Optimal Task Offloading and Resource Allocation for Fog Computing

Thai T. Vu, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz

School of Electrical and Data Engineering, University of Technology Sydney, Australia

Abstract—We propose a novel multi-tier fog and cloud computing architecture that enables edge nodes to cooperate in sharing computing and radio resources so as to minimize the total energy consumption of mobile users subject to their delay requirements. We first formulate the joint task offloading and resource allocation optimization problem as a mixed integer non-linear programming (MINLP). Due to the combination of binary (offloading decisions) and real variables (resource allocations), the problem is an NP-hard and computationally intractable. To circumvent, we relax the binary decision variables to transform the MINLP to a relaxed optimization problem with real variables. After proving that the relaxed problem is a convex one, we propose three solutions namely ROP, IBBA, and FFBD. In ROP, the solution of the relaxing problem is converted to the integer one and used as the final solution. In IBBA, the branch-and-bound method is combined with different offloading priority policies yielding two variants IBBA-LFC and IBBA-LCF. Note that the branch and bound method like IBBA may require exponential complexity in the worst case. As such, we develop a distributed approach, namely feasibility-finding Benders decomposition (FFBD), that decomposes the original problem into a master problem for the offloading decision and subproblems for resource allocation. These (simpler) subproblems can be solved in parallel at fog nodes, thus help reduce both the complexity and the computational time. FFBD has two different versions (FFBD-S and FFBD-F) is a distributed algorithm based on Benders decomposition. The numerical results show that IBBA-LFC/LCF and FFBD-S/F gain the optimal solution minimizing the total energy consumption and meet all delay requirements for mobile users. The FFBD-F with the fast solution detection method also has the least computation time of four algorithms IBBA-LFC/LCF and FFBD-S/F.

Keywords- Task offloading, fog computing, resource allocation, latency, MINLP, branch-and-bound algorithm, and Benders decomposition.

I. INTRODUCTION

Mobile applications that have been recently developed are often demanding in computation, e.g., 3D rendering and image processing, as well as low latency, e.g., interactive games and online object recognition (e.g., in mixed/augmented reality) [1], [2]. Nonetheless, mobile and Internet-of-Things (IoT) devices are usually limited by computing resources, battery life, and network connections. As a promising solution, a new network architecture, referred to as mobile edge or fog computing, has recently received paramount interest.

The key idea of fog computing is to "move" computing resources closer to mobile users [3]. In a fog architecture, 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. That helps save energy, increase operation time, and enable new applications/services

for mobile devices by utilizing powerful resources at the edge. Fog computing can also reduce operating costs for mobile network operators up to 67% by reducing the total throughput and peak backhaul bandwidth consumption [4]. Thus, fog computing with its characteristics (i.e., low latency, proximity, high bandwidth, mobility support and location awareness) [5] can solve two key challenges (ie., reducing latency, and energy consumption) to future wireless networks [6].

However, unlike public clouds, e.g., Amazon Web Services and Microsoft Azure, a fog node does not possess abundant computing resource. While the computation offloading demand from mobile users is huge, every fog node can support only a small amount of tasks. Moreover, not all computational tasks benefit from being offloaded to the fog node. Some tasks even consume more energy when being offloaded than being processed locally due to the communication overhead, i.e., transmitting requests and receiving results [3]. Given the above, this work considers the joint task offloading and resource allocation optimization problem, aiming to minimize the energy consumption for mobile devices under the fog nodes' resource constraints and tasks' delay requirements.

A. Related Work

The above problem has been visited from different angles and using different approaches over the last few years [7]–[25]. The earliest researches simply solve the offloading problem between a single device and a cloud server [7]–[12]. In [7], the whole soft real-time requirement mobile application is executed in either the user device or the cloud server to minimize energy consumption under a stochastic wireless channel. In [8]–[12], to improve performance, an application is split into tasks for collaborative task execution. Due to task dependence, graph or tree-based algorithms for path searching were used mostly. In [8], to minimize the energy consumption while meeting a deadline, sequential tasks with a hard time deadline were executed as a constrained stochastic shortest path problem over an acyclic graph. In [9], an approximation algorithm based on dynamic programming was proposed to solve the latency minimization problem. [10] formulated a problem to maximize the number of tasks that should be executed on wearable devices with guaranteed delay requirements. A fast genetic-based algorithm was proposed to approximate the optimal solution. In [11], applying the DVFS techniques, a linear-time rescheduling algorithm started from a minimal-delay scheduling solution and then performed energy reduction by migrating tasks among the local cores and the cloud server. [12] solved the same problem as in [11], additionally considered the power allocation in the task

offloading decision. A decomposition technique was utilized to design a distributed resource allocation algorithm.

Some other researches [13]–[16] focused on either offloading decisions or resources allocation among multiple user devices. In these works, there was independence between users' tasks. In [13], heuristic offline and online algorithms were proposed to achieve minimum average completion time for all the users based on the number of provisioned resources on the cloud. In [14], the interaction between selfish mobile users, who want to maximize offloading decisions in a three-tier fog architecture, were modeled in a non-cooperative game. In [15], the offloading in a two-tier architecture was formulated as a non-cooperative game minimizing a cost function of delay and consumed energy. [16] minimized the expected time-average energy consumption for task executions adopting network-assisted D2D communications to share resources between mobile users.

Recently, researchers have been investigated in joint offloading and resources allocation among devices with delay-tolerant tasks [17]–[20]. [17] developed a game-based distributed algorithm for allocating the computational tasks among nearby devices and the edge cloud minimizing the mean completion time of each device's tasks. [18] exploited a decomposition technique to maximize the weighted sum of all users' offloading utilities, which comprise the reducing levels of energy and delay from offloading tasks to one of fog servers. [19] developed a scheme to offload tasks on the edge cloud or process locally. Although the aim is to minimize the average task duration with the limited battery capacity, the individual delay is not satisfied. [20] aimed at minimizing average transmission energy consumption while guaranteeing the average queueing latency at IoT devices.

A three-tier system model including mobile devices, fog nodes and the cloud has been introduced in recent researches [23]–[25]. The works [23] and [24] adopted a three-tier system model including multiple users, a cloudlet and a cloud server, then jointly optimized offloading decisions as well as computation and communication resources allocation among users. The overall cost of energy, computation, and delay for all users was minimized. Although the cost could be optimized, the delay constraints were ignored. As in [24], [25] adopted three-tier system model including user devices, a fog node, and a cloud server, and then minimized the maximal weighted cost of delay and energy consumption among all user devices while meeting the delay thresholds. By doing so, a task with lower local energy consumption and local execution time satisfying its delay constraint can still be offloaded in order to keep the lower maximal cost of the whole system if its delay from offloading is much lower than local processing. In this situation, the user device does not gain any energy benefit from offloading. Thus, optimizing the cost function does not guarantee to minimize the overall users' energy consumption and the satisfaction of every individual.

We can see that all the mentioned researches worked on either a single mobile device or multiple user devices with delay-tolerant tasks. With the diversity of IoT applications, each computational task has a different delay requirement. For example, an interactive game may require task completion in some milliseconds while a smart meter application accepts a delay of hours. Therefore, the hard-time delay is a crucial condition in offloading and resources allocation problems. In [21], MIMO multicell communication and cloud-based computation resources were jointly allocated among multiple users with delay constraints to minimize the total their energy consumption. However, without fog nodes, a cloud server was an unique external computation resource for the whole system. Another drawback of this work was the assumption of fixed computation offloading decisions among users. Unlike [21], [22] minimized the total energy consumption in the cloud for executing the hard-delay constraint tasks and in C-RAN for transmitting the processing results back to the mobile users. Instead of task offloading, it assumes that every mobile device has a clone in the cloud.

As aforementioned, the fog nodes are limited in computing power, in comparison to cloud servers, but they can satisfy low latency applications. Unlike previous works, in this paper, we study a large scale multi-layer cooperative fog network including mobile devices, multiple fog nodes, and a cloud server. Under our architecture, the tasks can be offloaded to either one of fog nodes or the cloud server. Then, we formulate and solve a joint task offloading and resource allocation optimization problem for all mobile users and fog nodes, aiming to minimize the total energy consumption of mobile devices while meeting all tasks' delay requirements.

B. Main Contributions

Major contributions of this paper are as follows:

- We first formulate the joint task offloading and resource allocation optimization problem for all mobile users in a large scale network. Since the optimization problem is a mixed integer 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 one which can be effectively tackled with conventional optimization methods, e.g., the interior point method (IPM).

- Although IPM can find the optimal solution for the relaxed problem, the obtained decision variables are still real numbers. Therefore, a simple approximation method, namely "Relaxing Optimization Policy" (ROP), is proposed to roughly convert the optimal solution of the relaxed problem to binary decisions.

- Besides, we develop an improved branch-and-bound (BB) algorithm (IBBA) [26] to effectively find the optimal solution. The IBBA method not only finds the optimal offloading decisions but also utilizes the characteristics of the problem to reduce the computational complexity.

- A problem can have some optimal solutions with different numbers of tasks processed at the local device, edge nodes, and the cloud server. However, a general solver does not support solution selection in any priority order. Different from our previous work [26], we develop priority solution selection strategies combining with the branch-and-bound method yielding two variants, IBBA-

LFC/LCF. Consequently, the most wanted solution will be returned as the final results.

- However, the intermediate problems of IBBA still have a large size due to mixed offloading and resource allocation variables. Consequently, a distributed feasibility-finding Benders decomposition (FFBD) based on Benders decomposition is also developed in this paper. Exploiting special characteristics of our problem, the subproblems (SP) in FFBD can be solved independently at edge nodes. Unlike other variations of Benders decomposition [27] where the cuts are generated by solving the dual of SPs, the cutting-planes of FFBD are created from solving the subproblems based on the set theory.
- To reduce the number of iterations in the master problem (MP) of FFBD, MP is initially equipped a set of cutting-planes based on a theorem about the infeasibility of SPs. To reduce the time finding resources allocation solutions in SPs, a fast solution detection method is also proposed. Consequently, FFBD has two variants, FFBD-S with a standard solver and FFBD-F with the fast solution detection method.
- The complexities of the IBBA-LFC/LCF and FFBD-S/F algorithms in term of intermediate problems' size are also investigated intensively.
- The experimental results show that all the proposed algorithms, IBBA-LFC/LCF and FFBD-S/F, return the optimal solutions minimizing total consumed energy at mobile devices while they still satisfy the delay constraints of tasks. IBBA-LFC/LCF can return a solution with the most wanted offloading decisions, whereas FFBD-F has the least computation time of the four algorithms, IBBA-LFC/LCF and FFBD-S/F.

The rest of the paper is as follows. We describe the system model and the problem formulation in Section II. Section III presents the proposed algorithms, ROP, IBBA-LFC/LCF, and FFBD-S/F, the theoretical analyses, and the priority solution selection strategies as well. In Section IV, we evaluate the performance of the IBBA-LFC/LCF and FFBD-S/F algorithms and compare them with the relaxing optimization policy (ROP), and two methods: "Without Offloading" (WOP) and the policy in which all tasks are offloaded to the fog nodes or the cloud server as the "All Offloading" (AOP). Conclusion is drawn in Section V.

