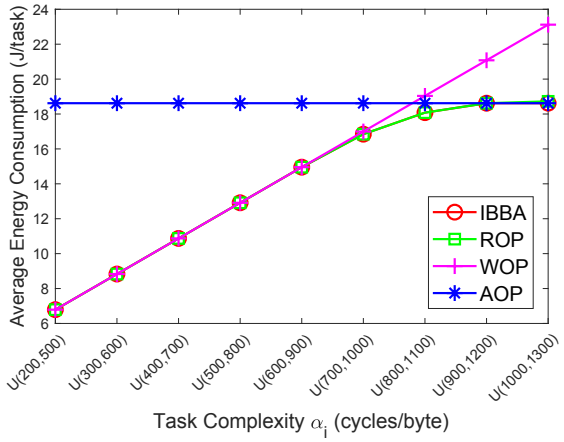

Fig. 3: Average energy consumption of mobile devices as the task complexity α_i is increased.

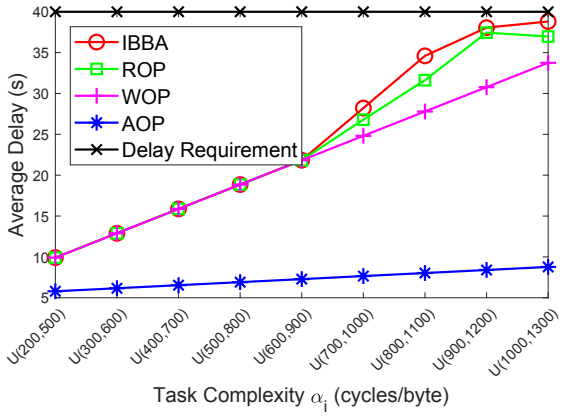

Fig. 4: Average task processing delay as the task complexity α_i is increased.

Fig. 4 shows the average delay as the task complexity is increased. Generally, the average delay increases for all

policies. Remarkably, the average delays of the IBBA and ROP are always lower than the delay requirement $T_i^r = 40$s. From the average delay of the AOP policy, we can observe that the offloading computation can support all tasks with less than 10s of the delay requirement T_i^r.

2) Scenario 2 - Vary the Task Delay Requirements: In this scenario, we study the impact of task delay requirements on the energy consumption and offloading decisions of mobile devices. We keep the settings as in Table I, and select a set of the tasks with complexity α_i following $U(800, 1100)$ from Scenario 1. Specifically, there are 6 tasks receiving benefits from offloading due to $\alpha_i > \alpha_i^* = 911$ cycles/byte. We then change input/output data sizes of one task so that even it does not get benefit from offloading, but its local processing delay T_i^l is greater than 60s. The delay requirement T_i^r for all tasks increases from 30s to 60s.

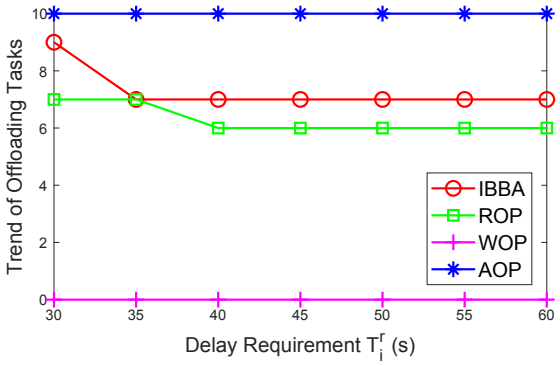

Fig. 5: Trend of offloading as the delay requirement is looser.

Figs. 5 and 6 illustrate the trend of offloading tasks and average energy consumption, respectively. As observed in Fig. 5, at $T_i^r = 30$s, while IBBA has 9 offloaded tasks, ROP has only 7 offloaded tasks. As mentioned before, there are only 6 beneficial tasks from offloading, and IBBA algorithm always returns the optimal integer solution. Thus, 9 offloaded tasks in IBBA including 6 tasks which get benefits from offloading and 3 tasks with the local processing delay T_i^l greater than T_i^r. Besides, ROP is derived directly from the solution by solving the relax problem $(\mathbf{P}_2)$. Thus, there is an inaccurate proportion in the results. With task i which does not get benefit from offloading and the local delay $T_i^l > T_i^r$, the IBBA will decide to offload it, but ROP will decide to process locally if x_i^l is greater than all x_{ij}^f and x_{ij}^c variables. Similarly, while the IBBA maintains 7 offloaded tasks including 6 beneficial tasks and a task with local delay $T_i^l > 60$s as T_i^r increases from 40s to 60s, ROP offloads only 6 beneficial tasks. Consequently, in Fig. 6, the consumed energy of ROP is always lower than IBBA, the actual MINLP solution. The ROP has to pay for this by having a proportion of tasks that will not satisfy the constraints. In summary, in both IBBA and ROP, when the delay requirements are looser, tasks without benefit from offloading, tend to be processed locally aiming at reducing the consumed energy.

While Scenario 1 exposes the benefit of offloading computation in term of consumed energy at mobile devices, in Scenario 2, the offloading policy is applied to reduce the delay of processing tasks satisfying delay constraints.

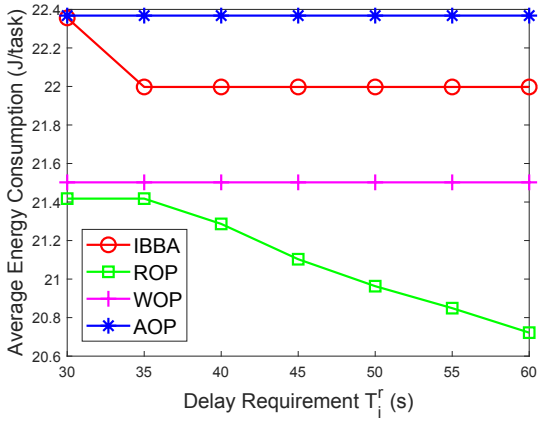

Fig. 6: Average consumed energy at mobile devices when the delay requirement is looser.

VI. Summary

In this paper, we study the offloading problem for the cooperative mobile edge computing network in which mobile edge nodes cooperate to perform computation requirements of the mobile users. To minimize the total energy consumption and meet all delay requirements of mobile users, we formulate the joint offloading decision and resource allocation optimization problem, and propose two effective methods, i.e., IBBA based on the Branch and bound method and ROP based on the interior point method, to find the optimal solution for both the mobile users and edge nodes. The numerical results then verify the efficiency of the proposed solutions.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628-1656, Mar. 2017.

[2] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757-6779, Mar. 2017.

[3] ETSI White Paper No. 11, Mobile Edge Computing: A Key Technology Towards 5G. Available Online: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[4] Z. Chang, Z. Zhou, T. Ristaniemi, and Z. Niu, "Energy efficient optimization for computation offloading in fog computing system," in *IEEE Global Communications Conference*, pp. 16, Singapore, Dec. 2017.

[5] M. H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *IEEE Conference on Computer Communications*, pp. 19, Atlanta, USA, May 2017.

[6] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 27952808, Oct. 2016.

[7] V. Cardellini, V. De N. Person, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, vol. 157, no. 2, pp. 421449, Jun. 2016.

[8] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *IEEE Conference on Computer Communications*, pp. 27162720, 2012.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[10] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," *HotCloud*, vol. 10, pp. 14, 2010.