Optimal Energy Efficiency with Delay Constraints for Multi-layer Cooperative Fog Computing Networks

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

***Abstract*—We develop a joint offloading and resource allocation framework for a multi-layer cooperative fog computing network, aiming to minimize the total energy consumption of multiple mobile devices subject to their service delay requirements. The resulting optimization involves both binary (offloading decisions) and real variables (resource allocations), making it an NP-hard and computationally intractable problem. To tackle it, we first propose an improved branch-and-bound algorithm (IBBA) that is implemented in a centralized manner. However, due to the large size of the cooperative fog computing network, the computational complexity of the proposed IBBA is relatively high. To speed up the optimal solution searching as well as to enable its distributed implementation, we then leverage the unique structure of the underlying problem and the parallel processing at fog nodes. To that end, we propose a distributed framework, namely feasibility finding Benders decomposition (FFBD), that decomposes the original problem into a master problem for the offloading decision and subproblems for resource allocation. The master problem (MP) is then equipped with powerful cutting-planes to exploit the fact of resource limitation at fog nodes. The subproblems (SP) for resource allocation can find their closed-form solutions using our fast solution detection method. These (simpler) subproblems can then be solved in parallel at fog nodes. The numerical results show that the FFBD always returns the optimal solution of the problem with significantly less computation time (e.g., compared with the centralized IBBA approach). The FFBD with the fast solution detection method, namely FFBD-F, can reduce up to 60% and 90% of computation time, respectively, compared with those of the conventional FFBD, namely FFBD-S, and IBBA.**

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

I. Introduction

Emerging mobile applications tend to be more and more demanding in computation (e.g., 3D rendering and image processing) as well as lower latency requirements (e.g., interactive games and online object recognition) [2], [3]. Nonetheless, mobile devices are usually limited in computing resources, battery life, and network connections. As a promising solution, a novel 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 [4]. For that, in a fog computing 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 with lower delay. Thanks to its unique advantages (e.g., low latency and high bandwidth connections with both mobile devices as well as cloud servers, in the proximity of mobile devices, agile mobility and location awareness support), the fog computing architecture [5] has proved itself as an effective solution to enable energy-efficient and low-latency mobile applications [6].

However, not all computational tasks benefit from being offloaded to a fog node. Some tasks even consume more energy when being offloaded than being processed locally due to the communication overhead, i.e., sending requests and receiving results [4], [7]. As such, the task offloading and resource allocation decisions should be jointly considered/optimized. Moreover, unlike public clouds, e.g., Amazon Web Services and Microsoft Azure, a fog/edge node does not possess abundant computing resource [8]. While the computation offloading demand from mobile users is huge, a fog node can support a limited number of tasks. As such the collaboration between fog nodes and cloud servers (referred to as *vertical* collaboration, e.g., [9]–[12]) or amongst fog nodes (referred to as *horizontal* collaboration, e.g., [13]–[15]) is a very promising approach.

Most existing work on joint task offloading and resource allocation of fog computing network only consider either the vertical or horizontal collaboration amongst fog nodes and servers. For example, in [16], a representative of user devices can offload tasks to either nearby fog nodes or a cloud server, aiming to minimize the total delay. However, the authors did not investigate the consumed energy and delay between users and the representative. In [17], the authors developed a joint task offloading and computation resource allocation scheme to minimize the average task duration. This work did not optimize communication resource allocation since the bandwidth was assumed to be equally shared among associated users. With regard to the horizontal collaboration amongst fog nodes, there are very few works [13]–[15]. Authors of [18] proposed the joint communication and computation resource sharing scheme in NOMA-aided cooperative computing system. However, a small-scale model with only a user device, a helper, and an access point is considered. Additionally, there is a rich literature on the energy and delay trade-off in fog computing. For that, in this work we focus on the potential collaboration amongst fog nodes and between fog nodes and the cloud to optimize the decision where is the best for a task to be offloaded to, considering the availability of both computing and communications resources as well as the latency requirement of each task.

Thai T. VU, Diep N. Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz are with University of Technology Sydney, Sydney, NSW 2007, Australia. E-mails: tienthai.vu@student.uts.edu.au, {Diep.Nguyen, Hoang.Dinh, and Eryk.Dutkiewicz}@uts.edu.au.

Thuy V. Nguyen is with the Faculty of Information Technology, Posts and Telecommunications Institute of Technology (PTIT), Hanoi, Vietnam. E-mail: thuynv@ptit.edu.vn.

An abridged version of this paper was presented at the IEEE Globecom Conference, Dec, 2018[1]

Moreover, as aforementioned, although fog nodes' computing capability and resource are more powerful than mobile devices, they are still limited, in comparison with cloud servers. Admitting tasks offloaded from one or a group of mobile devices may prevent it from serving others. Most of the above works and others in the literature tend to overlook this fact, considering only a single-user case. Given the above, this work aims to simultaneously leverage both vertical and horizontal collaboration (amongst fog nodes as well as between fog and cloud server nodes) to jointly optimize the task offloading and resource/computing allocation to minimize the total energy consumption of multiple mobile users (subject to their diverse latency requirements). To that end, we introduce a multi-layer mixed fog and cloud computing system including multiple users, multiple fog nodes, and the remote cloud server. The model allows us to exploit the advantages of both fog nodes and the cloud server as well as enables the scalability due to the collaboration between fog nodes. To overcome the drawbacks of [16] and [17], we investigate all factors (i.e., uplink, downlink and processing) contributing to the overall delay, consumed energy as well as communication resource allocation.

The resulting optimization involves both binary (offloading decisions) and real variables (resource allocations), called a mixed integer non-linear programming problem (MINLP). That makes it an NP-hard and computationally intractable problem. To tackle it, we first propose an improved branch-and-bound algorithm (IBBA) that is implemented in a centralized manner. However, due to the large size of the cooperative fog computing network, the computational complexity of the proposed IBBA is relatively high. To speed up the optimal solution searching as well as to enable its distributed implementation, we then leverage the unique structure of the underlying problem and the parallel processing at fog nodes. To that end, we then propose 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. The master problem (MP) for the offloading decisions is then equipped with powerful cutting-planes based on resource limitation of fog nodes. The subproblems (SP) for resource allocation can find closed-form solutions using our fast solution detection method. These (simpler) subproblems can be then solved in parallel at fog nodes. The numerical results show that the FFBD always returns the optimal solution of the problem with significantly less computation time (e.g., in comparing with the centralized approach). The FFBD with the fast solution detection method can reduce up to 60% and 90% of computation time, respectively, than those of the conventional FFBD and IBBA. Major contributions of this paper are as follows:

- We propose a cooperative computing framework which considers both vertical and horizontal collaboration amongst fog and cloud nodes while minimizing the total energy of all mobile devices, subject to their service delay constraints.
- We then propose an improved branch-and-bound (IBBA) method that exploits special features of our task offloading model to obtain different optimal offloading policies.
- To leverage the computation capability at all fog nodes, we develop a distributed feasibility-finding Benders decomposition (FFBD) algorithm. The algorithm decomposes the original problem into a master problem (MP) for offloading decisions and multiple subproblems (SP) for resources allocation. Exploiting special characteristics of the problem, the subproblems (SP) in the FFBD can be solved independently at edge nodes.
- To further reduce the computation time of FFBD, we develop a theoretical framework for the feasibility and infeasibility detection of the subproblems based on fog nodes' resource limitation. Then, the master problem is equipped with powerful cutting-planes using theoretical analysis on the infeasibility of SPs. The subproblems can then find closed-form solutions using the fast solution detection method.
- We perform intensive simulations to evaluate the efficiency of the proposed framework and solutions, and compare their performance (i.e., the consumed energy, delay, processing time and complexity) with those of the standard solutions. These results provide insightful information on factors affecting the performance of the proposed methods.

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 (IBBA, FFBD-S, FFBD-F with different optimal solution selection criteria), the theoretical analyses, the optimal solution selection strategies. In this section, we also design a protocol to implement the proposed algorithms. In Section IV, we evaluate the performance of proposed algorithms and compare them with different baseline methods. Conclusions are drawn in Section V.

II. System Model and Problem Formulation

A. Network Model

Fig. 1 illustrates a multi-layer fog computing system with N mobile devices $\mathbb{N} = \{1, \ldots, N\}$, M cooperative fog nodes $\mathbb{M} = \{1, \ldots, M\}$, and a cloud server V that can be directly reached by all mobile devices (e.g., the cloud server is co-located with the base station). A mobile device can offload a task to either one of fog nodes, the cloud via a fog node or directly to the cloud server V. We assume a given multi-access method is in place. The proposed scheme and analysis in the sequel can adopt any multi-access method, especially recent non-contention orthogonal ones, that are more suitable for delay-sensitive applications, like NOMA.

At each time slot, mobile user i can request to offload a[1] 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 the mobile device, fog nodes, or the cloud server satisfying the delay requirement are eligible to process the task.

Under the multi-tier/layer model, the highest tier is the cloud server that has the highest CPU rate but would require more energy and latency for the mobile nodes to offload to. On the other hand, the second tier, i.e., fog nodes, has lower CPU rates than the cloud's but is closer to mobile devices. As such,

[1]The following analysis also applies if a mobile user has multiple tasks at the same time

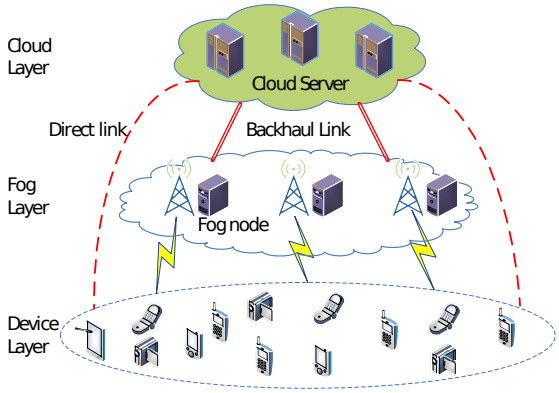


Fig. 1: Three-layer cooperative fog computing network.

in terms of communications latency and energy, fog nodes are more preferable than the cloud for mobile devices to offload on. The last tier is the mobile devices who have the lowest CPU rates but if they select to process the tasks locally, then the communications latency is the lowest or zero, compared with offloading the tasks to the fog or the cloud.

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 time 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], [20].

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 the fog node will allocate radio/communications and computation resources for the mobile device/task I_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 executing the task. 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.