II. System Model and Problem Formulation

A. Network Model

Fig. 1 illustrates a three-tier fog computing system with N mobile devices $\mathbb{N} = \{1, \ldots, N\}$, M cooperative fog nodes $\mathbb{M} = \{1, \ldots, M\}$, and one cloud server. Although mobile users may have some independent tasks, we assume that, at each time slot, mobile user i can request to offload a computing task $I_i \left(D_i^i, D_i^o, C_i, t_i^r \right)$, in which D_i^i and D_i^o respectively are the input (including input data and execution code) and output/result data lengths, C_i is the number of CPU cycles that are required to execute the task, and t_i^r is the maximum delay requirement of the task. Only mobile device, fog nodes, or the cloud server satisfying the delay requirement
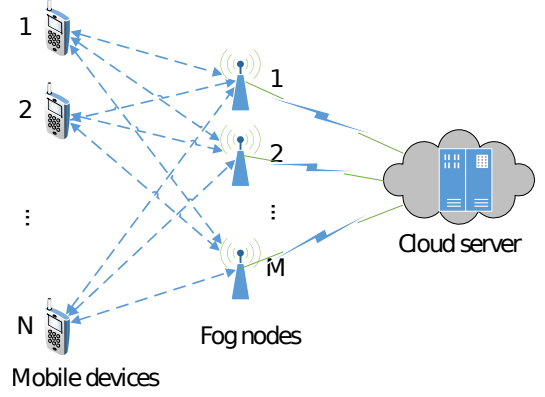

Fig. 1: Three-tier cooperative fog computing network.

are eligible to process the task. The proposals in this paper still work with mobile users having multiple tasks.

1) Local Processing: Mobile device i has a processing rate f_i^l in cycles per second. If task I_i is processed locally, the necessary time T_i^l to perform the task is given by

$$T_i^l = C_i / f_i^l. \tag{1}$$

The consumed energy E_i^l of the mobile device is proportional to the CPU cycles required for task I_i and is given by

$$E_i^l = v_i C_i, \tag{2}$$

where v_i denotes the consumed energy per CPU cycle [19], [28].

2) Fog Node Processing: Fog node j has capabilities denoted by a tuple (R_j^u, R_j^d, R_j^f) in which R_j^u, R_j^d, and R_j^f are the total uplink rate, total downlink rate, and CPU cycle rate respectively. If task I_i is processed at fog node j, then this node will allocate spectrum and computation resources for mobile device i, defined by a tuple $\mathbf{r}_{ij} = (r_{ij}^u, r_{ij}^d, r_{ij}^f)$, in which $r_{ij}^u, r_{ij}^d, r_{ij}^f$ respectively are uplink, downlink, and CPU cycle rates for input, output transmissions, and task execution. In this case, the energy consumption at the mobile user is for both transferring input to and receiving output from fog node j, and the delay includes time for transmitting input, receiving output and task processing at the fog node.

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

$$T_{ij}^f = D_i^i / r_{ij}^u + D_i^o / r_{ij}^d + C_i / r_{ij}^f, \tag{3}$$

and

$$E_{ij}^f = E_{ij}^u + E_{ij}^d, \tag{4}$$

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

3) Cloud Server Processing: Assume that all fog nodes are connected to a public cloud server. We denote the data rate between a fog node and the cloud server as r^{fc}, and the processing rate assigned to each task on the cloud server as f^c. Due to the optimization goal of the whole system, a fog node can forward a task to the cloud server for processing.

If fog node j forwards task I_i to the cloud server, it will allocate resources for mobile device i, defined by a tuple

$\mathbf{r}_{ij} = (r_{ij}^u, r_{ij}^d, r_{ij}^f)$, in which r_{ij}^u, r_{ij}^d are uplink rate, downlink rate for input and output transmissions, and $r_{ij}^f = 0$. After receiving the task, fog node j sends the input data to the cloud server for processing, then receives and sends the result back to the mobile user. In this case, the consumed energy E_{ij}^c at the mobile user is only for transmitting input and output data directly to and from fog node j as in the case of fog node processing, while the delay T_{ij}^c includes the time for transmitting the input from mobile user to the fog node, time from the fog 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:

$$T_{ij}^c = D_i^i/r_{ij}^u + D_i^o/r_{ij}^d + (D_i^i + D_i^o)/r^{fc} + C_i/f^c, \quad (5)$$

and

$$E_{ij}^c = E_{ij}^f = E_{ij}^u + E_{ij}^d. \quad (6)$$

B. Problem Formulation

We denote the binary offloading decision variable for task I_i by $\mathbf{x}_i = (x_i^l, x_{i1}^f, \ldots, x_{iM}^f, x_{i1}^c, \ldots, x_{iM}^c)$, in which x_i^l, x_{ij}^f, and x_{ij}^c respectively indicate that task I_i is processed locally at the mobile device, fog node j, or the cloud server (via fog node j). From Eq. (1)–(6), the consumed energy E_i of the mobile user and the delay T_i when task I_i is processed are given as

$$E_i = \mathbf{e}_i^\top \mathbf{x}_i, \text{ and } T_i = \mathbf{h}_i^\top \mathbf{x}_i, \quad (7)$$

where $\mathbf{e}_i = (E_i^l, E_{i1}^f, \ldots, E_{iM}^f, E_{i1}^c, \ldots, E_{iM}^c)$ and $\mathbf{h}_i = (T_i^l, T_{i1}^f, \ldots, T_{iM}^f, T_{i1}^c, \ldots, T_{iM}^c)$.

Let $\mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_N)$ and $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$. Then, the total consumed energy of mobile devices is given as

$$E = \mathbf{e}^\top \mathbf{x}. \quad (8)$$

In this paper, we address a joint offloading decision ($\mathbf{x}$) and resource allocation ($\mathbf{r} = \{\mathbf{r}_{ij}\}$) problem that aims to minimize the total energy consumption of all mobile devices under the delay requirement. The problem is formally stated as follows.

$$(\mathbf{P}_0) \quad \min_{\mathbf{x}, \mathbf{r}} \mathbf{e}^\top \mathbf{x}, \quad (9)$$

s.t.

$$(\mathbf{R}_0) \begin{cases} (\mathcal{C}_1) \ T_i \le t_i^r, \forall i \in \mathbb{N}, \\ (\mathcal{C}_2) \ \sum_{i=1}^N r_{ij}^f \le R_j^f, \forall j \in \mathbb{M}, \\ (\mathcal{C}_3) \ \sum_{i=1}^N r_{ij}^u \le R_j^u, \forall j \in \mathbb{M}, \\ (\mathcal{C}_4) \ \sum_{i=1}^N r_{ij}^d \le R_j^d, \forall j \in \mathbb{M}, \\ r_{ij}^u, r_{ij}^d, r_{ij}^f \ge 0, \forall (i,j) \in \mathbb{N} \times \mathbb{M}, \end{cases} \quad (10)$$

and

$$(\mathbf{X}_0) \begin{cases} (\mathcal{C}_5) \ x_i^l + \sum_{j=1}^M x_{ij}^f + \sum_{j=1}^M x_{ij}^c = 1, \forall i \in \mathbb{N}, \\ x_i^l, x_{ij}^f, x_{ij}^c \in \{0,1\}, \forall (i,j) \in \mathbb{N} \times \mathbb{M}. \end{cases} \quad (11)$$

where $(\mathcal{C}_1)$ is the delay requirement of tasks, $(\mathcal{C}_2), (\mathcal{C}_3)$ and $(\mathcal{C}_4)$ are resource constraints at fog nodes, and $(\mathcal{C}_5)$ is offloading decision constraints.

III. PROPOSED OPTIMAL SOLUTIONS

The optimization problem $(\mathbf{P}_0)$ is an NP-hard due to its mixed integer non-linear programming. Hence, it may take standard optimization solvers exponential time. We observe that by relaxing its binary variables to real numbers $x_i^l, x_{ij}^f, x_{ij}^c \in [0,1], \forall (i,j) \in \mathbb{N} \times \mathbb{M}$, the resulting problem is a convex optimization problem [29]. In the sequel, using on this characteristic, we introduce three effective approaches to address the problem $(\mathbf{P}_0)$.

A. Convexity of Relaxing Problem

We first convert binary decision variables, i.e., $\mathbf{x}$, to real variables, then prove that the relaxing problem is a convex optimization problem. By relaxing binary variables to real numbers, we can reformulate the optimization problem $(\mathbf{P}_0)$ as follows:

$$(\widetilde{\mathbf{P}}_0) \quad \min_{\mathbf{x}, \mathbf{r}} \mathbf{e}^\top \mathbf{x}, \quad (12)$$

s.t. $(\mathbf{R}_0)$ and

$$(\widetilde{\mathbf{X}}_0) \begin{cases} x_i^l + \sum_{j=1}^M x_{ij}^f + \sum_{j=1}^M x_{ij}^c = 1, \forall i \in \mathbb{N}, \\ x_i^l, x_{ij}^f, x_{ij}^c \in [0,1], \forall (i,j) \in \mathbb{N} \times \mathbb{M}. \end{cases} \quad (13)$$

To find the optimal solution for $(\mathbf{P}_0)$, we first will prove that the relaxed problem $(\widetilde{\mathbf{P}}_0)$ is a convex optimization problem. The convexity of the relaxed problem is maintained in subproblems, which are gained by fixing some variables. Based on this characteristic, then, we introduce three effective approaches to address the problem $(\mathbf{P}_0)$.

THEOREM 1. *The relaxed problem $(\widetilde{\mathbf{P}}_0)$ is a convex optimization problem.*

Proof: The detailed proof is presented in Appendix A.

B. Relaxing Optimization Solution

One approach to solve problem $(\mathbf{P}_0)$ is called "Relaxing Optimization Policy" (ROP). In ROP, we first solve the relaxed optimization problem $(\widetilde{\mathbf{P}}_0)$. The convex optimization problem with constraints can be solved efficiently by the *interior-point method* [29], which is implemented in many popular solvers such as CPLEX, MOSEK, and the fmincon() function in MATLAB. Then, the real offloading decision solution of $(\widetilde{\mathbf{P}}_0)$ is converted to the closest integer decision for the problem $(\mathbf{P}_0)$. Finally, to improve the performance of the ROP method, we can solve again the resource allocation of problem $(\mathbf{P}_0)$ with the fixed offloading decision. The resource allocation solution after this step with the fixed offloading decision is the final solution of the ROP method.

Although the approximation method, ROP, can quickly return a solution (after solving the relaxed problem one or two times), it is often inaccurate and not optimal. There is nothing guarantee that the final solution satisfying all constraints and minimizing the total energy consumption of mobile devices. Thus, in the following sections, we will introduce two effective methods to find the optimal solutions satisfying every condition of the problem $(\mathbf{P}_0)$.

C. Improved Branch and Bound Algorithm

The Branch and bound method (BB) is the framework for almost all commercial software for solving mixed integer programming (MIP) models. The conventional BB works as a search tree, in which every node (*not the concept of fog nodes in the network model*) on the tree represents a subproblem after fixing a binary variable. The relaxed subproblem at that node can be solved to detect the potential of that node before branching and visiting the left or right children nodes. As so, in the standard BB, the size of subproblems at children nodes is 1 less than that of the father node in term of the number of variables. In other words, the complexity of the intermediate problems is reduced slowly in the conventional BB.

In this section, we introduce an improved branch and bound algorithm, namely IBBA, which efficiently solves the MINLP ($\mathbf{P}_0$) by utilizing the characteristics of binary decision variables to reduce the complexity. IBBA, summarized in Algorithm 1, has the following features:

- **Branching task** dictates that a task can be executed at only one place, i.e, at the mobile device, one of the fog nodes, or the cloud server via a fog node. Thus, for the offloading decisions $\mathbf{x}_i$ of task i there is only one variable that is equal to 1, and all others are equal to 0. Thus, at a node in the search tree of IBBA, we choose to branch the decisions of a task, forming a $(2M+1)$-tree with height N.

- **Simplifying problem** dictates that when a task is executed at the mobile device, a fog node, or the cloud server via a fog node, all other fog 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 r_{ij}^u, r_{ij}^d, and r_{ij}^f in ($\mathbf{P}_0$). 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 ($\mathbf{P}_0$) and set all other variables to be real variables, the corresponding relaxed subproblems are always convex.

Priority Solution Selection

An MIP problem can have some different optimal solutions in which the numbers of tasks are processed at the local device, edge nodes or the cloud server may vary. However, a general solver for MIP does not support solution selection in any priority order. In IBBA, this drawback can be overcome by implementation techniques.

Recall that IBBA is a search tree based algorithm, and the first optimal solution during traveling is returned as the final solution. Therefore, we develop a tree so that the most expected solution will be found first. We consider the local device, edge nodes and the cloud server as service providers, and tasks as customers. Then, the priority is defined by which customer will be served first, and which service provider will be chosen first when there is a demand from customers.
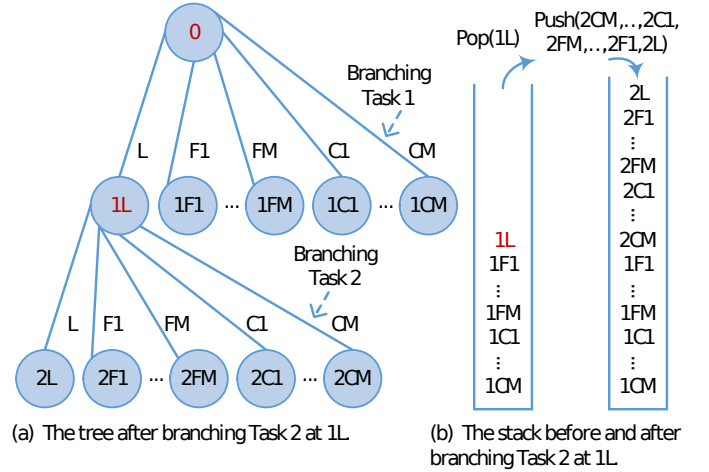

(a) The tree after branching Task 2 at 1L.

(b) The stack before and after branching Task 2 at 1L.

0: The original problem
xL: Subproblem where Task x processed locally
xFy: Subproblem where Task x processed at MEC node y
xCy: Subproblem where Task x processed at Cloud via MEC node y

Fig. 2: Priority Search Tree for IBBA.

Without loss of generality, we assume tasks and services providers are sorted in priority descending order: $T_1, \ldots, T_N$ and $L, F_1, \ldots, F_M, C_1, \ldots, C_M$. Here, T_i is task i, L is a local device, F_j is fog node j, and C_j is the cloud server via fog node j. The search tree in IBBA, which is depicted in Fig. 2, is developed as follows.

- The deep-first search (DFS) algorithm and stack data structure are used.
- Tasks are evaluated in priority descending order: $T_1, \ldots, T_N$. Tasks are called *active* tasks if they have not yet been fixed the offloading decisions on the search tree.
- When branching a task – deciding where the task being processed at, the consequent subproblems are generated and pushed in the stack in priority ascending order: $C_M, \ldots, C_1, F_M, \ldots, F_1, L$.

In Fig. 2a, the DFS search algorithm travels the tree from left to right, thus, the most left branch solution, which has the highest priority, will be found first. Fig. 2b shows the status of the stack before and after processing subproblem 1L. We can see that the most priority subproblem always on the top of the stack. Remarkably, when branching a task, we can change the order of subproblems pushing in the stack according to its specific demand.

Pseudo code in Algorithm 1 describes the details of IBBA. In IBBA algorithm, some major data structures are used including an object s for storing the updated solution, a stack t with $push()$ and $pop()$ operations for adding in and taking out a subproblem, $empty()$ and $isNotEmpty()$ operations for making empty and checking if the stack is empty or not. First, it initializes an empty solution set s, assigns a positive infinity to the optimal result $minE$, which is the total consumed energy at all mobile devices as defined in Eq. (8), and the stack is made empty before pushing the original problem ($\mathbf{P}_0$) into. Second, the algorithm repeatedly pops a subproblem from the top of the stack and solves the

relaxing version of the subproblem. In the case of the relaxing problem with no better result than the previous one or being infeasible, we prune the current subproblem. Otherwise, in the case of the relaxing problem with a better result, this result will be recorded as the optimal value if it satisfies all integer constraints or the next *active* task with the highest priority will be fixed offloading decisions yielding a set of smaller subproblems. These subproblems will be pushed into the stack in the order of increasing interest in resource providers so that the task with the most interested provider will be at the top of the stack. The algorithm is stopped only when the stack is empty.

Fig. 3: Feasibility-Finding Benders Decomposition.

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 fog nodes $\{Node_j(R_j^u, R_j^d, F_j^f)\}$
Cloud server r^{fc}, f^c

Output: Optimal solution and value of problem $(\mathbf{P}_0)$

1 **begin**
2    $s \leftarrow \emptyset$ ▷ Initialize empty solution
3    $minE \leftarrow +\infty$ ▷ Initialize consumed energy $+\infty$
4    $t.empty()$ ▷ Make stack empty
5    $t.push((\mathbf{P}_0))$ ▷ Put $(\mathbf{P}_0)$ into stack
6   **while** $t.isNotEmpty()$ **do**
7      $p \leftarrow t.pop()$ ▷ Get subproblem from top of stack
8      $subs, subminE \leftarrow$ <u>Solve</u> relaxing problem of p then return its optimal solution and value
9   **if** $subminE > minE$ **or** p is infeasible **then**
10   <u>Prune</u> p ▷ Delete subproblem p
11   **end**
12   **if** $subminE < minE$ **then**
13   **if** $subs$ satisfies all integer constraints of $\{\mathbf{x}_i\}$ **then**
14          $s \leftarrow subs$ ▷ Update solution
15          $minE \leftarrow subminE$ ▷ Update optimal result
16   <u>Prune</u> p ▷ Delete subproblem p
17   **end**
18   **else**
19          $children \leftarrow$ <u>Branch</u> p by fixing the decisions of an *active* task with the highest priority in the set $\{I_i\}$ based on **Branching task** property.
20   Sort $children$ in the order of increasing interest in resource providers.
21   **for each** $child$ **in** $children$ **do**
22   Simplify $child$ based on **Simplifying problem** property.
23            $t.push(child)$ ▷ Put subproblem into stack
24   **end**
25   **end**
26   **end**
27   **end**
28   **Return** s and $minE$
29 **end**
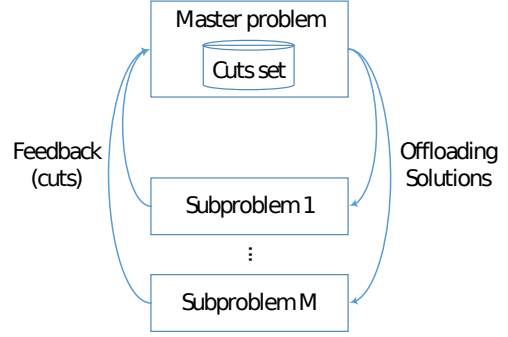
D. Feasibility-Finding Benders Decomposition

Although IBBA has some improvements in compared with the conventional BB algorithm as well as can be implemented in parallel or distributed way [30], it still has some limitations. First, the size of the intermediate subproblems reduces slowly. Second, IBBA is not a distributed algorithm in essence. Therefore, a distributed algorithm in which the original problem is broken down into low complexity subproblems is a real requirement.

In $(\mathbf{P}_0)$, there are two sets of variables $\mathbf{x}$ and $\mathbf{r}$ in which the former is complex due to its binary properties and the latter is simple real values. Therefore, the Bender decomposition (BD) method can be applied to this problem. However, due to the multiple non-linear constraint expressions between integer and real variables, the Benders decomposition method in [31] is inapplicable. Additionally, this linearization method with the dual multipliers faces the zig-zagging issue [32], [33] which increases the computation time. This method also has difficulty in achieving the optimal solution even though the lower bound is close to the upper bound. Thus, we introduce a distributed algorithm named Feasibility-Finding Benders decomposition (FFBD) as illustrated in Fig. 3. The key point of FFBD is the generation of *Benders cuts*, which exclude superfluous solutions, based on the set theory. This completely differs from the work [31], where the Benders cuts are created by solving the dual problem.

Specifically, we first decompose $(\mathbf{P}_0)$ into a master problem $(\mathbf{MP}_0)$ for the offloading decision and a subproblem $(\mathbf{SP}_0)$ for the resource allocation. Then, the FFBD algorithm finds the optimal solution of $(\mathbf{P}_0)$ by iteratively solving $(\mathbf{MP}_0)$ and $(\mathbf{SP}_0)$ at either the cloud server or a fog node.

$$(\mathbf{MP}_0) \quad \min_{\mathbf{x} \in \mathbf{X}_0} \{\mathbf{e}^\top \mathbf{x} | cuts^{(k)}\}, \quad (14)$$

and

$$(\mathbf{SP}_0) \quad \min_{\mathbf{x}^{(k)}, \mathbf{r} \in \mathbf{R}_0} \{0\}, \quad (15)$$

where $\{0\}$ is the zero constant function, Benders cutting-planes $cuts^{(k)}$ are restrictions on integer offloading solution $\mathbf{x}^{(k)}$ of $(\mathbf{MP}_0)$ at iteration (k).

From Eq. (14) and (15), the cost function of $(\mathbf{P}_0)$ is identical with that of $(\mathbf{MP}_0)$. $(\mathbf{SP}_0)$ only verifies if integer offloading solution $\mathbf{x}^{(k)}$ of $(\mathbf{MP}_0)$ leads to a feasible resource allocation solution $\mathbf{r}$. Theorem 2 shows that the iteration can

stop when a feasible solution $(\mathbf{x}, \mathbf{r})$ is found or $(\mathbf{MP}_0)$ is infeasible. Additionally, in FFBD, Benders cuts are created by projecting subsets of the finite set of computational tasks into the resource space of fog nodes. Consequently, the number of Benders cuts is limited. Thus, FFBD always returns the optimal solution after a limited number of iterations.

Distributed Subproblems

At iteration (k), by solving the integer programming problem $(\mathbf{MP}_0)$ the offloading decision variables are fixed $\mathbf{x}^{(k)}$. Solution $\mathbf{x}^{(k)}$ determines every task to be processed at either mobile device, one of fog nodes, or the cloud server via a fog node. Thus, $(\mathbf{SP}_0)$ is equivalently divided into a set of M independent resource allocation subproblems of M fog nodes.

Let $\mathbf{x}_j = (\mathbf{x}_j^f, \mathbf{x}_j^c)$ be variables defining the offloading decisions of N tasks onto fog node j or the cloud server (via fog node j), where $\mathbf{x}_j^f = (x_{1j}^f, \dots, x_{Nj}^f)$ and $\mathbf{x}_j^c = (x_{1j}^c, \dots, x_{Nj}^c)$. Here, $\mathbf{x}_j$ is a part of $\mathbf{x} = \bigcup_{j=1}^M \mathbf{x}_j$.