Various physical factors (i.e., channel fading, antenna gain, power level, circuit power, and active/receive time of transmitter/modem) affect energy efficiency of communication at each mobile device [21], [22]. These parameters are readily available at the physical layer can be captured through the energy required per up-/down-link (transmit/receive) data unit.

Let e_{ij}^u and e_{ij}^d denote the energy consumption for transmitting and receiving a unit of data between the mobile device i and the fog node j, respectively. Let ζ denote the multi-access delay to capture the accusation delay of the initial data frame. ζ depends on the frame structure/duration and the multi-access method. The delay T_{ij}^f and the consumed energy E_{ij}^f of mobile device are given by:

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

and

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

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

Note that the impact of multiple users and multi-access is captured through the resource allocation and offloading decisions. For a given up and downlink capacity/constraint, if more users are admitted, each will get less rate. In reality, this is less the number of time slots or carriers or frames (e.g., in TDMA/FDMA or NOMA) to be allocated, resulting longer delay to transmit/receive a given data unit. Additionally, serving a task/user may prevent a given fog node from serving others due to the limited computing capability of the fog node.

3) Cloud Server Processing (offloaded via a fog node): Without loss of generality, we can assume that all fog nodes are connected to a public cloud server with a backhaul capacity/bandwidth rate as W_j^c.

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. All tasks that are processed at the cloud server via fog node j share the backhaul capacity W_j^c and the total CPU capacity F_j^c at the cloud (for the fog node j). We denote the backhaul data rate allocated for task I_i between the fog node j and the cloud server as w_{ij}^c and the processing rate assigned to task I_i on the cloud server as f_{ij}^c. 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 fog node, and task-execution time at the cloud server. These performance metrics are as follows:

$$T_{ij}^c = D_i^i / r_{ij}^u + D_i^o / r_{ij}^d + (D_i^i + D_i^o) / w_{ij}^c + C_i / f_{ij}^c + \zeta, \tag{5}$$

and

$$E_{ij}^c = E_{ij}^f = E_{ij}^u + E_{ij}^d. \tag{6}$$

4) Cloud Server Processing (directly offloaded by mobile devices): Let $\mathbb{M}^* = \mathbb{M} \cup \{V\}$ be the set of M fog nodes and the cloud server V. To simplify the notation in the sequel, the cloud V is also denoted as the $(M+1)$th node of $\mathbb{M}^*$. The direct connection from mobile devices to the cloud is captured by the tuple $(R_{(M+1)}^u, R_{(M+1)}^d, R_{(M+1)}^f)$ in which $R_{(M+1)}^u$, $R_{(M+1)}^d$, and $R_{(M+1)}^f$ are uplink capacity, downlink capacity, and the CPU cycle rate of the cloud, respectively. We have $R_{(M+1)}^f = f^c$.

For a task I_i to be directly offloaded and executed by the cloud V (or the $(M+1)$ node), the cloud will allocate radio/communications and computation resources, defined by a tuple $\mathbf{r}_{i(M+1)} = (r_{i(M+1)}^u, r_{i(M+1)}^d, r_{i(M+1)}^f)$, in which $r_{i(M+1)}^u$, $r_{i(M+1)}^d$, $r_{i(M+1)}^f$ respectively are the cloud's uplink, downlink, and CPU cycle rates for input, output transmissions, and executing the task. The corresponding delay $T_{i(M+1)}^f$ and

energy consumption $E^f_{i(M+1)}$ for this case can then be calculated similarly to those in equation (3) and (4):

$$T^f_{i(M+1)} = D^i_i / r^u_{i(M+1)} + D^o_i / r^d_{i(M+1)} + C_i / r^f_{i(M+1)} + \zeta, \quad (7)$$

and

$$E^f_{i(M+1)} = E^u_{i(M+1)} + E^d_{i(M+1)}, \quad (8)$$

where $E^u_{i(M+1)} = e^u_{i(M+1)} D^i_i$ and $E^d_{i(M+1)} = e^d_{i(M+1)} D^o_i$ (with that $e^u_{i(M+1)}$ and $e^d_{i(M+1)}$ are the energy consumption for transmitting and receiving a unit of data between the mobile device i and the cloud).

Note that for the cloud (i.e., node $(M+1)$), as itself is the top/last tier, it can't choose to offload a task to a higher tier. This is mathematically captured by setting $T^c_{i(M+1)} = \infty$ and $E^c_{i(M+1)}$ as a constant.

B. Problem Formulation

We denote the binary offloading decision variable for task I_i by $\mathbf{x}_i = (x^l_i, x^f_{i1}, \ldots, x^f_{i(M+1)}, x^c_{i1}, \ldots, x^c_{i(M+1)})$, in which $x^l_i = 1$, $x^f_{ij} = 1$, and $x^c_{ij} = 1$ respectively indicate that task I_i is processed at the mobile device, fog node j (or directly at the cloud if $j = M+1$), or the cloud server (via fog node j for $j < M+1$). As mentioned above, the cloud V or the node $(M+1)$th is the top-tier, so it can't offload a task to a higher tier or $x^c_{i(M+1)} = 0$.

Let $\mathbf{h}_i = (T^l_i, T^f_{i1}, \ldots, T^f_{i(M+1)}, T^c_{i1}, \ldots, T^c_{i(M+1)})$. From Eqs. (1), (3), (5), and (7), the delay T_i when task I_i is processed is given as

$$T_i = \mathbf{h}_i^\top \mathbf{x}_i. \quad (9)$$

Note that the delay T_i in Eq. (9) is not a convex function w.r.t. the offloading and resource allocation decisions. This is due to the its non-convex components in the form of the ratio of these variables x/r (where x and r are offloading decision and resource allocation variables, respectively). Consequently, the formulated problem with delay constraints is not a convex optimization problem. To leverage convex optimization, we convert T_i in Eq. (9) to a convex one. Because $x^l_i, x^f_{ij}, x^c_{ij}$ are binary variables, we have $x^l_i = (x^l_i)^2, x^f_{ij} = (x^f_{ij})^2$, and $x^c_{ij} = (x^c_{ij})^2$. Thus, we can equivalently reformulate T_i as

$$T_i = \mathbf{h}_i^\top \mathbf{y}_i, \quad (10)$$

where $\mathbf{y}_i = ((x^l_i)^2, (x^f_{i1})^2, \ldots, (x^f_{i(M+1)})^2, (x^c_{i1})^2, \ldots, (x^c_{i(M+1)})^2)$.

In the rest of the paper we will use T_i described in Eq. (10) for the delay of task I_i. The convexity of T_i in Eq. (10) will be proven and used in Theorem 1.

Let $\mathbf{e}_i = (E^l_i, E^f_{i1}, \ldots, E^f_{i(M+1)}, E^c_{i1}, \ldots, E^c_{i(M+1)})$. From Eqs. (2), (4), (6), and (8), the consumed energy E_i of the mobile user when task I_i is processed are given as

$$E_i = \mathbf{e}_i^\top \mathbf{x}_i. \quad (11)$$

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

$$E = \mathbf{e}^\top \mathbf{x}. \quad (12)$$

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

$$(\mathbf{P}_0) \quad \min_{\mathbf{x},\mathbf{r},\mathbf{w},\mathbf{f}} \mathbf{e}^\top \mathbf{x}, \quad (13)$$

s.t.

$$(\mathbf{R}_0) \begin{cases} (\mathcal{C}_1)\ T_i \le t^r_i, \forall i \in \mathbb{N}, \\ (\mathcal{C}_2)\ \sum_{i=1}^N r^f_{ij} \le R^f_j, \forall j \in \mathbb{M}^*, \\ (\mathcal{C}_3)\ \sum_{i=1}^N r^u_{ij} \le R^u_j, \forall j \in \mathbb{M}^*, \\ (\mathcal{C}_4)\ \sum_{i=1}^N r^d_{ij} \le R^d_j, \forall j \in \mathbb{M}^*, \\ (\mathcal{C}_6)\ \sum_{i=1}^N w^c_{ij} \le W^c_j, \forall j \in \mathbb{M}^*, \\ (\mathcal{C}_7)\ \sum_{i=1}^N f^c_{ij} \le F^c_j, \forall j \in \mathbb{M}^*, \\ r^u_{ij}, r^d_{ij}, r^f_{ij}, w^c_{ij}, f^c_{ij} \ge 0, \forall (i,j) \in \mathbb{N} \times \mathbb{M}^*, \\ f^c_{ij} \le f^{max}_j, \forall (i,j) \in \mathbb{N} \times \mathbb{M}^*, \end{cases} \quad (14)$$

and

$$(\mathbf{X}_0) \begin{cases} (\mathcal{C}_5)\ x^l_i + \sum_{j=1}^{M+1} x^f_{ij} + \sum_{j=1}^{M+1} x^c_{ij} = 1, \forall i \in \mathbb{N}, \\ x^l_i, x^f_{ij}, x^c_{ij} \in \{0,1\}, \forall (i,j) \in \mathbb{N} \times \mathbb{M}^*. \end{cases} \quad (15)$$

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, $(\mathcal{C}_5)$ is offloading decision constraints, $(\mathcal{C}_6)$ is the backhaul capacity constraints, and $(\mathcal{C}_7)$ is the constraints of CPU cycle capacity at the cloud. For the cloud of direct connections from mobile device, as the cloud can't offload tasks to a higher tier, we set $W^c_{(M+1)} = 0$ Mbps and $F^c_{(M+1)} = 0$ CPU cycles.

Among the three tiers, the local processing (at the mobile devices) does not incur transmission/reception and propagation delay, compared with offloading the task (to either the fog node tier or the cloud tier). Local processing also does not consume energy to offload the task. So, in terms of energy consumption and delay, local processing tends to be the best option. However, due to the limited computing capability of mobile devices (also to conserve their battery), more demanding tasks tend to be offloaded .

For the last two tiers, i.e., the fog nodes and the cloud, the latency and required energy to offload the tasks to the second tier are generally less than those of the case that tasks are offloaded to the third/cloud tier. So, for tasks that can't be processed locally, they should be offloaded to and processed by the second tier (the fog nodes). For tasks that can't be locally processed and the fog nodes also can't process them (e.g., due to fog nodes' limited communications resources or computing capability), they will be offloaded to the cloud tier.

Remark: In the above, for simplicity, we adopt a linear relationship between the data size and the energy consumption. Note that our analysis below and proposed framework can be extended to a nonlinear relationship between the data size and the energy consumption. This is because the energy consumption calculated for different offloading options/decisions (local processing, fog processing, or cloud processing) in Eqs. (2), (4), (6), and (8) for a given data size is provided as an input (the vector $\mathbf{e}$) to the problem $\mathbf{P}_0$ above but not part of the optimizing variables, the vector $\mathbf{x}$.