Without loss of generality, we assume $\mathbb{N}_j^t$ and $\mathbb{N}_j^s$, respectively, be the sets of tasks to be processed at fog node j and at the cloud server via fog node j. Here, $\mathbb{N}_j^t$ and $\mathbb{N}_j^s$ are equivalently determined by two offloading decision variables $\mathbf{x}_j^{f(k)}$ and $\mathbf{x}_j^{c(k)}$ in $\mathbf{x}^{(k)}$. We can write $\mathbb{N}_j^t = \{1, \dots, t\}$, $\mathbb{N}_j^s = \{t+1, \dots, t+s\}$, and $\mathbb{N}_j^{t+s} = \mathbb{N}_j^t \cup \mathbb{N}_j^s = \{1, \dots, t+s\}$ is defined by $\mathbf{x}_j^{(k)} = (\mathbf{x}_j^{f(k)}, \mathbf{x}_j^{c(k)})$. Variable $\mathbf{r}_j = (\mathbf{r}_{1j}, \dots, \mathbf{r}_{(t+s)j})$ is resources allocation of fog node j towards its assigned set of tasks $\mathbb{N}_j^{t+s}$. Noticeably, fog node j does not need to allocate the resources towards other tasks except $\mathbb{N}_j^{t+s}$. The resource allocation problem at fog node j can be defined as

$$(\mathbf{SP}_1) \quad \min_{\mathbf{x}_j^{(k)}, \mathbf{r}_j \in \mathbf{R}_j} \{0\}, \qquad (16)$$

where

$$(\mathbf{R}_j) \begin{cases} (\mathcal{C}_{1j}) \ T_i \leq t_i^r, \forall i \in \mathbb{N}_j^{t+s}, \\ (\mathcal{C}_{2j}) \ \sum_{i \in \mathbb{N}_j^t} r_{ij}^f \leq R_j^f, \\ (\mathcal{C}_{3j}) \ \sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^u \leq R_j^u, \\ (\mathcal{C}_{4j}) \ \sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^d \leq R_j^d, \\ r_{ij}^f, r_{ij}^u, r_{ij}^d \geq 0, \forall i \in \mathbb{N}_j^{t+s}, \\ r_{ij}^f = 0, \forall i \in \mathbb{N}_j^s. \end{cases} \qquad (17)$$

Thus, instead of solving $(\mathbf{SP}_0)$, all subproblems $(\mathbf{SP}_1)$ can be solved distributedly among fog nodes in cooperation with the cloud server for $(\mathbf{MP}_0)$. Besides, these subproblems $(\mathbf{SP}_1)$ can also be solved in parallel. Fig. 3 shows the model of the distributed FFBD method.

THEOREM 2. *At any iteration (k), if a feasible solution $(\mathbf{x})$ of $(\mathbf{MP}_0)$ leads to a feasible solution $(\mathbf{r})$ of $(\mathbf{SP}_0)$. Then, $(\mathbf{x}, \mathbf{r})$ is the optimal solution of the original problem $(\mathbf{P}_0)$.*

At any iteration (k), if the master problem $(\mathbf{MP}_0)$ is infeasible, then the original problem $(\mathbf{P}_0)$ is infeasible.

Proof: The detailed proof is presented in Appendix B.

In practice, the solution of $(\mathbf{SP}_1)$ can be found by solving its equivalent problem $(\mathbf{SP}_2)$ with additional slack variables $\mathbf{z}$ using any solver.

$$(\mathbf{SP}_2) \quad \min_{\mathbf{x}_j^{(k)}, \mathbf{r}_j \in \mathbf{RZ}_j} (z_1 + z_2 + z_3), \qquad (18)$$

where

$$(\mathbf{RZ}_j) \begin{cases} T_i \leq t_i^r, \forall i \in \mathbb{N}_j^{t+s}, \\ \sum_{i \in \mathbb{N}_j^t} r_{ij}^f - z_1 \leq R_j^f, \\ \sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^u - z_2 \leq R_j^u, \\ \sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^d - z_3 \leq R_j^d, \\ r_{ij}^f, r_{ij}^u, r_{ij}^d \geq 0, \forall i \in \mathbb{N}_j^{t+s}, \\ r_{ij}^f = 0, \forall i \in \mathbb{N}_j^s, \\ z_1, z_2, z_3 \geq 0. \end{cases} \qquad (19)$$

If $(\mathbf{SP}_2)$ is feasible and its cost function is zero, then $(\mathbf{SP}_1)$ is feasible. Otherwise, $(\mathbf{SP}_1)$ is infeasible.

At iteration (k), if $(\mathbf{SP}_1)$ is feasible at every fog nodes, then $\mathbf{x}^{(k)}$ and $\mathbf{r} = (\mathbf{r}_1, \dots, \mathbf{r}_M)$ are optimal solution of $(\mathbf{P}_0)$. Otherwise, if $(\mathbf{SP}_1)$ is infeasible at fog node j, a new cutting-plane $c_j^{(k)}$ will be added to the cut set of $(\mathbf{MP}_0)$ for the next iteration: $cuts^{(k+1)} = cuts^{(k)} \cup c_j^{(j)}$. The details of cutting-planes generation are in the later *Cutting-Plane Generation* section.

Fast Feasibility and Infeasibility Detection

Normally, FFBD can work by repeatedly solving $(\mathbf{MP}_0)$ and M independent subproblems of the form $(\mathbf{SP}_1)$ using solvers, then update the cutting-plane set $cuts^{(k+1)} = cuts^{(k)} \cup c_j^{(j)}$. The closer to the optimal binary offloading decisions, the less number of iterations the master problem $(\mathbf{MP}_0)$ is solved. Moreover, in many cases, we can quickly determine if $(\mathbf{SP}_1)$ is feasible or not without using any solver. Consequently, the computation time is reduced. Theoretical analyses below help to improve the efficiency of the FFBD algorithm.

From Eq. (3) and (5), the delay constraint $(\mathcal{C}_{1j})$ $T_i \leq t_i^r$ in $(\mathbf{R}_j)$ of $(\mathbf{SP}_1)$ can be rewritten as

$$\begin{cases} \left(\frac{D_i^i}{r_{ij}^u} + \frac{D_i^o}{r_{ij}^d} + \frac{C_i}{r_{ij}^f} \right) \leq t_i^r, & \forall i \in \mathbb{N}_j^t \\ \left(\frac{D_i^i}{r_{ij}^u} + \frac{D_i^o}{r_{ij}^d} \right) \leq t_i^r - \left(\frac{(D_i^i + D_i^o)}{r^{fc}} + \frac{C_i}{f^c} \right), & \forall i \in \mathbb{N}_j^s. \end{cases} \qquad (20)$$

Remarkably, the component $\left(\frac{(D_i^i + D_i^o)}{r^{fc}} + \frac{C_i}{f^c} \right)$ is a constant. If $\exists i \in \mathbb{N}_j^s, t_i^r - \left(\frac{(D_i^i + D_i^o)}{r^{fc}} + \frac{C_i}{f^c} \right) \leq 0$, then processing task i at the cloud server does not satisfy its delay requirement $T_i \leq t_i^r$. In other words, $(\mathbf{SP}_1)$ is infeasible. A Benders cut to prevent offloading task i to the cloud server can be created directly for this case. Otherwise, if $t_i^r - \left(\frac{(D_i^i + D_i^o)}{r^{fc}} + \frac{C_i}{f^c} \right) > 0, \forall i \in \mathbb{N}_j^s$, then we define the relative size $(D_i^{i'}, D_i^{o'}, C_i')$ of task i as below.

$$\begin{cases} \left(\frac{D_i^i}{t_i^r}, \frac{D_i^o}{t_i^r}, \frac{C_i}{t_i^r} \right), & \forall i \in \mathbb{N}_j^t \\ \left(\frac{D_i^i}{\left(t_i^r - \frac{(D_i^i + D_i^o)}{r^{fc}} - \frac{C_i}{f^c} \right)}, \frac{D_i^o}{\left(t_i^r - \frac{(D_i^i + D_i^o)}{r^{fc}} - \frac{C_i}{f^c} \right)}, 0 \right), & \forall i \in \mathbb{N}_j^s. \end{cases} \qquad (21)$$

Let $\beta_i = \left(\frac{D_i^{i'}}{r_{ij}^u} + \frac{D_i^{o'}}{r_{ij}^d} + \frac{C_i'}{r_{ij}^f} \right)$ be the satisfaction rate of Task i. The delay constraint in Eq. (20) becomes

$$\beta_i = \left(\frac{D_i^{i'}}{r_{ij}^u} + \frac{D_i^{o'}}{r_{ij}^d} + \frac{C_i'}{r_{ij}^f} \right) \leq 1, \forall i \in \mathbb{N}_j^{t+s}. \qquad (22)$$

Two theorems below will fast check the feasibility and infeasibility of $(\mathbf{SP}_1)$.

THEOREM 3. *Define balancing rates* $\beta_{bal}^u = \frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{i'}}{R_j^u}$, $\beta_{bal}^d = \frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{o'}}{R_j^d}$, *and* $\beta_{bal}^f = \frac{\sum_{i \in \mathbb{N}_j^{t+s}} C_i'}{R_j^f}$. *If* $\beta_{bal} = \beta_{bal}^u + \beta_{bal}^d + \beta_{bal}^f \leq 1$, *then the problem* $(\mathbf{SP}_1)$ *is feasible.*

Proof. We need to show a feasible solution of $(\mathbf{SP}_1)$.

Task I_i will be allocate resources r_{ij}^u, r_{ij}^d and r_{ij}^f as $r_{ij}^u = \frac{D_i^{i'}}{\beta_{bal}^u}$, $r_{ij}^d = \frac{D_i^{o'}}{\beta_{bal}^d}$, and $r_{ij}^f = \frac{C_i'}{\beta_{bal}^f}$. We have $\beta_i = \frac{D_i^{i'}}{r_{ij}^u} + \frac{D_i^{o'}}{r_{ij}^d} + \frac{C_i'}{r_{ij}^f} = \left(\beta_{bal}^u + \beta_{bal}^d + \beta_{bal}^f \right)$. Here, $r_{ij}^f = 0$ and $\frac{C_i'}{r_{ij}^f} = 0, \forall i \in \mathbb{N}_j^s$. Thus, $\beta_i = \beta_{bal} \leq 1, \forall i \in \mathbb{N}_j^{t+s}$.

Besides, $\sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^u = R_j^u$, $\sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^d = R_j^d$ and $\sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^f = R_j^f$ satisfying resource limit conditions. In conclusion, the problem $(\mathbf{SP}_1)$ is feasible. ∎

Corollary 3.1. *Let* $\beta_{bal}^f = \frac{\sum_{i \in \mathbb{N}_j^t} C_i'}{R_j^f}$, $\gamma_{bal}^u = \frac{\sum_{i \in \mathbb{N}_j^t} D_i^{i'}/(1-\beta_{bal}^f) + \sum_{i \in \mathbb{N}_j^s} D_i^{i'}}{R_j^u}$, *and* $\gamma_{bal}^d = \frac{\sum_{i \in \mathbb{N}_j^t} D_i^{o'}/(1-\beta_{bal}^f) + \sum_{i \in \mathbb{N}_j^s} D_i^{o'}}{R_j^d}$.

If $\beta_{bal}^f \leq 1$ *and* $\gamma_{bal} = \gamma_{bal}^u + \beta_{bal}^d \leq 1$, *then the problem* $(\mathbf{SP}_1)$ *is feasible.*

The Corollary 3.1 is derived from Theorem 3. Fog node j first allocates the computation resource toward tasks in the set $\mathbb{N}_j^t$, which are processed at the fog node, then allocates both the uplink and downlink resources toward all tasks in $\mathbb{N}_j^{t+s}$ based on their remaining delay requirements. Noticeably, tasks in $\mathbb{N}_j^{t+s}$ are either processed at fog node j or forwarded to the cloud server by this fog node. Then, we apply sequentially Theorem 3 to β_{bal}^f and γ_{bal}.

Lemma 1. *Assume variables* $p_i \geq 0$, $q_i > 0$, $\forall i \in N$, *satisfying conditions:* $\sum_{i \in N} p_i = P$ *and* $\sum_{i \in N} q_i = Q$. *We have* $\max_{i \in N}\{\frac{p_i}{q_i}\} \geq \frac{P}{Q}$.

Proof: The detailed proof is presented in Appendix C.

THEOREM 4. *If* $\frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{i'}}{R_j^u} > 1$ *or* $\frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{o'}}{R_j^d} > 1$ *or* $\frac{\sum_{i \in \mathbb{N}_j^{t+s}} C_i'}{R_j^f} > 1$, *then the problem* $(\mathbf{SP}_1)$ *is infeasible.*

Proof: The detailed proof is presented in Appendix D.