III. Proposed Optimal Solutions

The optimization problem $(\mathbf{P}_0)$ is an NP-hard due to its mixed integer non-linear programming. We observe that by relaxing all 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 (referred to as the fully-relaxed problem) is a convex optimization problem [23]. The convexity of the fully-relaxed problem is maintained in partly-relaxed problems, that are obtained by fixing some binary variables (to be 0 or 1) and relaxing the remaining ones. Using this characteristic, in the sequel we introduce three effective approaches to address the problem $(\mathbf{P}_0)$.

A. Convexity of Relaxed Problems

The fully-relaxed problem is written as follows:

$$(\widetilde{\mathbf{P}}_0) \quad \min_{\mathbf{x},\mathbf{r},\mathbf{w},\mathbf{f}} \mathbf{e}^\top \mathbf{x}, \tag{16}$$

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

$$(\widetilde{\mathbf{X}}_0) \begin{cases} (\mathcal{C}_5) \;\; x_i^l + \sum_{j=1}^{M+1} x_{ij}^f + \sum_{j=1}^{M+1} x_{ij}^c = 1, \forall i \in \mathbb{N}, \\ x_i^l, x_{ij}^f, x_{ij}^c \in [0,1], \forall (i,j) \in \mathbb{N} \times \mathbb{M}^*. \end{cases} \tag{17}$$

We first will prove that the fully-relaxed problem $(\widetilde{\mathbf{P}}_0)$ is a convex optimization problem.

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

Proof: The detailed proof is presented in Appendix A.

The below (suboptimal) solution to $(\mathbf{P}_0)$ obtained through solving $(\widetilde{\mathbf{P}}_0)$ is referred to as "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* [23], 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 $(\mathbf{P}_0)$. Due to the interdependence between the resource allocation and the offloading decision, the resource allocation solution of the relaxed problem has to be re-visited after rounding the integer decisions. Specifically, after fixing the offloading strategy with converted integer decisions, we again solve the resource allocation of problem $(\mathbf{P}_0)$ to find feasible solutions.

Although this approximation method (ROP) can quickly find a solution, the solution is suboptimal. In the following sections, we introduce two effective methods to find the optimal solution of $(\mathbf{P}_0)$.

B. Improved Branch and Bound Algorithm

The conventional BB works by searching through a tree, in which every node of it represents a subproblem after fixing a binary variable. The relaxed version of the subproblem at that node, which is equivalent to a partly-relaxed problem of the original one, can be solved to evaluate the potential of that node before branching and visiting the left or right children nodes. As such, in the standard BB, the size of subproblems at children nodes is 1 unit less than that of the father node in term of the number of variables. In other words, the complexity of the subproblems 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 leveraging the unique characteristics of its binary decision variables to reduce the complexity. The 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. 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 every node in the search tree of IBBA, we choose to branch the decisions of a task, forming a $(2(M+1)+1)$-tree with height N.
- **Simplifying problem** dictates that when a task is executed at the mobile device, a fog node, or the cloud server, 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, r_{ij}^f, w_{ij}^c, and f_{ij}^c in $(\mathbf{P}_0)$. Consequently, we have subproblems, namely $(\mathbf{RS})$, with a less number of variables.
- **Preserving convexity** dictates that the relaxed versions of subproblems $(\mathbf{RS})$ are convex optimization problems. In particular, based on Theorem 1, it can be observed that if we fix some binary variables in $(\mathbf{P}_0)$ and set all other binary variables to be real ones, the corresponding relaxed subproblems, also called partly-relaxed problems, are always convex.

The superiority of IBBA over the standard BB method in terms of complexity reduction is analyzed and presented in Section III-D2.

IBBA's Optimal Solution Selection: The joint offloading and resource allocation problem may have more than one optimal solutions in which the numbers of tasks processed at the mobile device, edge nodes and the cloud server are different. While some network operators prefer the optimal solutions with more tasks processed at fog nodes, the others may not. For example, if the cost of cloud's computation resource is lower than that of fog nodes', then the network operators may prefer offloading to the cloud server. Otherwise, the fog computing is more preferable since it can reduce the backhaul throughput between fog nodes and the cloud server. However, general solvers for MIP, e.g., CPLEX and MOSEK, do not allow us to select an optimal solution. In the IBBA, the selection of an optimal solution can be realized by leveraging the unique structure of the joint offloading and resource allocation problem.

Without loss of generality, we develop an optimal solution selection policy, namely LFC (L, F and C, respectively, stands for local, fog and cloud processing), so that the final solution is the optimal one with tasks preferably processed at mobile device, then at fog nodes, and at the cloud server. In other words, if a problem has more than one optimal solutions, then the solution with more tasks processed locally at the mobile devices will be chosen as the final solution. Otherwise, if these optimal solutions have the same number of tasks processed locally, then the optimal solution with more tasks processed at fog nodes will be chosen as the final solution (and so on). Recall that the IBBA

is a search tree-based algorithm, and the first optimal solution during the search is returned as the final solution. As mentioned above, **branching task** at a node will create a list of children nodes, each is equivalent to a fixed offloading decision of that computational task. Therefore, to enable the LFC policy, the list of children nodes will be visited in the order of offloading decisions, i.e., processing at mobile device, at fog nodes, and at cloud server.

Generally, we assume that computational tasks are chosen to branch in the order of $I_1, \ldots, I_N$, and the processors (i.e., mobile device, fog nodes and cloud server) are chosen to serve these tasks in the order of $L, F_1, \ldots, F_{(M+1)}, C_1, \ldots, C_{(M+1)}$. Here, I_i stands for task I_i, L is a mobile device, F_j is fog node j, and C_j is for the case in which fog node j forwards the computational task to the cloud server. The search tree in the IBBA method with this optimal solution selection policy, LFC, depicted in Fig. 2, is described as follows.

- The *deep-first search* (DFS) algorithm is used to travel the search tree of IBBA, and a *stack* is used to store the subproblems ($\mathbf{RS}$), which are generated during traveling the tree. Here, the stack is a popular data structure for adding and removing subproblems at one end called top of the stack as in Fig. 2(b).
- Tasks are chosen to branch in the order of $I_1, \ldots, I_N$. Tasks are called *active* tasks if their offloading decisions have not been fixed on the search tree.
- When branching a task, i.e., deciding where the task to be processed, the current subproblem on the top of the stack is deleted and the next subproblems ($\mathbf{RS}$) are generated and pushed in the stack in the order of processors $C_{(M+1)}, \ldots, C_1, F_{(M+1)}, \ldots, F_1, L$.

In Fig. 2(a), the DFS search algorithm scans the tree from left to right, and thus the most left branch optimal solution will be found first. Fig. 2(b) shows the status of the stack, which is a data structure to store the subproblems, before and after processing subproblem 1L. Here, the subproblem 1L on the top of the stack is deleted and the consequent subproblems are generated and pushed in the stack. We can see that the stack is suitable to the DFS search algorithm since the most left subproblem is always on the top of the stack. Consequently, the final solution is the optimal one with much tasks processed at mobile device, then at fog nodes, and at the cloud server.

In practice, we can sort tasks using different priority orders (e.g., their application types, or tasks' resource demand). For example, the higher-demand tasks are chosen to branch before the lower-demand tasks. Besides, each task can define its own order of processors. Thus, when branching a task at a node, the order of subproblems being pushed into the stack can be different according to its specific order of processors.

In the IBBA, **Branching task** defines an efficient partition of the search space, and **Simplifying problem** only eliminates the cases which can not lead to a optimal solution. Note that the optimal solution selection schemes (i.e., LFC and LCF) do not reduce the search space. Thus, the IBBA's optimality is preserved, i.e., the same as that under the standard BB method. The detail of IBBA is summarized in Algorithm 1.

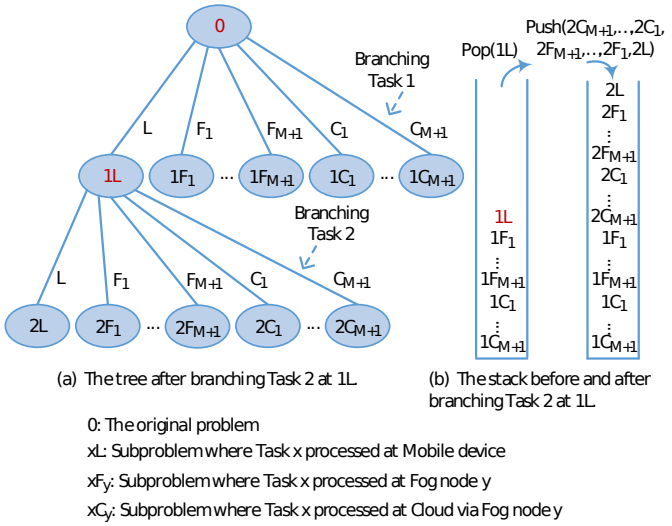


Fig. 2: Search Tree for IBBA with Optimal Solution Selection.

C. Feasibility-Finding Benders Decomposition

Although the IBBA has less and smaller size intermediate subproblems than the conventional BB algorithm, as analyzed in Section III-D2, the size of the intermediate subproblems reduces slowly. Moreover, the IBBA is not a distributed algorithm in essence. In this section, we design a distributed algorithm that decomposes the original problem into low-complexity subproblems that can be solved parallelly. Note that the dual decomposition method that is often instrumental to convex problem is not applicable to the underlying MINLP problem of which decisions for a task to be processed locally, at a fog, or at the cloud couple.

The Benders method [24] transforms the original problem into a master problem and subproblems for both integer and continuous variables. These two simpler problems are solved iteratively so that Benders cuts (also called Benders cutting-planes) can be applied to both the subproblems and the master problem. Note that the Benders decomposition method used in [25] is not efficient for large scale problems. First, in [25], only one Benders cut is created in each iteration and the master problem may have to try most of its solutions. Second, this linearization method with the dual multipliers faces the zig-zagging issue [26], [27] which increases the computation time.

To tackle these issues, we develop a distributed algorithm named Feasibility-Finding Benders decomposition (FFBD) as illustrated in Fig. 3. The key idea of FFBD is the generation of *Benders cuts* which exclude superfluous solutions, by assessing the communication and computation resources of fog nodes to satisfy tasks' requirements. Specifically, the Benders cuts are created simultaneously by solving the resource allocation problems at fog nodes. This completely differs from the approach presented in [25] in which only one Benders cut is created by solving the dual problem in each iteration.