Fast Feasibility Detection: Theorem 3 can support to find a feasible solution of the subproblem at fog node j with assigned tasks $\mathbb{N}_j^{t+s}$ so that using any solver is not necessary. Consequently, the computation time is reduced. Especially, for the case of the ratio of $D_i^{i'}$, $D_i^{o'}$ and C_i' approximately equal

between all tasks $\mathbb{N}_j^{t+s}$, this theorem matches almost feasible subproblems.

Corollary 3.1 is theoretically stronger than Theorem 3 because it repeatedly applies the theorem to the computation resource then calculates the remain delay requirements to allocate the uplink and downlink resources. Therefore, we can use Corollary 3.1 instead of Theorem 3 for feasibility detection.

Fast Infeasibility Detection: While Theorem 3 for feasible solution finding, in many cases, we can use Theorem 4 to detect the infeasibility of a subproblem. Especially, the cutting-planes based on Theorem 4 are useful for the large scale system (e.g., a thousand of tasks and a hundred of edge nodes). For the case that a fog node can approximately support a maximum of 10 tasks, these cutting-planes can avoid the generation of subproblems with more than 10 assigned tasks.

Cutting-Plane Generation

In this paper, we introduce three types of cutting-planes which will be updated in $(\mathbf{MP}_0)$, namely "Resource Cutting-Plane", "Subproblem Cutting-Plane", and "Prefixed Decision Cutting-Plane".

Resource Cutting-Plane: From Theorem 4, it dictates that every subset of tasks $\mathbb{N}_j^{t+s} \subseteq \mathbb{N}$ must not violate the uplink, downlink and computation resources constraints at every fog node j.

Let $\mathbf{c}_j^{u(fog)} = (D_1^{i'}, \ldots, D_N^{i'})/R_j^u$, $\mathbf{c}_j^{d(fog)} = (D_1^{o'}, \ldots, D_N^{o'})/R_j^d$ and $\mathbf{c}_j^{f(fog)} = (C_1', \ldots, C_N')/R_j^f$. Here, $(D_i^{i'}, D_i^{o'}, C_i')$ is calculated as in Eq. (21) for $i \in \mathbb{N}_j^t$.

Let $\mathbf{c}_j^{u(cloud)} = (D_1^{i'}, \ldots, D_N^{i'})/R_j^u$, $\mathbf{c}_j^{d(cloud)} = (D_1^{o'}, \ldots, D_N^{o'})/R_j^d$ and $\mathbf{c}_j^{f(cloud)} = (C_1', \ldots, C_N')/R_j^f$. Here, $(D_i^{i'}, D_i^{o'}, C_i')$ is calculated as in Eq. (21) for $i \in \mathbb{N}_j^s$.

To eliminate all the subsets of $\mathbb{N}$ which violates the resources constraints at edge node j, we add three following Benders cuts into *cuts* set of the Master problem $(\mathbf{MP}_0)$:

$$c_j^u = \{\mathbf{c}_j^{u(fog)\top}\mathbf{x}_j^f + \mathbf{c}_j^{u(cloud)\top}\mathbf{x}_j^c \leq 1\},$$

$$c_j^d = \{\mathbf{c}_j^{d(fog)\top}\mathbf{x}_j^f + \mathbf{c}_j^{d(cloud)\top}\mathbf{x}_j^c \leq 1\}, \text{ and}$$

$$c_j^f = \{\mathbf{c}_j^{f(fog)\top}\mathbf{x}_j^f + \mathbf{c}_j^{f(cloud)\top}\mathbf{x}_j^c \leq 1\}.$$

From Eq. (21), we have $\mathbf{c}_j^{f(cloud)} = \mathbf{0}$. Therefore, c_j^f can be shorten as

$$c_j^f = \{\mathbf{c}_j^{f(fog)\top}\mathbf{x}_j^f \leq 1\}.$$

Subproblem Cutting-Plane: At iteration (k), fog node j is assigned a set of tasks $\mathbb{N}_j^{t+s} = \mathbb{N}_j^t \cup \mathbb{N}_j^s$, which is defined by offloading decision $\mathbf{x}_j^{(k)} = (\mathbf{x}_j^{f(k)}, \mathbf{x}_j^{c(k)})$.

If the resource allocation problem $(\mathbf{SP}_1)$ at fog node j is infeasible, then any resource allocation problem at edge node j with assigned tasks $\mathbb{N}_j \supseteq \mathbb{N}_j^{t+s}$ is infeasible. Thus, to eliminate the all subproblems at edge node j containing $\mathbb{N}_j^{t+s}$, a new Benders cut $c_j^{(k)}$ is added into *cuts* set of the Master problem $(\mathbf{MP}_0)$ after iteration (k).

$$c_j^{(k)} = \{\mathbf{x}_j^{f(k)\top}\mathbf{x}_j^f + \mathbf{x}_j^{c(k)\top}\mathbf{x}_j^c \leq t + s - 1\}$$

Prefixed Decision Cutting-Plane: If task I_i satisfies $E_i^l < E_{ij}^f$ and $T_i^l \leq t_i^r$, then it can be pre-decided as local processing. As mentioned in *Fast Feasibility and Infeasibility Detection*, if $\left(t_i^r - \frac{(D_i^i + D_i^o)}{r^{fc}} - \frac{C_i}{f^c}\right) \leq 0$, then task I_i could not be processed at the cloud cluster. In these cases, the suitable cutting-planes can be created and added to set $cuts$ of $(\mathbf{MP}_0)$.

Algorithm 2: FFBD Algorithm

Input : Set of N tasks $\{I_i\left(D_i^i, D_i^o, C_i, t_i^r\right)\}$
         Set of M fog nodes $\{Node_j(R_j^u, R_j^d, F_j^f)\}$
         Cloud server r^{fc}, f^c

Output: Optimal solution $(\mathbf{x}, \mathbf{r})$ of Problem $(\mathbf{P}_0)$

1 **begin**
2   Initialize k and $cuts^{(k)}$ as in **Initialization**.
3   **while** solution $(\mathbf{x}, \mathbf{r})$ has not been found **do**
4     $\mathbf{x} \leftarrow$ <u>Solve</u> $(\mathbf{MP}_0)$ with $cuts^{(k)}$ as in **Master Problem**. ▷ $\mathbf{x}$ store solution $\mathbf{x}^{(k)}$ at iteration k
5   **if** $\mathbf{x}$ is feasible **then**
6   Based on $\mathbf{x}$, create M subproblems $(\mathbf{SP}_1)$ with asigned tasks $\mathbb{N}_1^{t+s}, \ldots, \mathbb{N}_M^{t+s}$.
7   **end**
8   **else**
9   **Return** Problem $(\mathbf{P}_0)$ is infeasible.
10   **end**
11   **for** $(j = 1; \; j \leq M; \; j = j + 1)$ **do**
12       $\mathbf{r}_j \leftarrow$ <u>Solve</u> $(\mathbf{SP}_1)$ at fog node j with task set $\mathbb{N}_j^{t+s}$ as in **Subproblems**.
13   **if** $\mathbf{r}_j$ is infeasible **then**
14   Add a new Benders cut $c_j^{(k)}$ into $cuts^{(k+1)}$ as in **Subproblems**.
15   **end**
16   **end**
17   **if** $\mathbf{r} = (\mathbf{r}_1 \cup \mathbf{r}_2 \cup \cdots \cup \mathbf{r}_M)$ is feasible **then**
18   Solution $(\mathbf{x}, \mathbf{r})$ has been found.
19   **end**
20     $k \leftarrow (k + 1)$ ▷ Increase iteration index
21   **end**
22   **Return** $\mathbf{x}$ and $\mathbf{r}$
23 **end**

FFBD Procedure

Using the aforementioned analyses, we define the procedure of the distributed FFBD algorithm, modeled in Fig. 3. At iteration (k), based on the offloading decision solution $\mathbf{x}^{(k)}$ of $(\mathbf{MP}_0)$, all the tasks in $\mathbb{N}$ are fixed their offloading decisions. Assume $\mathbb{N}_j^{s+t} \subseteq \mathbb{N}$ be the set of tasks assigned to fog node j. Then every fog node j independently solves their own resources allocation problem of the form $(\mathbf{SP}_1)$ toward $\mathbb{N}_j^{s+t}$. Naturally, this process works in a distributed/parallel manner. The Feasibility-Finding Benders decomposition, Algorithm 2, is stated as bellows.

- **Initialization:** Set the iterator $k = 1$. Then, initialize $cuts^{(k)}$ in $(\mathbf{MP}_0)$ with $3M$ resource cutting-planes as in

Resource Cutting-Plane:

$$cuts^{(k)} = \bigcup_{j=1}^{M}\{c_j^u, c_j^d, c_j^f\}.$$

Other Benders cuts, as in *Prefixed Decision Cutting-Plane*, are also added to the $cuts^{(k)}$ of $(\mathbf{MP}_0)$.

- **Master Problem:** At iteration (k), $(\mathbf{MP}_0)$ is solved to find $\mathbf{x}^{(k)} \in X_0$ satisfying $cuts^{(k)}$. Here, $\mathbf{x}^{(k)}$ defines M subproblems of the form $(\mathbf{SP}_1)$. If $(\mathbf{MP}_0)$ is infeasible, then FFBD is terminated with the infeasibility of $(\mathbf{P}_0)$. If $(\mathbf{MP}_0)$ with its solution $\mathbf{x}^{(k)}$ leads to a feasible solution $\mathbf{r} = (\mathbf{r}_1 \cup \mathbf{r}_2 \cup \cdots \cup \mathbf{r}_M)$ of M subproblems of the form $(\mathbf{SP}_1)$, then FFBD is terminated, and $(\mathbf{x}^{(k)}, \mathbf{r})$ is the optimal feasible solution of the original problem $(\mathbf{P}_0)$.

- **Subproblems:** At iteration (k), fog node j independently solves $(\mathbf{SP}_1)$ toward its own assigned $\mathbb{N}_j^{t+s}$ tasks. Before calling a solver, Theorem 3 is used to check its feasibility. If $(\mathbf{SP}_1)$ is infeasible, then a new Benders cut $c_j^{(k)}$ as in *Subproblem Cutting-Plane* is created and added into $cuts^{(k+1)}$ of $(\mathbf{MP}_0)$ for the next iteration $(k + 1)$:

$$cuts^{(k+1)} = cuts^{(k)} \cup \{c_j^{(k)}\}.$$

In Algorithm 2, Theorem 3 is used to check the feasibility of $(\mathbf{SP}_1)$ before calling a solver. Additionally, based on Theorem 4 the resource cutting-planes are created at the initial stage of $(\mathbf{MP}_0)$. Thus, the subproblems violating Theorem 4 are prevented during iterations. Consequently, the computation time of FFBD is reduced.

E. Complexity Analysis

The computation time necessary to find an optimal solution depends on many factors, i.e., the size of problems, the implementation, the runtime environment, and how close to the optimal solution of intermediate results, which is influenced by the data set and the initial point. These are big topics so that we only analyze briefly one factor of the complexity in term of problems' size.

With N tasks and M fog nodes, the original problem $(\mathbf{P}_0)$ has respectively $N(2M + 1)$ integer and $3NM$ real variables, and $(2N + 3M)$ constraints including N for the offloading decisions, N for the delay of the tasks and $3M$ for the resources requirements of the fog nodes as described in Eq. (10) and Eq. (11). Thus, the relaxed problem $(\widetilde{\mathbf{P}}_0)$ of $(\mathbf{P}_0)$ has totally $N(5M + 1)$ real variables and $(2N + 3M)$ constraints.