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.

$$(\mathbf{MP}_0) \quad \mathbf{x}^{(k)} = \underset{\mathbf{x} \in \mathbf{X}_0}{\operatorname{argmin}} \{\mathbf{e}^\top \mathbf{x}\}, \tag{18}$$

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 $(M+1)$ nodes $\{Node_j(R_j^u, R_j^d, R_j^f)\}$
Fog nodes to cloud server $\{W_j^c,\ F_j^c\}$
Output: Optimal solution and value of problem $(\mathbf{P}_0)$

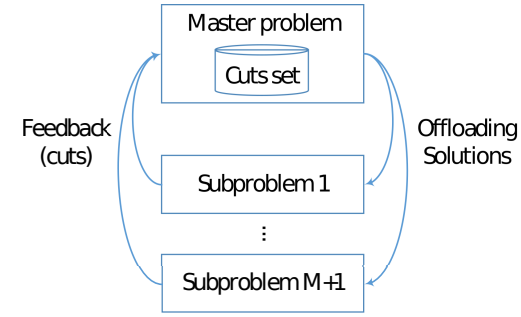
begin
  $s \leftarrow \emptyset$ ▷ Initialize empty solution
  $minE \leftarrow +\infty$ ▷ Initialize consumed energy $+\infty$
  $t.empty()$ ▷ Make stack empty
  $t.push((\mathbf{P}_0))$ ▷ Put $(\mathbf{P}_0)$ into stack
  while $t.isNotEmpty()$ **do**
    $p \leftarrow t.pop()$ ▷ Get subproblem from top of stack
    $subs, subminE \leftarrow$ Solve relaxed problem of p then return its optimal solution and value
    if $subminE > minE$ **or** p is infeasible **then**
      Prune p ▷ Delete subproblem p
    end
    if $subminE < minE$ **then**
      if $subs$ satisfies all integer constraints of $\{\mathbf{x}_i\}$ **then**
        $s \leftarrow subs$ ▷ Update solution
        $minE \leftarrow subminE$ ▷ Update optimal result
        Prune p ▷ Delete subproblem p
      end
      else
        $children \leftarrow$ Branch p by fixing the decisions of an *active* task in the order of $\{I_i\}$ based on **Branching task** property.
        Sort $children$ in the order of increasing prority of processors.
        for each $child$ **in** $children$ **do**
          Simplify $child$ based on **Simplifying problem** property.
          $t.push(child)$ ▷ Put subproblem into stack
        end
      end
    end
  end
  Return s and $minE$
end

s.t. $cuts^{(k)}$, and

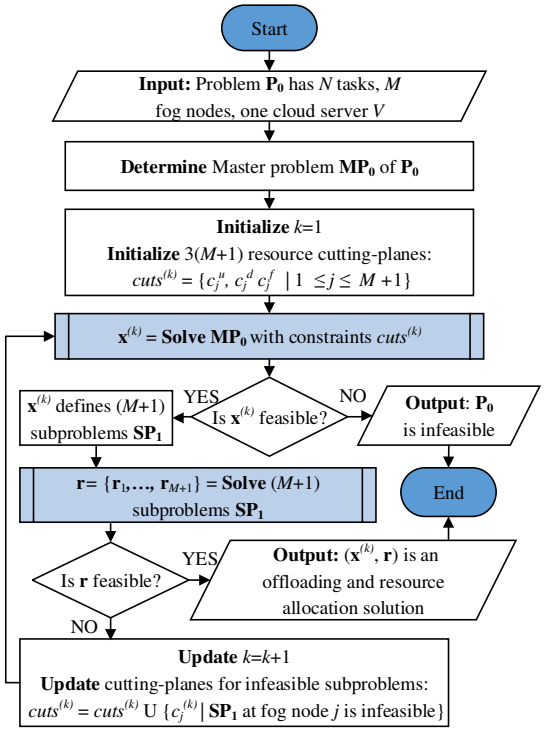
$$(\mathbf{SP}_0) \quad \min_{\mathbf{r},\mathbf{w},\mathbf{f}\in\mathbf{R}_0}\{0\}, \tag{19}$$

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

From Eqs. (18) and (19), 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 $(\mathbf{r}, \mathbf{w}, \mathbf{f})$. Theorem 2 shows that the iteration can stop when a feasible solution $(\mathbf{x}, \mathbf{r}, \mathbf{w}, \mathbf{f})$ is found or $(\mathbf{MP}_0)$ is infeasible. The convergence of the FFBD to an optimal solution in a finite number of iterations is analyzed in Section III-C3.



(a) FFBD Model.



(b) FFBD Procedure.

Fig. 3: Feasibility-Finding Benders Decomposition.

1) Distributed Subproblems: At iteration (k), by solving the integer programming problem $(\mathbf{MP}_0)$ the offloading decision variables $\mathbf{x}^{(k)}$ are found. The solution $\mathbf{x}^{(k)}$ determines if a task is processed at either mobile device, one of fog nodes, or the cloud server. Thus, $(\mathbf{SP}_0)$ is equivalently divided into a set of $(M+1)$ independent resource allocation subproblems (corresponding to $(M+1)$ nodes including the M fog nodes and the cloud V).

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 $\mathbf{x}_j^f = (x_{1j}^f, \ldots, x_{Nj}^f)$ and $\mathbf{x}_j^c = (x_{1j}^c, \ldots, x_{Nj}^c)$. Here, $\mathbf{x}_j$ is a part of $\mathbf{x} = \bigcup_{j=1}^{(M+1)} \mathbf{x}_j$.

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 respectively determined by two offloading decision variables $\mathbf{x}_j^{f(k)}$ and $\mathbf{x}_j^{c(k)}$ in $\mathbf{x}^{(k)}$ at iteration kth. We can write $\mathbb{N}_j^t = \{1, \ldots, t\}$, $\mathbb{N}_j^s = \{t+1, \ldots, t+s\}$, and $\mathbb{N}_j^{t+s} = \mathbb{N}_j^t \cup \mathbb{N}_j^s = \{1, \ldots, t+s\}$ is defined by $\mathbf{x}_j^{(k)} = (\mathbf{x}_j^{f(k)}, \mathbf{x}_j^{c(k)})$. Variables $\mathbf{r}_j = (\mathbf{r}_{1j}, \ldots, \mathbf{r}_{(t+s)j})$, $\mathbf{w}_j = (\mathbf{w}_{1j}^c, \ldots, \mathbf{w}_{(t+s)j}^c)$, and $\mathbf{f}_j = (\mathbf{f}_{1j}^c, \ldots, \mathbf{f}_{(t+s)j}^c)$, respec-

tively, are resources allocation and backhaul rate of fog node j, and the computation rate at the cloud towards its assigned set of tasks $\mathbb{N}_j^{t+s}$. Note that the 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 then written as

$$(\mathbf{SP}_1) \quad \min_{\mathbf{r}_j, \mathbf{w}_j, \mathbf{f}_j \in \mathbf{R}_j} \{0\}, \tag{20}$$

where

$$(\mathbf{R}_j) \begin{cases} (\mathcal{C}_{1j}) \;\; T_i \leq t_i^r, \forall i \in \mathbb{N}_j^{t+s}, \\ (\mathcal{C}_{2j}) \;\; \sum_{i \in \mathbb{N}_j^t} r_{ij}^f \leq R_j^f, \\ (\mathcal{C}_{3j}) \;\; \sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^u \leq R_j^u, \\ (\mathcal{C}_{4j}) \;\; \sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^d \leq R_j^d, \\ (\mathcal{C}_{6j}) \;\; \sum_{i \in \mathbb{N}_j^{t+s}} w_{ij}^c \leq W_j^c, \\ (\mathcal{C}_{7j}) \;\; \sum_{i \in \mathbb{N}_j^{t+s}} f_{ij}^c \leq F_j^d, \\ r_{ij}^f, r_{ij}^u, r_{ij}^d, w_{ij}^c, f_{ij}^c \geq 0, \forall i \in \mathbb{N}_j^{t+s}, \\ f_{ij}^c \leq f_j^{max}, \forall i \in \mathbb{N}_j^{t+s}, \\ r_{ij}^f = 0, \forall i \in \mathbb{N}_j^s, \\ w_{ij}^c = 0, f_{ij}^c = 0, \forall i \in \mathbb{N}_j^f. \end{cases} \tag{21}$$

As in Eq. (14), $(\mathcal{C}_{1j})$ is the delay requirement of tasks, $(\mathcal{C}_{2j}), (\mathcal{C}_{3j})$ and $(\mathcal{C}_{4j})$ are resource constraints at fog node j, $(\mathcal{C}_{6j})$ and $(\mathcal{C}_{7j})$ are the constraints of backhaul capacity at fog node j and CPU cycles at the cloud.

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 at all fog nodes. 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 $(\mathbf{r}, \mathbf{w}, \mathbf{f})$ of $(\mathbf{SP}_0)$. Then, $(\mathbf{x}, \mathbf{r}, \mathbf{w}, \mathbf{f})$ 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.

The solution of $(\mathbf{SP}_1)$ can be found by solving its equivalent problem $(\mathbf{SP}_2)$, which is always feasible, with additional slack variables $\mathbf{z}$ using any solver.

$$(\mathbf{SP}_2) \quad \min_{\mathbf{r}_j, \mathbf{w}_j, \mathbf{f}_j \in \mathbf{RZ}_j} (z_1 + z_2 + z_3 + z_4 + z_5), \tag{22}$$

where

$$(\mathbf{RZ}_j) \begin{cases} T_i \leq t_i^r, \forall i \in \mathbb{N}_j^{t+s}, \\ \sum_{i \in \mathbb{N}_j^t} r_{ij}^f - z_1 \leq R_j^f, \\ \sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^u - z_2 \leq R_j^u, \\ \sum_{i \in \mathbb{N}_j^{t+s}} r_{ij}^d - z_3 \leq R_j^d, \\ \sum_{i \in \mathbb{N}_j^{t+s}} w_{ij}^c - z_4 \leq W_j^c, \\ \sum_{i \in \mathbb{N}_j^{t+s}} f_{ij}^c - z_5 \leq F_j^c, \\ r_{ij}^f, r_{ij}^u, r_{ij}^d, w_{ij}^c, f_{ij}^c \geq 0, \forall i \in \mathbb{N}_j^{t+s}, \\ f_{ij}^c \leq f_j^{max}, \forall i \in \mathbb{N}_j^{t+s}, \\ r_{ij}^f = 0, \forall i \in \mathbb{N}_j^s, \\ w_{ij}^c = 0, f_{ij}^c = 0, \forall i \in \mathbb{N}_j^f, \\ z_1, z_2, z_3, z_4, z_5 \geq 0. \end{cases} \tag{23}$$

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 $\mathbf{x}^{(k)}$, $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_{(M+1)})$, $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_{(M+1)})$, and $\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_{(M+1)})$ 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 Section III-C3.

2) Fast Feasibility and Infeasibility Detection: Normally, the FFBD repeatedly solves $(\mathbf{MP}_0)$ 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 $(\mathbf{MP}_0)$ needs to be 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. The theoretical analysis below can be used to improve the efficiency of the FFBD algorithm.

From Eqs. (3), (5), and (7), the delay constraint $(\mathcal{C}_{1j})$ $T_i \leq t_i^r$ in $(\mathbf{R}_j)$ of $(\mathbf{SP}_1)$ can be rewritten as

$$\begin{cases} \left(\frac{D_i^i}{r_{ij}^u} + \frac{D_i^o}{r_{ij}^d} + \frac{C_i}{r_{ij}^f}\right) \leq t_i^r - \zeta, & \forall i \in \mathbb{N}_j^t \\ \left(\frac{D_i^i}{r_{ij}^u} + \frac{D_i^o}{r_{ij}^d}\right) + \left(\frac{(D_i^i + D_i^o)}{w_{ij}^c} + \frac{C_i}{f_{ij}^c}\right) \leq t_i^r - \zeta, & \forall i \in \mathbb{N}_j^s. \end{cases} \tag{24}$$

Remarkably, the component $(t_i^r - \zeta)$ is a constant. If $\exists i \in \mathbb{N}_j^{t+s}, t_i^r - \zeta \leq 0$, then processing task I_i at either fog node j or 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_i to the cloud server can be directly created for this case. Otherwise, if $t_i^r - \zeta > 0, \forall i \in \mathbb{N}_j^{t+s}$, then we define the relative size of task I_i by converting the delay requirement to 1 and adding the data size for backhaul link transmission as well as the number of CPU cycles that are required to process at the cloud. The relative size is defined as a 5-tuple $(D_i^{i'}, D_i^{o'}, C_i', D_i^{c'}, C_i^{c'})$, which $D_i^{i'}$, $D_i^{o'}$, respectively, are the uplink/downlink data lengths between the mobile device and fog node j, $C_i^{'}$ is the task's CPU cycles for execution at the fog node, $D_i^{c'}$ is the data length for the backhaul transmission, and $C_i^{c'}$ is the task's CPU cycles for execution at the cloud.

$$\begin{cases} \left(\frac{D_i^i}{t_i^r - \zeta}, \frac{D_i^o}{t_i^r - \zeta}, \frac{C_i}{t_i^r - \zeta}, 0, 0\right), & \forall i \in \mathbb{N}_j^t \\ \left(\frac{D_i^i}{t_i^r - \zeta}, \frac{D_i^o}{t_i^r - \zeta}, 0, \frac{(D_i^i + D_i^o)}{t_i^r - \zeta}, \frac{C_i}{t_i^r - \zeta}\right), & \forall i \in \mathbb{N}_j^s. \end{cases} \tag{25}$$

Let $\beta_i = \left(\frac{D_i^{i'}}{r_{ij}^u} + \frac{D_i^{o'}}{r_{ij}^d} + \frac{C_i'}{r_{ij}^f} + \frac{D_i^{c'}}{w_{ij}^c} + \frac{C_i^{c'}}{f_{ij}^c}\right)$ be the satisfaction rate of task I_i. The delay constraint in Eq. (24) becomes

$$\beta_i = \left(\frac{D_i^{i'}}{r_{ij}^u} + \frac{D_i^{o'}}{r_{ij}^d} + \frac{C_i'}{r_{ij}^f} + \frac{D_i^{c'}}{w_{ij}^c} + \frac{C_i^{c'}}{f_{ij}^c}\right) \leq 1, \forall i \in \mathbb{N}_j^{t+s}. \tag{26}$$

The Theorems 3 and 4 below can quickly check the feasibility and infeasibility of $(\mathbf{SP}_1)$.

THEOREM 3. *When fog node j and the cloud allocate all resources proportional to the input, output data sizes and CPU cycles of tasks, the equivalent delay components of these tasks are equal and defined as $\beta_{bal}^u = \frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{i'}}{R_j^u}$, $\beta_{bal}^d = \frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{o'}}{R_j^d}$, $\beta_{bal}^f = \frac{\sum_{i \in \mathbb{N}_j^{t+s}} C_i'}{R_j^f}$, $\beta_{bal}^w = \frac{\sum_{i \in \mathbb{N}_j^{t+s}} D_i^{c'}}{W_j^c}$, $\beta_{bal}^c = \frac{\sum_{i \in \mathbb{N}_j^{t+s}} C_i^{c'}}{F_j^c}$. If*

$\beta_{bal} = \beta^u_{bal} + \beta^d_{bal} + \beta^f_{bal} + \beta^w_{bal} + \beta^c_{bal} \leq 1$, *then the problem* $(\mathbf{SP}_1)$ *is feasible.*

Proof: The detailed proof is presented in Appendix D.

Corollary 3.1. *Let* $\beta^f_{bal} = \frac{\sum_{i\in\mathbb{N}^t_j} C_i^{'}}{R^f_j}$, $\beta^w_{bal} = \frac{\sum_{i\in\mathbb{N}^s_j} D_i^{c'}}{W^c_j}$, $\beta^c_{bal} = \frac{\sum_{i\in\mathbb{N}^s_j} C_i^{c'}}{F^c_j}$, *then calculate* $\gamma^u_{bal} = \frac{\sum_{i\in\mathbb{N}^t_j} D_i^{i'}/\left(1-\beta^f_{bal}\right)+\sum_{i\in\mathbb{N}^s_j} D_i^{i'}/(1-\beta^w_{bal}-\beta^c_{bal})}{R^u_j}$, *and* $\gamma^d_{bal} = \frac{\sum_{i\in\mathbb{N}^t_j} D_i^{o'}/\left(1-\beta^f_{bal}\right)+\sum_{i\in\mathbb{N}^s_j} D_i^{o'}/(1-\beta^w_{bal}-\beta^c_{bal})}{R^d_j}$.

If $\beta^f_{bal}, \beta^w_{bal}, \beta^c_{bal} \leq 1$ *and* $\gamma_{bal} = \gamma^u_{bal} + \beta^d_{bal} \leq 1$, *then the problem* $(\mathbf{SP}_1)$ *is feasible.*

The Corollary 3.1 is derived from Theorem 3. Fog node j first allocates the fog/cloud computation resources and the backhaul rate toward tasks in the set $\mathbb{N}^t_j$, which are processed at the fog node, and the tasks in the set $\mathbb{N}^s_j$, which are processed at the cloud, then allocates both the uplink and downlink resources toward all tasks in $\mathbb{N}^{t+s}_j$ based on their remaining delay requirements. Noticeably, tasks in $\mathbb{N}^{t+s}_j$ 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^f_{bal}, β^w_{bal}, β^c_{bal} 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* $\frac{\sum_{i\in\mathbb{N}^{t+s}_j} D_i^{i'}}{R^u_j} > 1$ *or* $\frac{\sum_{i\in\mathbb{N}^{t+s}_j} D_i^{o'}}{R^d_j} > 1$ *or* $\frac{\sum_{i\in\mathbb{N}^{t+s}_j} C_i^{'}}{R^f_j} > 1$ *or* $\frac{\sum_{i\in\mathbb{N}^{t+s}_j} D_i^{c'}}{W^c_j} > 1$ *or* $\frac{\sum_{i\in\mathbb{N}^{t+s}_j} C_i^{c'}}{F^c_j} > 1$, *then the problem* $(\mathbf{SP}_1)$ *is infeasible.*

Proof: The detailed proof is presented in Appendix E.

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

Corollary 3.1 is theoretically stronger than Theorem 3 because it repeatedly applies the theorem to the computation and backhaul resources then calculates the remaining 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: The cutting-planes based on Theorem 4 are useful for the large scale system (e.g., thousands of tasks and hundreds of fog nodes). For example, if a fog node can approximately support a maximum of n tasks, these cutting-planes can avoid the generation of subproblems with more than n assigned tasks.

3) 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". Although the FFBD can find the optimal solution only by using the subproblem cutting-planes (as analyzed below), by using the resource and prefixed decision cutting-planes, the master problem can avoid the offloading decisions that violate the resources and delay constraints. That helps reduce the search space.

Resource Cutting-Plane:

Recall that, $\mathbf{x}^f_j$ is the offloading decision variable vector that determines the subset of tasks $\mathbb{N}^t_j \subseteq \mathbb{N}$ being processed at fog node j, and $\mathbf{x}^c_j$ is the offloading decision variable vector that determines the subset of tasks $\mathbb{N}^s_j \subseteq \mathbb{N}$ being sent to the cloud by fog node j. Thus, vector $(\mathbf{x}^f_j, \mathbf{x}^c_j)$ determines the subset of tasks $\mathbb{N}^{t+s}_j \subseteq \mathbb{N}$.

Let $\mathbf{c}^{u(fog)}_j$, $\mathbf{c}^{d(fog)}_j$ and $\mathbf{c}^{f(fog)}_j$, respectively, be the coefficient vectors of $\mathbf{x}^f_j$ in the uplink, downlink and computation resource cutting-planes below. From Theorem 4, we have $\mathbf{c}^{u(fog)}_j = (D_1^{i'}, \ldots, D_N^{i'})/R^u_j$, $\mathbf{c}^{d(fog)}_j = (D_1^{o'}, \ldots, D_N^{o'})/R^d_j$ and $\mathbf{c}^{f(fog)}_j = (C_1^{'}, \ldots, C_N^{'})/R^f_j$. Here, $(D_i^{i'}, D_i^{o'}, C_i^{'})$ is calculated as in Eq. (25) for $i \in \mathbb{N}^t_j$.

Let $\mathbf{c}^{u(cloud)}_j$, $\mathbf{c}^{d(cloud)}_j$, $\mathbf{c}^{w(cloud)}_j$, and $\mathbf{c}^{f(cloud)}_j$, respectively, be the coefficient vectors of $\mathbf{x}^c_j$ in the uplink, downlink, backhaul link, and computation resource cutting-planes below. From Theorem 4, we have $\mathbf{c}^{u(cloud)}_j = (D_1^{i'}, \ldots, D_N^{i'})/R^u_j$, $\mathbf{c}^{d(cloud)}_j = (D_1^{o'}, \ldots, D_N^{o'})/R^d_j$, $\mathbf{c}^{w(cloud)}_j = (D_1^{c'}, \ldots, D_N^{c'})/W^c_j$, and $\mathbf{c}^{f(cloud)}_j = (C_1^{c'}, \ldots, C_N^{c'})/F^c_j$. Here, $(D_i^{i'}, D_i^{o'}, D_i^{c'}, C_i^{c'})$ is calculated as in Eq. (25) for $i \in \mathbb{N}^s_j$.