As so, the approximation method, ROP, has to solve two problems having, respectively, $N(5M + 1)$ real variables and $(2N + 3M)$ constraints, and at most $(3NM)$ real variables and $(2N + 3M)$ constraints. ROP solves these problems only one time for each.

In IBBA, a node at the depth l on the search tree is equivalent to a subproblem with l fixed offloading decisions. As so, the depth of the root node (equivalent to the original problem) is 0, while the depth of a leaf node (at which all tasks are fixed offloading decisions) is N. The subproblem has respectively $(N - l)(2M + 1)$ integer and at most $(3(N - l)M + 3l)$ real

variables, and at most $(2N + 3M - l)$ constraints including $(N - l)$ for the offloading decisions, at most N for the delay of the tasks and $3M$ for the resources requirements of the fog nodes. Consequently, the node at level l of the tree has to solve a relaxed problem having at most $(N(5M + 1) - l(5M - 2))$ variables and at most $(2N + 3M - l)$ constraints. We can see that the number of variables of intermediate problems in IBBA is at least $l(5M - 2)$ less than that of the original one. Besides, there are $(2M + 1)^l$ subproblems at the level l of the $(2M + 1)$-tree. However, most of them are eliminated without solving.

In FFBD, the master problem $(\mathbf{MP}_0)$ and M subproblems of the form $(\mathbf{SP}_1)$ are repeatedly solved. At iteration k, $(\mathbf{MP}_0)$ is an integer problem with $N(2M + 1)$ variables and at least $(N + 3M + k)$ constraints including N for the offloading decisions as in Eq. (11), $3M$ for resource cutting-planes as in the **Initialization** step, and at least k for the sub-problem cutting-planes from solving M subproblems k times as in **Subproblem** step. Besides, each subproblem $(\mathbf{SP}_1)$ is assigned an average of N/M tasks. Thus, it has approximate $3N/M$ resources allocation variables and $(N/M + 3)$ constraints including N/M for the delay of N/M tasks and 3 for the resources requirements at the fog node as described in Eq. (17). In the worst case, $(\mathbf{SP}_1)$ is assigned all N tasks, thus, has at most $3N$ resources allocation variables and $(N + 3)$ constraints. However, if this big subproblem violates the resources constraints at the fog node according to Theorem 4, it could not be created due to the resources cutting-planes generation in the **Initialization** step.

F. Offloading Analysis

Before conducting experiments, we analyze when mobile users can benefit from offloading. Let α_i be the ratio between the number of required CPU cycles C_i and input data size D_i^i. We have $C_i = \alpha_i \times D_i^i$. Then, the local consumed energy is $E_i^l = v_i C_i = \alpha_i v_i D_i^i$.

A mobile user is said to benefit from offloading if its total energy consumption from task offloading is lower than processing locally. Thus, for task i, offloading will benefit if $E_i^l > E_i^f$. In other words, we have $\alpha_i v_i D_i^i > e_{ij}^u D_i^i + e_{ij}^d D_i^o$.

Let α_i^* be the task complexity ratio at which $E_i^l = E_i^f$. We have:

$$\alpha_i^* = \frac{e_{ij}^u D_i^i + e_{ij}^d D_i^o}{v_i D_i^i}. \tag{23}$$

Thus, task i is likely to be offloaded if $E_i^l > E_i^f$ or $\alpha_i > \alpha_i^*$. The task complexity α_i is especially important in evaluating offloaded tasks as well as analyzing the performance of the whole system.

IV. PERFORMANCE EVALUATION

A. Experiment Setup

We use the configuration of a Nokia N900 mobile device described in [34] 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. In the IoT ecosystem, offloading demand applications

often have different characteristics in term of tasks' data size and complexity. These applications share the same communication and computation resources of fog nodes and the cloud server. Therefore, it is reasonable to choose randomly data size and complexity. 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 $U(a, b)$MB and $U(c, d)$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. All parameters are given in Table I.

TABLE I: Experimental parameters

Parameters	Value
Number of mobile devices N	10
Number of fog 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	$U(a, b)$ MB
Output data size D_i^o	$U(c, d)$ 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	$[1, 10]$s
Processing rate of each fog node R_j^f	10 Giga cycles/s
Uplink data rate of each fog node R_i^u	72 Mbps
Downlink data rate of each fog 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

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 fog nodes or the cloud server then minimized the average delay of all tasks as the "All Offloading" (AOP). Due to the simplicities of WOP, AOP, and ROP, these policies are modeled using the GAMS language and solved by the ANTIGONE solver. We develop the proposed algorithms, IBBA and FFBD, as in Algorithm 1 and 2 using the Optimizer API of the **MOSEK** solver [35]. To evaluate the efficiency of theoretical proposals, each of these methods is implemented with two variants. Specifically, the IBBA variant in which tasks are branched in the priority order of local device → fog nodes → cloud server is denoted as IBBA-LFC, whereas IBBA-LCF is the name of the IBBA variant with branching in the priority order of local device → cloud server → fog nodes. In other words, IBBA-LFC tries to offload tasks to fog nodes as much as possible before offloading to the cloud server. On the contrary, IBBA-LCF tries to offload to the cloud server before considering fog nodes. Similarly, the FFBD variant using the standard MOSEK solver for subproblems is denoted as FFBD-S, and the one first using the fast solution detection method described in Theorem 3 is denoted as FFBD-F. The results obtained by IBBA-LFC/LCF and FFBD-S/F will be compared with the policies ROP, WOP, and AOP. To compare the computation time of IBBA-LFC/LCF and FFBD-S/F, the same runtime environment is a normal laptop with Intel Core i5 2.30GHz CPU and 8GB of RAM. Each method runs every experiment 10 times continuously, then the performance is calculated as
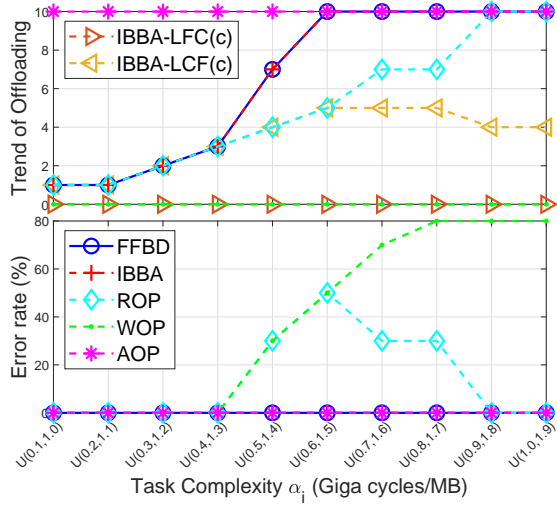
Fig. 4: Trend of offloading and error rate as the task complexity α_i is increased.

the average of these 10 runs.

Noticeably, WOP, AOP, and ROP may not satisfy either the delay constraints of tasks or the consumed energy optimization. Thus, all methods will be evaluated the error rates, defined as the proportion of tasks that do not satisfy their delay constraints.

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, N tasks $I_i\left(D_i^i, D_i^o, C_i, t_i^r\right)$ are generated as $D_i^i \sim U(1.0, 10.0)$MB, $D_i^o \sim U(0.1, 1.0)$MB, $C_i \sim \alpha_i \times D_i^i$ Giga cycles, and the delay requirement t_i^r is set to $10s$ for all tasks. Then, the complexity ratio of tasks α_i starts from $U(0.1, 1.0)$ Giga cycles/MB, then increases each task 0.1 Giga cycles/MB for each experiment. Other parameters are set as in Table I.

Fig. 4 depicts the trend of offloading tasks and error rates when the task complexity ratio α_i is increased. Generally, while the offloading trends of WOP and AOP are constants, i.e., 0 and 10, respectively, the offloading trends (to either fog nodes or the cloud) of the FFBD, IBBA, and ROP methods increase dramatically from 1 to 10 since the task complexity α_i is bigger. This is because when α_i goes up, there are more tasks benefit from offloading since their complexity greater than α_i^*, which is equal to 0.911 Giga cycles/MB according to Eq. (23) and parameters in Table I.

In the IBBA-LFC method, the number of tasks processed at the cloud server (line with label *IBBA-LFC(c)*) is zero. This is because fog nodes have enough communication and computation resources to process all offloading tasks satisfying delay requirements. Whereas in IBBA-LCF, at first the number of tasks processed at the cloud server (line with label *IBBA-LCF(c)*) are equal to the number of offloading tasks, then only a proportion of offloaded tasks (e.g., 4 of 7 offloaded tasks at $\alpha_i = U(0.5, 1.4)$) is processed at the cloud server when

$\alpha_i \geq U(0.5, 1.4)$. This is because the cloud server does not have enough resources to process all offloaded tasks satisfying their delay requirements. In these situations, some tasks will be processed at fog nodes. In the FFBD-S/F methods, we use the default solver to find the solutions of the integer master problem $(\mathbf{MP}_0)$, therefore their offloading decisions to either fog nodes or the cloud server are not analyzed here.

Fig. 4 also shows error rates, which are the proportion of tasks violating their delay requirements. The zero error rates indicate the reliability of the FFBD and IBBA methods since their offloading and resource allocation solutions satisfy tasks' delay requirements, $t_i^r = 10s$. Additionally, we can see that the error rate of AOP is also zero in all experiments. In other words, the fog computing system can support all offloading tasks satisfying their delay threshold, $t_i^r = 10s$. Beside, due to the high complexity since $\alpha_i \geq U(0.5, 1.4)$, the local execution does not satisfy the delay requirements of all tasks. Thus, WOP records the increase of the error rate from 30% to 80% for the last six experiments. Noticeably, the error rate of the approximation method, ROP, is proportional to the difference between the offloading rates of ROP and the optimal methods, FFBD and IBBA (e.g., error rate of 30% with 3 offloading task difference at $\alpha_i = U(0.5, 1.4)$ and 50% with 5 offloading task difference at $\alpha_i = U(0.6, 1.5)$).

Fig. 5 shows the average energy consumption of mobile devices for the FFBD, IBBA, ROP, WOP, and AOP methods when α_i increases from $U(0.1, 1.0)$ to $U(1.0, 1.9)$ Giga cycles/MB. Generally, FFBD and IBBA have the lowest energy consumption in comparing with other methods in all experiments which satisfy the delay requirements. Specifically, the AOP's constant energy consumption (i.e., 7.37J/task) is not lower than that of the FFBD and IBBA methods. This is because in AOP, all tasks are offloaded without considering energy benefit. The equality occurs only when $\alpha_i \geq U(0.6, 1.5)$ with all tasks being offloaded in the FFBD, IBBA, and AOP methods. Additionally, Fig. 4 and 5 show that ROP and WOP can be more energy-efficient at some points in comparing with FFBD and IBBA, but they must suffer from latency constraint errors. For examples, when $\alpha_i = U(0.6, 1.5)$ the consumed energy of ROP and WOP are 6.2J/task and 6.6J/task, respectively, whereas that of the optimal methods, i.e., FFBD and IBBA, is 7.4J/task. However, they suffer the equivalent error rates of 50% and 50%. Noticeably, WOP is the method in which all tasks are processed locally, thus, the consumed energy increases linearly according to the task complexity ratio.