From Theorem 4, to avoid the generation of every subset $\mathbb{N}^{t+s}_j \subseteq \mathbb{N}$ that violates the uplink, downlink and computation resource constraints at edge node j, we add the following Benders cuts into $cuts$ set of the Master problem $(\mathbf{MP}_0)$:

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

The above resource cutting-planes are the linear functions of offloading decisions with non-zero coefficients as calculated in Eq. (25).

Subproblem Cutting-Plane: At iteration (k), fog node j is assigned a set of tasks $\mathbb{N}^{t+s}_j = \mathbb{N}^t_j \cup \mathbb{N}^s_j$, which is defined by offloading decision $\mathbf{x}^{(k)}_j = (\mathbf{x}^{f(k)}_j, \mathbf{x}^{c(k)}_j)$. 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 $\mathbb{N}_j \supseteq \mathbb{N}^{t+s}_j$ is infeasible. Thus, to eliminate all subproblems at edge node j containing $\mathbb{N}^{t+s}_j$, a new Benders cut $c^{(k)}_j$ is added into $cuts$ set of the Master problem $(\mathbf{MP}_0)$ after iteration (k):

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

Prefixed Decision Cutting-Plane: If task I_i satisfies $E^l_i < E^f_{ij}$ and $T^l_i \leq t^r_i$, then it can be pre-decided as local processing. As mentioned in *Fast Feasibility and Infeasibility Detection*, if $(t^r_i - \zeta) \leq 0$, then task I_i could not be offloaded. In these cases, the suitable cutting-planes can be created and added to set $cuts$ of $(\mathbf{MP}_0)$.

In each iteration of the FFBD, if a subproblem $(\mathbf{SP}_1)$ is infeasible then a *subproblem cutting-plane* is created. Each *subproblem cutting-plane* is equivalent to a set of computational tasks, which are either processed or forwarded to the cloud server for execution by fog node j. Besides, due to the finite numbers of tasks and fog nodes, the number of *subproblem cutting-planes* is finite. Consequently, the FFBD stops after has a finite number of iterations. Based on Theorem 2 we can conclude that the FFBD always returns the optimal solution after a limited number of iterations. This is equivalent to the conditions to converge to an optimal solution in the standard Benders decomposition [28]–[30].

Algorithm 2: FFBD Algorithm

Input : Set of tasks $\{I_i\left(D_i^i, D_i^o, C_i, t_i^r\right)\}$
Set of $(M+1)$ nodes $\{Node_j(R_j^u, R_j^d, R_j^f)\}$
Fog nodes to cloud server $\{W_j^c,\ F_j^c\}$
Output: Optimal solution $(\mathbf{x}, \mathbf{r}, \mathbf{w}, \mathbf{f})$ of Problem $(\mathbf{P}_0)$

begin
  Initialize k and $cuts^{(k)}$ as in **Initialization**.
  while solution $(\mathbf{x}, \mathbf{r})$ has not been found **do**
    $\mathbf{x} \leftarrow$ Solve $(\mathbf{MP}_0)$ with $cuts^{(k)}$ as in **Master Problem**. ▷ $\mathbf{x}$ store solution $\mathbf{x}^{(k)}$ at iteration k
    if $\mathbf{x}$ is feasible **then**
      Based on $\mathbf{x}$, create $(M+1)$ subproblems $(\mathbf{SP}_1)$ with asigned tasks $\mathbb{N}_1^{t+s}, \ldots, \mathbb{N}_{M+1}^{t+s}$.
    end
    else
      Return Problem $(\mathbf{P}_0)$ is infeasible.
    end
    for $(j=1;\ j \le (M+1);\ j=j+1)$ **do**
      $(\mathbf{r}_j, \mathbf{w}_j, \mathbf{f}_j) \leftarrow$ Solve $(\mathbf{SP}_1)$ at fog node j with task set $\mathbb{N}_j^{t+s}$ as in **Subproblems**.
      if $(\mathbf{r}_j, \mathbf{w}_j, \mathbf{f}_j)$ is infeasible **then**
        Add a new Benders cut $c_j^{(k)}$ into $cuts^{(k+1)}$ as in **Subproblems**.
      end
    end
    if $(\mathbf{r}, \mathbf{w}, \mathbf{f}) = ((\mathbf{r}_1 \cup \ldots \mathbf{r}_{(M+1)}), (\mathbf{w}_1 \cup \ldots \mathbf{w}_{(M+1)}), (\mathbf{f}_1 \cup \ldots \mathbf{f}_{(M+1)})$ is feasible **then**
      Solution $(\mathbf{x}, \mathbf{r}, \mathbf{w}, \mathbf{f})$ has been found.
    end
    $k \leftarrow (k+1)$ ▷ Increase iteration index
  end
  Return $\mathbf{x}$ and $(\mathbf{r}, \mathbf{w}, \mathbf{f})$
end

4) FFBD Procedure: The operation of the distributed FFBD algorithm is summarized in Fig. 3. At the iteration (k), the offloading decision solution $\mathbf{x}^{(k)}$ of $(\mathbf{MP}_0)$ determines where the tasks in $\mathbb{N}$ will be processed (i.e., the mobile device, fog nodes and the cloud server). Assume $\mathbb{N}_j^{s+t} \subseteq \mathbb{N}$ to be the set of tasks assigned to fog node j. Then every fog node j independently solves its own resource allocation problem of the form $(\mathbf{SP}_1)$. The Feasibility-Finding Benders decomposition, Algorithm 2, is described as below.

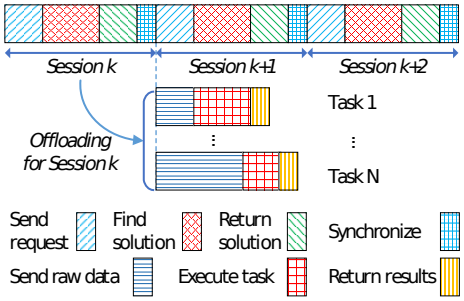


Fig. 4: Protocol defining the operation of proposed methods.

- **Initialization:** Set the iterator $k = 1$. Then, initialize $cuts^{(k)}$ in $(\mathbf{MP}_0)$ with $5(M+1)$ resource cutting-planes as in Section III-C3: $cuts^{(k)} = \bigcup_{j=1}^{M+1}\{c_j^u, c_j^d, c_j^f, c_j^{w(cloud)}, c_j^{f(cloud)}\}$. Other Benders cuts (i.e., prefixed decision cutting-planes) as in Section III-C3, 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 $(M+1)$ subproblems of the form $(\mathbf{SP}_1)$. If $(\mathbf{MP}_0)$ is infeasible, then the 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 $(\mathbf{r}, \mathbf{w}, \mathbf{f}) = ((\mathbf{r}_1 \cup \ldots \mathbf{r}_{(M+1)}), (\mathbf{w}_1 \cup \ldots \mathbf{w}_{(M+1)}), (\mathbf{f}_1 \cup \ldots \mathbf{f}_{(M+1)})$ of $(M+1)$ subproblems of the form $(\mathbf{SP}_1)$, then the FFBD is terminated, and $(\mathbf{x}^{(k)}, \mathbf{r}, \mathbf{w}, \mathbf{f})$ is the optimal feasible solution of the original problem $(\mathbf{P}_0)$.
- **Subproblems:** At iteration (k), a set of computational tasks $\mathbb{N}_j^{t+s} = \mathbb{N}_j^t \cup \mathbb{N}_j^s$ are assigned to fog node j, in which $\mathbb{N}_j^t$ and $\mathbb{N}_j^s$ are, respectively, processed and forwarded to the cloud for execution by fog node j. Then, fog node j independently solves $(\mathbf{SP}_1)$ in order to allocate resource to its own assigned tasks. Before calling a solver, Theorem 3 is used to check its feasibility. If $(\mathbf{SP}_1)$ is infeasible, then a new subproblem cutting-plane $c_j^{(k)}$ as in Section III-C3 is created and added into $cuts^{(k+1)}$ of $(\mathbf{MP}_0)$ for the next iteration $(k+1)$: $cuts^{(k+1)} = cuts^{(k)} \cup \{c_j^{(k)}\}$.

D. Implementation Protocol and Complexity Analysis

1) Implementation Protocol: We first introduce a method to identify the user devices, then propose a protocol that defines the operation of the ROP, IBBA and FFBD methods.

Device Identification: In order to cooperate horizontally and vertically, all devices, i.e., mobile devices, fog nodes and cloud server need to have a unique identification (ID), which can be determined by the MAC address or a temporary granted number. We assume that fog nodes and the cloud server have permanent IDs, e.g., MAC address. For the mobile devices without the permanent IDs, they can be periodically granted two-component temporary IDs of the form (FID_j, NID), in which FID_j is the permanent ID of fog node j, and NID is an integer number managed by fog node j. To make a unique temporary ID (FID_j, NID), fog node j will allocate different NID in each period. Besides, each mobile device can hold at most one temporary ID (FID_j, NID) provided by at most one fog node. Due to the cooperation among fog nodes and the cloud server, the mobile devices can be managed using these temporary IDs.

Proposed Protocol: The timeline of the fog computing system is divided into *session* including a fixed number of adjacent time slots. A *sessions*, then, includes four stages, namely **Send request**, **Find solution**, **Return solution** and **Synchronize** as in Fig. 4. Here, we assume that the communication and computation resources are managed so that the operation of the protocol in the current session (i.e., *session k*) and the execution of the tasks from previous sessions (i.e., *session k-1, k-2*) can occur simultaneously. The following are the details of these stages.

- **Send request:** At the begin of *session k*, mobile devices send offloading requests containing a unique identification (FID_j, NID, tk), where *tk* is a number to identify the task, and the task's information.
- **Find solution:** At this stage, a fog node or the cloud server with powerful computation capacities will run either the ROP, IBBA or FFBD. The FFBD method can also be run in a parallel or distributed cooperative manner. Noticeably, the tasks of *session k* will be offloaded then executed from the beginning of *session k+1*, thus all fog nodes need to estimate their available resources from *session k+1* to find the solution in session k.
- **Return solution:** The joint offloading and resource allocation solution found in the **Find solution** stage will be sent to mobile devices via fog nodes.
- **Synchronize:** This gap period is for synchronizing the fog computing system, preparing the offloading requests and estimating the available resources for the next session.