Fig. 6 shows the upward trends in the average delay of all policies when the task complexity increases. For the IBBA-LFC/LCF and FFBD-S/F methods, the average delays are always lower than the threshold $t_i^r = 10s$, matching with their zero error rates. Additionally, FFBD-F uses the fast feasible detection method based on Theorem 3, which allocates the whole communication and computation resources of fog nodes among their assigned tasks. Consequently, FFBD-F has the lowest average delay in comparing with IBBA-LFC/LCF and FFBD-S. Noticeably, the fluctuation of delay in FFBD-F is caused by the different distributions of tasks among fog nodes. The lowest average delay of the AOP policy shows that the fog
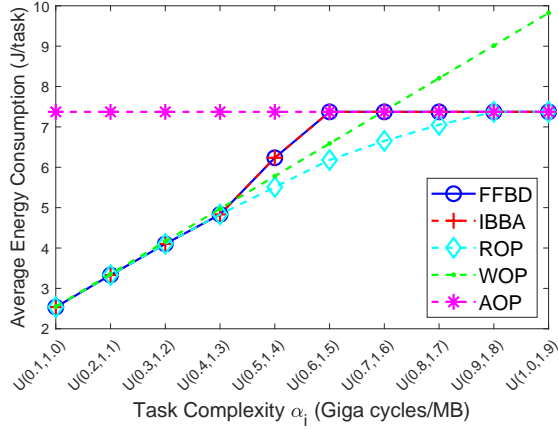
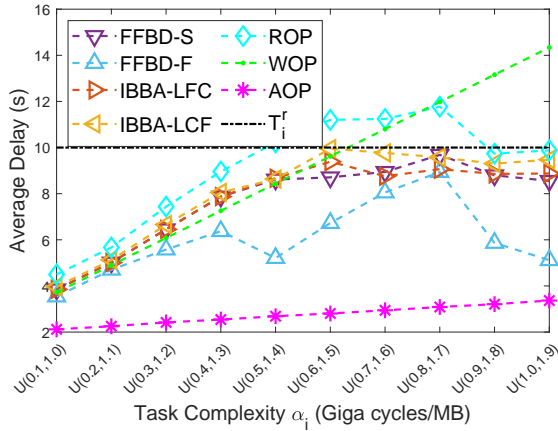Fig. 5: Average consumed energy at mobile devices as the task complexity α_i is increased.

Fig. 6: Average task processing delay as the task complexity α_i is increased.

Fig. 7: Trend of offloading and error rate as the delay requirement is looser.

computing can process all tasks with less than 4s of the delay requirements. Besides, we can see the drawback of the ROP since its average delay is bigger than the threshold $t_i^r = 10$s at all experiments with errors.

2) Scenario 2 - Vary the Task Delay Requirements: In this scenario, we study the impact of task delay requirements on the energy consumption of mobile devices and the computation time of the proposed methods.

In this scenario, N tasks $I_i\left(D_i^i, D_i^o, C_i, t_i^r\right)$ are generated as $D_i^i \sim U(1.0, 10.0)$MB, $D_i^o \sim U(0.1, 1.0)$MB, $C_i \sim \alpha_i \times D_i^i$ Giga cycles, and t_i^r varying between $(2, 10)$s. In comparing with Scenario 1, we choose a wider complexity rate $\alpha_i = U(0.1, 6.0)$. After creating the data set, we detect that there are 5 tasks receiving benefits from offloading $(E_i^l > E_i^f)$ due to $\alpha_i > \alpha_i^* = 0.911$ Giga cycles/MB, and all tasks have the local delay between 2s and 24s. Thus, during experiments, the delay requirement t_i^r of all tasks is vary between 2s and 10s.

Fig. 7 shows the trends of offloading tasks and error rates when the delay requirement is looser. Generally, while the trends of WOP and AOP are constants, i.e., 0 and 10, respectively, the offloading trends (to either fog nodes or cloud servers) of the FFBD and IBBA methods decrease from 9 to 5 tasks since the delay requirement goes up. Specifically, at
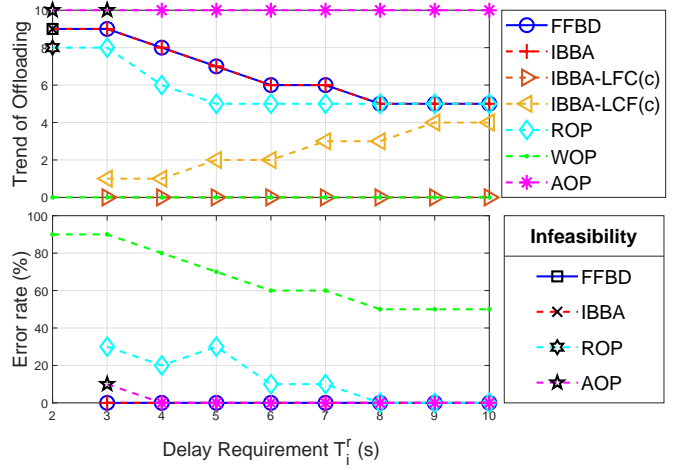
first some tasks without offloading benefits still have to be offloaded due to their high local processing delay $(T_i^l > t_i^r)$. Then, when t_i^r is larger, these tasks will be executed locally to reduce the consumed energy if $T_i^l \leq t_i^r$. Noticeably, the fog computing has not enough resources to accept all offloaded tasks satisfying the delay requirement $t_i^r \leq 3$s. Hence, it is infeasible at $t_i^r = 2$s for the FFBD, IBBA and ROP methods, and at $t_i^r = 2$s and 3s for AOP. Since $t_i^r \geq 8$s, FFBD and IBBA return the optimum solution with only 5 offloaded tasks, which get energy benefit from offloading.

In the IBBA-LFC method, the number of tasks processed at the cloud server (line with label *IBBA-LFC(c)*) is zero because fog nodes have enough communication and computation resources to process all offloading tasks satisfying delay constraints. Whereas in the IBBA-LCF method, the number of tasks processed at the cloud server (line with label *IBBA-LCF(c)*) increases from 1 to 4 when the delay requirment t_i^r gradually grows from 3s to 10s. Due to the lack of resources at the cloud server, some other offloading tasks are processed at fog nodes in order to satisfy their delay requirements.

Fig. 7 also shows the error rate of all methods. Generally, while the error rate of FFBD and IBBA is zero, it is generally decreases for other methods. The without offloading method, WOP, has the highest error rate, steadily decreasing from 90% to 50% when the delay requirement is looser. Besides, in 8 feasible points, the approximation method, ROP, has five experiments with a non-zero error rate when T_i^r is between 3s and 7s. Additionally, at $t_i^r = 3$s, AOP is infeasible, but it is equivalent to the error rate of 10% since FFBD and IBBA are feasible with 9 offloading tasks.

The offloading trends completely match with the average energy consumption depicted in Fig. 8. Generally, while it is a constant for both WOP with 8.5J/task and AOP with 7.4J/task, the consumed energy in FFBD and IBBA decreases from 6.3J/task to 3.5J/task when increasing the delay requirement. Equivalently, both the FFBD and IBBA methods reduce the consumed energy from 15% to 52% and from 26% to 59%, respectively, in comparing with AOP and WOP. Especially, in comparing with the optimal methods, FFBD and IBBA,
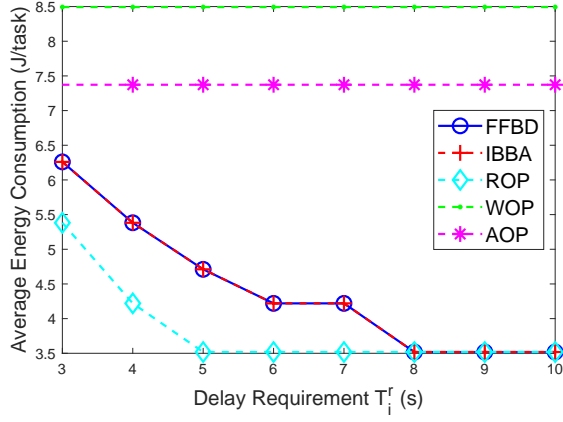
Fig. 8: Average consumed energy at mobile devices when the delay requirement is looser.

Fig. 9: Computation time and number of solved intermediate problems in order to find an optimal solution as the task co...

Fig. 10: Computation time and number of solved intermediate problems in order to find an optimal solution when the delay requirement is looser.

although the ROP method achieves energy benefits at some experiments, it must suffer from latency constraint errors.

From both Scenario 1 and 2, we can conclude that FFBD and IBBA are the optimal methods returning the solutions satisfying constraints. The approximation method, ROP, is not a reliable algorithm when the delay errors often occur.

3) Complexity and Computation Time: In this subsection, we present some results on the complexity of the four algorithms, IBBA-LFC/LCF and FFBD-S/F, and the necessary time to find an optimal solution in each experiment. Here, WOP, AOP, and ROP are ignored due to their inadequacy of the goal.

The complexity of the four algorithms is calculated by the number of intermediate problems (i.e., the intermediate relaxing problems during searching trees in IBBA-LFC/LCF, master problem $(\mathbf{MP}_0)$ and subproblem $(\mathbf{SP}_1)$ solved by the standard solver in FFBD-S/F). In FFBD-F, due to the low complexity, the problems solved by the fast feasible detection method as in Theorem 3 are ignored.

From other angle, the complexity is also evaluated by the number of subproblems, which are either the intermediate relaxing problems in IBBA-LFC/LCF or the subproblem $(\mathbf{SP}_1)$ solved by the standard solver in FFBD-S/F. For the FFBD-S/F methods, the number of master problem iterations also provided. Noticeably, the analyses below are only relative because the computation time and complexity of the algorithms also depends on other factors (e.g., the problems' type and size).

Fig. 9 and 10 show the computation time and the number of intermediate problems being solved since either the task complexity goes up or the delay requirement is looser. Generally, the computation time is proportional to the number of intermediate problems. In both Scenario 1 (the complexity varies) and Scenario 2 (the delay requirement is looser), comparing with IBBA-LFC/LCF, the FFBD-S/F methods are more effective due to their decomposition, initial Benders cuts based on Theorem 4 and the cutting-plane generation from the results of subproblems. Noticeably, when either tasks have a higher complexity or a lower delay requirement, it takes more time for FFBD-S/F and IBBA-LFC/LCF to find
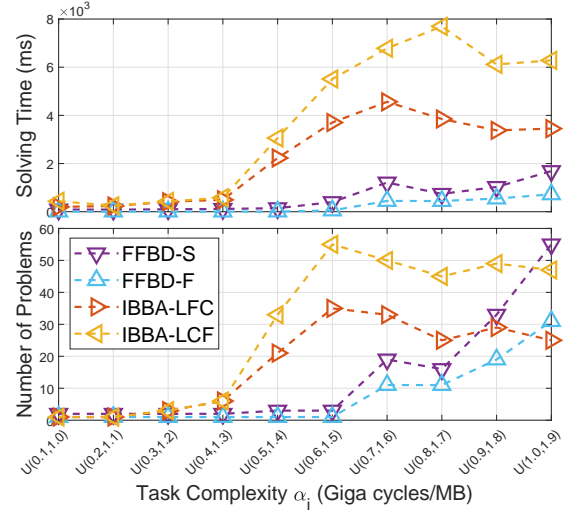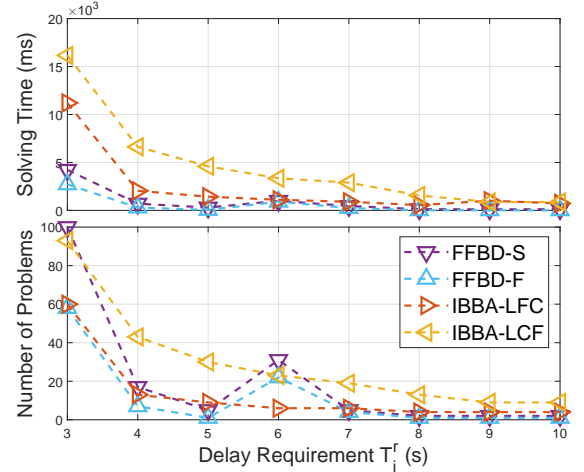
the optimal solutions. This is because the methods must try more intermediate solutions in order to satisfy more resource demands of tasks.