Since the timeline is divided into adjacent sessions as presented in Fig. 4, the offloading, executing and returning the results of tasks for *session k* can spread over the next sessions, e.g., *sessions k+1, k+2*.

2) Complexity Analysis: In this section, we evaluate the complexity of the proposed methods w.r.t. the numbers of tasks and fog nodes.

Original Problem: With $(M+1)$ nodes (including M fog nodes and the cloud V), the original problem $(\mathbf{P}_0)$ has $5(M+1)$ resource constraints as $(\mathcal{C}_2), (\mathcal{C}_3), (\mathcal{C}_4), (\mathcal{C}_6)$ and $(\mathcal{C}_7)$ described in Eq. (14). Besides, for each task I_i in $(\mathbf{P}_0)$, it needs $(2(M+1)+1)$ binary and $5(M+1)$ real variables, two delay and offloading constraints $(\mathcal{C}_1)$ as in Eq. (14) and $(\mathcal{C}_5)$ as in Eq. (15). Therefore, with N tasks and $(M+1)$ nodes, the original problem $(\mathbf{P}_0)$ has respectively $N(2(M+1)+1)$ integer and *5N(M+1)* real variables, and $(2N+5(M+1))$ constraints including N for the offloading decisions, N for the delay of the tasks and $5(M+1)$ for the resource requirements of the fog nodes and the cloud as described in Eq. (14) and Eq. (15). Consequently, the relaxed problem $(\widetilde{\mathbf{P}}_0)$ of $(\mathbf{P}_0)$ has totally $N(7(M+1)+1)$ real variables and $(2N+5(M+1))$ constraints.

Standard BB Method: For the standard BB method, it works as a binary search tree, in which every node on the tree represents a subproblem after fixing a binary offloading variable. Thus, in the worst case, the standard BB method has to solve $(2^1+2^2+\cdots+2^{N(2(M+1)+1)}) = (2^{N(2(M+1)+1)+1}-2)$ intermediate relaxed problems with the size decreasing from $(N(7(M+1)+1)-1)$ to $(5N(M+1))$ real variables. So its complexity is in the order of $O(2^{NM})$.

IBBA Method: For the IBBA method, it works as a $(2(M+1)+1)$-tree, in which every node on the tree represents a subproblem after deciding where a task is processed. Thus, in the worst case, the IBBA method has to solve $((2(M+1)+1)^1+(2(M+1)+1)^2+\cdots+(2(M+1)+1)^N) = \frac{((2(M+1)+1)^{N+1}-(2(M+1)+1))}{(2(M+1))} \approx (2(M+1)+1)^{N+1}/(2(M+1))$ intermediate relaxed problems with the size decreasing from $((N(7(M+1)+1)-(2(M+1)+1)-5(M))$ to $(5N)$ real variables. So its complexity is in the order of $O(M^N)$.

FFBD Method: For the FFBD method, the master problem $(\mathbf{MP}_0)$ and $M+1$ subproblems of the form $(\mathbf{SP}_1)$ are iteratively solved. At iteration k, $(\mathbf{MP}_0)$ is an integer problem with $N(2(M+1)+1)$ binary offloading variables and at most $(N+5(M+1)+k(M+1)$ constraints including N for the offloading decision constraints as $(\mathcal{C}_5)$ described in Eq. (15), $5(M+1)$ for resource cutting-planes as in the **Initialization** step, and at most $k(M+1)$ for the subproblem cutting-planes from solving $(M+1)$ subproblems k times as in **Subproblem** step. Besides, each subproblem $(\mathbf{SP}_1)$ is assigned an average of $N/(M+1)$ tasks. Thus, it has approximate $5N/(M+1)$ resources allocation variables and $(N/(M+1)+5)$ constraints including $N/(M+1)$ for the delay of $N/(M+1)$ tasks as $(\mathcal{C}_{1j})$ in Eq. (21) and 5 constraints for the resources requirements at the fog node and the cloud as $(\mathcal{C}_{2j})$, $(\mathcal{C}_{3j})$, $(\mathcal{C}_{4j})$, $(\mathcal{C}_{6j})$ and $(\mathcal{C}_{7j})$ described in Eq. (21). In the worst case, $(\mathbf{SP}_1)$ is assigned all N tasks, thus, has at most $5N$ resources allocation variables and $(N+5)$ 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. Additionally, considering all the combinations of N tasks for $(M+1)$ fog nodes, there are $(M+1)^{N+1}$ possible subproblems and $(M+1)^N$ master problems for the number of tasks per fog node varying from 0 to N. For that, the (worst case) complexity of the FFBD method is in the order of $O(M^N)$, the same as that of the IBBA method. However, under FFBD most of subproblems are eliminated by the cutting-planes generation, so the computing time in practice is much shorter than that of IBBA and BB (reducing up to 90% computation time as seen in the Section IV below). This is also thanks to the fact that the size of the MP and subproblems in the FFBD is a linear function of the number of tasks, whereas the size of the intermediate problems in the IBBA is exponential w.r.t. the number of tasks.

IV. Performance Evaluation

A. 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 being locally processed. Thus, for task I_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:

$$\alpha_i^* = \frac{e_{ij}^u D_i^i + e_{ij}^d D_i^o}{v_i D_i^i}. \tag{27}$$

Thus, task I_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.

B. Experiment Setup

We use the configuration of a Nokia N900 mobile device as in [31] 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 (Energy characteristics of local computing for Nokia N900/500 MHz in Table 1 in [31]). 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 the discrete uniform distribution between a and b. Here, we assume that 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. Specially, the maximum theoretically supported physical-layer data rate of 72 Mbps (the WiFi highest physical-layer data rate of 802.11n smartphones in [32], [33]) is used for both the uplink and downlink of each fog node and cloud server V. Here, the energy characteristics of a Nokia N900 with 3G near connection (Table 2 in [31]) are used to configure the direct connection between mobile devices and the cloud server V. Consequently, the energy consumption for transmitting and receiving a unit of data between mobile devices and the cloud server V are $e_{i(M+1)}^u = 0.658$ J/Mb and $e_{i(M+1)}^d = 0.278$ J/Mb, respectively. Besides, the energy consumption for transmitting and receiving a unit of data between mobile device i and fog node j ($j \leq M$) are lesser, $e_{ij}^u = 0.142$ J/Mb and $e_{ij}^d = 0.142$ J/Mb, respectively (the energy characteristics of a Nokia N900 with WLAN connection in Table 2 in [31]).

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 [34]. To evaluate the efficiency of theoretical proposals, each of these methods is implemented with two variants. Specifically, the IBBA variant with the optimal solution selection LFC policy is denoted as IBBA-LFC, whereas IBBA-LCF is the name of the IBBA variant with the LCF policy. In the IBBA-LFC, tasks are branched in the order of mobile device $\rightarrow$ fog nodes $\rightarrow$ cloud server. In the IBBA-LCF, the tasks are branched in the order of mobile device $\rightarrow$ cloud server $\rightarrow$ fog nodes. In other words, the IBBA-LFC tries to offload tasks to fog nodes as much as possible before offloading to the cloud server. On the contrary, the IBBA-LCF tries to offload to the cloud server before considering fog nodes.

TABLE I: Experimental parameters

Parameters	**Value**
Number of mobile devices N	10
Number of fog nodes M	4
Number of cloud server V	1
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 to fog nodes e_{ij}^u $(\forall j \leq M)$	0.142 J/Mb
Unit receiving energy consumption from fog nodes $e_{ij}^d (\forall j \leq M)$	0.142 J/Mb
Unit transmission energy consumption to cloud server V $e_{i(M+1)}^u$	0.658 J/Mb
Unit receiving energy consumption from cloud server V $e_{i(M+1)}^d$	0.278 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_j^u	72 Mbps
Downlink data rate of each fog node R_j^d	72 Mbps
CPU capacity of the cloud F_j^c (via fog node j)	40 Giga cycles/s
Maximum CPU rate of cloud to each task f_j^{max}	10 Giga cycles/s
Backhaul capacity between FNs and the cloud W_j^c	$[1, 10]$ Mbps
Multi-access delay ζ	20ms

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 the IBBA-LFC/LCF and FFBD-S/F will be compared with the 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 the average of these 10 runs. Noticeably, the 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.

C. Numerical Results

1) Scenario 1 - Varying 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. The backhaul capacity between FNs and the cloud is set as $W_j^c = 5$ Mbps. The cloud server can allocate a total of 40 Giga cycles/s to process tasks (each with maximum 10 Giga cycles/s) that are forwarded via each fog node. 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. 5 depicts the percentage of offloaded tasks and error rates, which are the proportion of tasks violating their delay requirements, when the task complexity α_i increases from $U(0.1, 1.0)$ to $U(1.0, 1.9)$. Generally, while the offloading trends of WOP

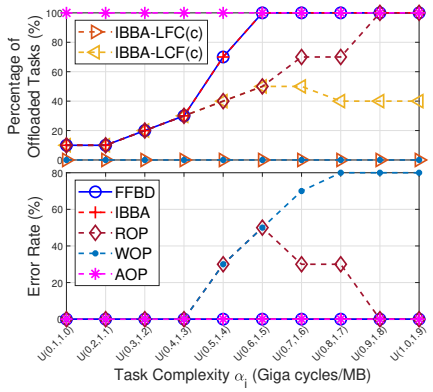


Fig. 5: Percentage of offloaded tasks and error rate as the task complexity α_i is increased.

and AOP are constants, i.e., 0% and 100%, respectively, the offloading trends of the FFBD, IBBA, and ROP methods increase dramatically from 10% to 100%. This is because there are more tasks benefit from offloading since their complexity α_i is greater than $\alpha_i^* = 0.911$ Giga cycles/MB according to Eq. (27). Besides, the percentage of tasks processed at the cloud server (i.e., labeled *IBBA-LFC(c)*) for the IBBA-LFC method is 0% due to the fog processing priority and the sufficient resources at fog nodes. However, only a proportion of offloaded tasks (e.g., 40% in 70% offloaded tasks at $\alpha_i = U(0.5, 1.4)$) are processed at the cloud server (i.e., labelled *IBBA-LCF(c)*) for the IBBA-LCF method when $\alpha_i \geq U(0.5, 1.4)$. This is because the cloud processing does not satisfy the delay requirements of all offloaded tasks. From Fig. 5, the zero error rates indicate the reliability of the FFBD and IBBA methods since their offloading and resource allocation solutions satisfy tasks' delay requirements, $t_i^r = 10s$. Besides, since $\alpha_i \geq U(0.5, 1.4)$, the local execution does not satisfy the delay requirements of all tasks. Thus, the WOP method records an increasing 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 30% offloaded task difference at $\alpha_i = U(0.5, 1.4)$).