In both scenarios, the FFBD-F method also has better results than FFBD-S due to the implementation of the fast solution finding method described in Theorem 3. Recall that, IBBA-LFC tries to offload tasks to fog nodes as much as possible before offloading to the cloud server. On the contrary, IBBA-LCF tries to offload to the cloud server before considering fog nodes. Besides, the fog nodes tend to "closer" to mobile devices than the cloud server does. Therefore, offloading to fog nodes is easier to satisfy the delay requirements than to the cloud server. Consequently, we can see that IBBA-LFC is more effective than IBBA-LCF in term of computation time.

Table II summarizes the major performance (e.g., minimum, maximum, and average computation time) involving the computation time. For Scenario 1, the FFBD-F, FFBD-S,

TABLE II: Complexity and computation times

Scenario 1: Increasing complexity				
	FFBD-F	**FFBD-S**	**IBBA-LFC**	**IBBA-LCF**
Min. time	5ms	86ms	200ms	248ms
Max. time	721ms	1687ms	4562ms	7694ms
Average time	224ms	561ms	2252ms	3714ms
Average num. of standard solve	3.5 (37.2%)	9.4 (100%)	17.9 (100%)	29 (100%)
Average num. of fast solve	5.90 (62.8%)	NA (0%)	NA (0%)	NA (0%)
Max. MP Iterations	16	16	NA	NA
Average MP Iterations	4.3	4.3	NA	NA
Scenario 2: Vary the task delay requirements				
	FFBD-F	**FFBD-S**	**IBBA-LFC**	**IBBA-LCF**
Min. time	8ms	111ms	548ms	866ms
Max. time	2715ms	4224ms	11206ms	16160ms
Average time	530ms	878ms	2371ms	4615ms
Average num. of standard solve	6.75 (43.9%)	15.38 (100%)	14.5 (100%)	29.88 (100%)
Average num. of fast solve	8.63 (56.1%)	NA (0%)	NA (0%)	NA (0%)
Max. MP Iterations	20	20	NA	NA
Average MP Iterations	5.13	5.13	NA	NA

IBBA-LFC and IBBA-LCF algorithms, respectively, have the average solving time of 224ms, 561ms, 2252ms, and 3714ms equivalent to an average of 3.5, 9.4, 17.9, and 29 times using the standard solver for the subproblems. For Scenario 2, the maximum number of master problem iterations is 20 for FFBD-F/S, and average 8.63 (56.1%) of 15.38 subproblems are solved by the fast feasible method for FFBD-F.

From Fig. 9, 10 and Table II, although FFBD-F/S have to solve much more intermediate problems in some experiments, their computation time is still remarkably lower than that of the IBBA-LFC/LCF methods due to the small size of their intermediate problems. Thus, we can conclude that the computation time depends not only on the number and size of intermediate problems but also their specific properties, which correlate with the distance between the intermediate solutions and the optimal one.

V. CONCLUSION

We have proposed a joint offloading decision and resource allocation optimization framework for three-tier mobile fog computing networks. To find the optimal solution, we have developed three effective algorithms called IBBA with two variants IBBA-LFC/LCF (based on the Branch and bound method), the distributed method, FFBD, with two variants FFBD-S/F (based on the Benders decomposition) and ROP (an approximation policy based on the solution of the relaxing problem). While IBBA-LFC/LCF and FFBD-S/F can find the optimal solution, ROP returns an approximated solution with error rates. FFBD-F implemented the fast feasible detection method is the fasted algorithm in term of the computation time. Whereas, IBBA-LFC/LCF with the priority solution selection strategies can return the most wanted solution. Numerical results have demonstrated the efficiency in terms of energy consumption reduction of the proposed solutions.

APPENDIX A
PROOF OF THEOREM 1

Proof. From Eq. (2), (4), (6), and (8) the objective function, E, is a linear expression of decision variables $\mathbf{x}$. From Eq. (7), 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 is a concave function with respect to $\mathbf{x}$ and $\mathbf{r}$ [29]. Since the objective function in (8) is a linear function, and the constraints in $(\tilde{\mathbf{R}}_0)$ and $(\tilde{\mathbf{X}}_0)$ are concave functions, the relaxed problem $(\tilde{\mathbf{P}}_0)$ is a convex optimization problem [29]. ∎

APPENDIX B
PROOF OF THEOREM 2

Proof. We assume the cutting-plane sets of $(\mathbf{MP}_0)$ at iterations (k) and $(k+1)$ are $cuts^{(k)}$ and $cuts^{(k+1)}$, respectively. At iteration k, assume $(\mathbf{MP}_0)$ is feasible, and there is at least one infeasible subproblem $(\mathbf{SP}_1)$. Consequently, we have $cuts^{(k)} \subset cuts^{(k+1)}$. This leads to $\min_{\mathbf{x} \in \mathbf{X}_0} \{\mathbf{c}^\top \mathbf{x} | cuts^{(k)}\} \leq \min_{\mathbf{x} \in \mathbf{X}_0} \{\mathbf{c}^\top \mathbf{x} | cuts^{(k+1)}\}$. In other words, $\min_{\mathbf{x} \in \mathbf{X}_0} \{\mathbf{c}^\top \mathbf{x} | cuts^{(k)}\}$ is a function that does not decrease with iteration k.

Thus, the first found feasible solution $(\mathbf{x}, \mathbf{r})$ of $(\mathbf{MP}_0)$ and $(\mathbf{SP}_0)$ is the optimal solution of $(\mathbf{P}_0)$.

In the case that $(\mathbf{MP}_0)$ is infeasible at iteration k, it means that $(\mathbf{MP}_0)$ will be infeasible at all later iterations due to $cuts^{(k)} \subset cuts^{(k+v)}, \forall v \geq 1$. In other words, the original problem $(\mathbf{P}_0)$ is infeasible. ∎

APPENDIX C
PROOF OF LEMMA 1

Proof. if $\frac{p_1}{q_1} \geq \frac{p_2}{q_2}$ then $\max\{\frac{p_1}{q_1}, \frac{p_2}{q_2}\} = \frac{p_1}{q_1} \geq \frac{p_1 + p_2}{q_1 + q_2}$. Otherwise, if $\frac{p_1}{q_1} < \frac{p_2}{q_2}$ then $\max\{\frac{p_1}{q_1}, \frac{p_2}{q_2}\} = \frac{p_2}{q_2} > \frac{p_1 + p_2}{q_1 + q_2}$. In other words, $\max\{\frac{p_1}{q_1}, \frac{p_2}{q_2}\} \geq \frac{p_1 + p_2}{q_1 + q_2}$.

Similarly, $\max\{\frac{p_1 + p_2}{q_1 + q_2}, \frac{p_3}{q_3}\} \geq \frac{p_1 + p_2 + p_3}{q_1 + q_2 + q_3}$. Therefore, $\max\{\frac{p_1}{q_1}, \frac{p_2}{q_2}, \frac{p_3}{q_3}\} \geq \max\{\frac{p_1 + p_2}{q_1 + q_2}, \frac{p_3}{q_3}\} \geq \frac{p_1 + p_2 + p_3}{q_1 + q_2 + q_3}$.

Repeatedly, we have $\max_{i \in N}\{\frac{p_i}{q_i}\} \geq \frac{P}{Q}$. ∎

APPENDIX D
PROOF OF THEOREM 4

Proof. Applying Lemma 1 into $\{D_i^{i'}\}_{i \in \mathbb{N}_j^{t+s}}$ and $\{r_{ij}^u\}_{i \in \mathbb{N}_j^{t+s}}$, we have $\max_{i \in \mathbb{N}_j^{t+s}}\{\frac{D_i^{i'}}{r_{ij}^u}\} \geq \frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{i'}}{\sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^u}$.

According to resource allocation conditions, $\sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^u \leq R_j^u$, we have $\max_{i \in \mathbb{N}_j^{t+s}}\{\frac{D_i^{i'}}{r_{ij}^u}\} \geq \frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{i'}}{R_j^u}$. Therefore, $\max_{i \in \mathbb{N}_j^{t+s}}\{\frac{D_i^{i'}}{r_{ij}^u}\} > 1$. Without loss of generality, we assume $\exists i* \in \mathbb{N}_j^{t+s}, \frac{D_{i*}^{i'}}{r_{i*j}^u} = \max_{i \in \mathbb{N}_j^{t+s}}\{\frac{D_i^{i'}}{r_{ij}^u}\} > 1$. Consequently, $\beta_{i*} = \left(\frac{D_{i*}^{i'}}{r_{i*j}^u} + \frac{D_{i*}^{o'}}{r_{i*j}^d} + \frac{C_{i*}^{'}}{r_{i*j}^f}\right) > \frac{D_{i*}^{i'}}{r_{i*j}^u} > 1$. It contradicts the delay

requirement of Task I_{i*}, $\beta_{i*} \leq 1$ as in Eq. (22). In conclusion, the problem $(\mathbf{SP}_1)$ is infeasible.

The cases $\frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{o'}}{R_j^d} > 1$ and $\frac{\sum_{i \in \mathbb{N}_j^{t+s}} C_i'}{R_j^f} > 1$ are proved in the similar way. $\blacksquare$

REFERENCES

[1] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *2012 IEEE Symposium on Computers and Communications (ISCC)*, 2012, Conference Proceedings, pp. 000 059–000 066.

[2] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[3] 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, 2017.

[4] 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, 2017.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[6] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[7] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[8] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, pp. 81–93, 2015.

[9] Y. Kao, B. Krishnamachari, M. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, 2017.

[10] Z. Cheng, P. Li, J. Wang, and S. Guo, "Just-in-time code offloading for wearable computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 1, pp. 74–83, 2015.

[11] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2015.

[12] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, Conference Proceedings, pp. 1–9.

[13] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253–2266, 2015.

[14] V. Cardellini, V. De Nitto Personé, 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. 421–449, 2016.

[15] 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. 2795–2808, 2016.

[16] L. Pu, X. Chen, J. Xu, and X. Fu, "D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[17] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 85–97, 2019.

[18] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[19] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[20] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Energy efficient dynamic offloading in mobile edge computing for internet of things," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[21] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[22] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in c-ran with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2018.

[23] M. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6790–6805, 2018.

[24] M. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2868–2881, 2018.

[25] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[26] T. T. Vu, N. V. Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Offloading energy efficiency with delay constraint for cooperative mobile edge computing networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, Conference Proceedings, pp. 1–6. [Online]. Available: https://arxiv.org/abs/1811.12686

[27] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei, "The benders decomposition algorithm: A literature review," *European Journal of Operational Research*, vol. 259, no. 3, pp. 801–817, 2017.

[28] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *2012 Proceedings IEEE INFOCOM*, 2012, Conference Proceedings, pp. 2716–2720.

[29] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[30] M. Chen, Y. Bao, X. Fu, G. Pu, and T. Wei, "Efficient resource constrained scheduling using parallel two-phase branch-and-bound heuristics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 5, pp. 1299–1314, 2017.

[31] Y. Yu, X. Bu, K. Yang, and Z. Han, "Green fog computing resource allocation using joint benders decomposition, dinkelbach algorithm, and modified distributed inner convex approximation," in *2018 IEEE International Conference on Communications (ICC)*, 2018, Conference Proceedings, pp. 1–6.

[32] M. Fischetti and D. Salvagnin, "A relax-and-cut framework for gomory's mixed-integer cuts," ser. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer Berlin Heidelberg, 2010, Conference Proceedings, pp. 123–135.

[33] M. Fischetti, I. Ljubić, and M. Sinnl, "Benders decomposition without separability: A computational study for capacitated facility location problems," *European Journal of Operational Research*, vol. 253, no. 3, pp. 557–569, 2016.

[34] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," *HotCloud*, vol. 10, pp. 4–4, 2010.

[35] E. D. Andersen and K. D. Andersen, "The mosek documentation and api reference," Report, 2019. [Online]. Available: https://www.mosek.com/documentation/