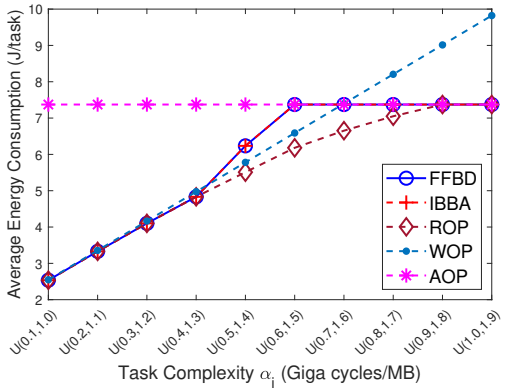
Fig. 6(a) and 6(b), respectively, show the average consumed energy of mobile devices and delay for the proposed methods when α_i increases from $U(0.1, 1.0)$ to $U(1.0, 1.9)$ Giga cycles/MB. Generally, the FFBD and IBBA have the lowest energy consumption in comparing with other methods in all experiments which satisfy the delay requirements. Specifically, due to all offloading without considering energy benefit, the AOP records a constant energy consumption (i.e., 7.37J/task) which is not lower than that of the FFBD and IBBA methods. 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. 6(a) shows that the ROP and WOP can be more energy-efficient at some points in comparing with the FFBD and IBBA, but they must suffer from latency constraint errors as in Fig. 5. For example, 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 FFBD and IBBA methods is 7.4J/task. However, the ROP and WOP suffer the equivalent error rates of 50% and 50%. Noticeably, all tasks are processed locally in the WOP method, thus the consumed energy increases linearly according to the task complexity ratio. From Fig. 6(b), the average delays of the IBBA-LFC/LCF and FFBD-S/F are always lower than the threshold $t_i^r = 10$s, matching with their zero error rates. Additionally, the 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, the FFBD-F records the lowest average delay in comparing with the IBBA-LFC/LCF and FFBD-S. Moreover, the fluctuation of delay in the FFBD-F is caused by the different distributions of tasks among fog nodes. We also 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 - Varying 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.

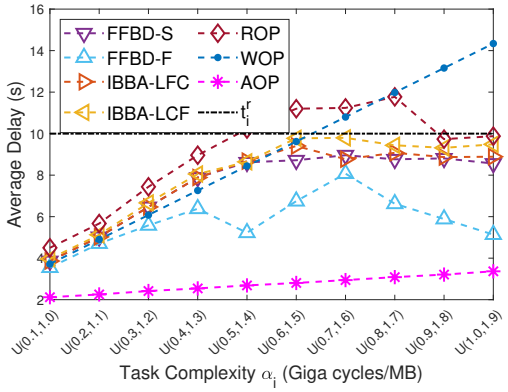
In this scenario, N tasks $I_i \left(D_i^i, D_i^o, C_i, t_i^r\right)$ are generated as $D_i^i \sim U(1.0, 10.0)$MB, $D_i^o \sim U(0.1, 1.0)$MB, $C_i \sim \alpha_i \times D_i^i$ Giga cycles, and t_i^r varying between $(2, 10)$s. Besides, we choose a wider complexity rate $\alpha_i = U(0.1, 6.0)$. The backhaul capacity between FNs and the cloud is set as $W_j^c = 5$ Mbps. 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.

Fig. 7(a) shows the offloading trends and error rates when the delay requirement goes up from 2s to 10s. Generally, while the trends of WOP and AOP are constants, i.e., 0% and 100%, respectively, the offloading trends of the FFBD and IBBA methods decrease from 90% to 50%. 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 system does not have enough resources to process all tasks satisfying the delay requirement $t_i^r \leq 3$, 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 the AOP. Since $t_i^r \geq 8$s, the FFBD and IBBA return the optimum solution with only 50% offloaded tasks, which get energy benefit from offloading. Besides, the proportion of tasks processed at the cloud server (i.e., labelled *IBBA-LFC(c)*) for the IBBA-LFC method is 0% due to the fog processing priority and the sufficient resources at fog nodes. However, in the IBBA-LCF method, the proportion of offloaded tasks being processed at the cloud server (i.e., labeled *IBBA-LCF(c)*) increases from 10% in 90% to 40% in 50% when the delay requirements t_i^r gradually grows from 3s to 10s. This is because the cloud computing can satisfy more tasks with the looser delay thresholds. From Fig. 7(a), generally, while the error rate of FFBD and IBBA is zero, it is generally decreases for other methods. The WOP records the highest error rate, steadily decreasing from 90% to 50% when the delay requirement is looser. The ROP records non-zero error rates when T_i^r is between 3s and 7s.

The offloading trends completely match with the average energy consumption depicted in Fig. 7(b). Generally, while it is a constant for both WOP with 8.5J/task and AOP with 7.4J/task,

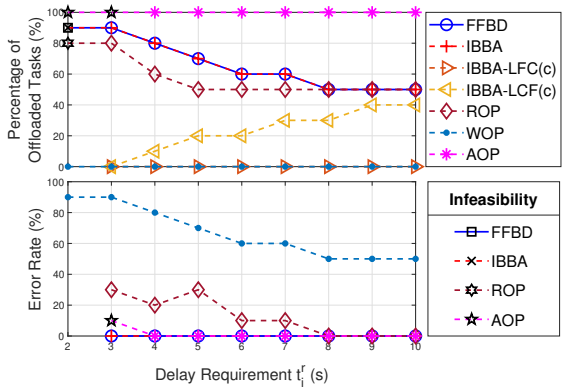


(a) Average consumed energy at mobile devices

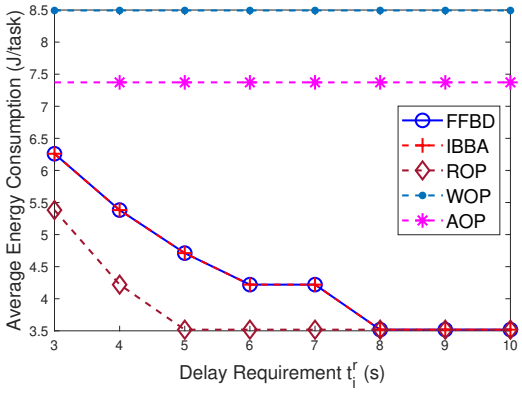


(b) Average task processing delay

Fig. 6: Consumed energy and task processing delay as the task complexity α_i is increased.



(a) Percentage of offloaded tasks and error rate



(b) Average consumed energy at mobile devices

Fig. 7: Percentage of offloaded tasks, error rate, and average consumed energy as the delay requirement t_i^r is less strict.

the consumed energy in the 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 FFBD and IBBA methods, although the ROP method achieves energy benefits at some experiments, it must suffer from latency errors.

3) Scenario 3 - Varying the Backhaul Capacity: To further illustrate the effect of the backhaul capacity on the percentage of offloaded tasks and the average delay, in this scenario, we set $\alpha_i = U(0.6, 1.5)$ Giga cycles/MB, then vary the backhaul capacity W_j^c from 1 Mbps to 10 Mbps. From Figure 8, we can see that the proportional of offloaded tasks does not change much for both FFBD and IBBA methods since the fog computing has enough resources to process all tasks while offloading directly to the cloud consumes more energy (due to the distance) and introduces longer delay. However, the IBBA-LCF method records the increasing percentage of tasks that are offloaded indirectly to the cloud server via fog nodes (denoted by IBBA-LCF(c)). Specifically, the proportion of tasks being processed at the cloud server increases from 10% to 80% in the IBBA-LCF method when the backhaul capacity increases from 1 Mbps to 10 Mbps. Due to the unchanged proportion of offloaded tasks, both FFBD and IBBA methods record the same energy consumption as the experiment $\alpha_i = U(0.6, 1.5)$ Giga cycles/MB in Scenario 1.

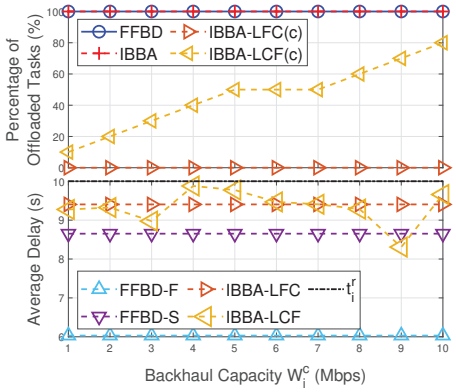


Fig. 8: Percentage of offloaded tasks and average delay as the backhaul capacity W_j^c increases.

4) 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, the WOP, AOP, and ROP are ignored due to their inadequacy of the goal.

We also evaluate the efficiency of integrating ROP into the FFBD method. Specifically, the methods, in which the offloading decision and resource allocation solutions of ROP are used as the initial point for FFBD-S/F, are named FFBD-S-ROP and

FFBD-F-ROP, respectively. Moreover, the solutions of the master problem and subproblems from previous iterations are also used as the initial points of the current iteration.

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

Fig. 9(a) and 9(b) 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. Noticeably, although the FFBD-F/S methods 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. This shows the efficiency of the decomposition, initial Benders cuts based on Theorem 4 and the cutting-plane generation from the results of subproblems in the FFBD. Besides, when either tasks have a higher complexity or a lower delay requirement, the FFBD-S/F and IBBA-LFC/LCF methods need to solve more intermediate problems in order to satisfy more resource demands of tasks, thus they require more time to find the optimal solutions. Especially, the FFBD-F with the fast solution detection method can reduce, respectively, 60%, 90% and 94% of the average computation time in comparing with the FFBD-S, IBBA-LFC and IBBA-LCF methods for Scenario 1, and 40%, 78% and 89% for Scenario 2. Fig. 9 also shows that integrating the approximated solution of ROP and intermediate solutions of iterations into FFBD does not reduce the number of problems, but it improves the solving time. Specifically, the solving time of FFBD-S-ROP and FFBD-F-ROP are, respectively, lower than that of FFBD-S and FFBD-F in all experiments.

Table II summarizes the major performance involving the computation time, the standard solver and fast solution detection method usages, and the master problem iterations. For Scenario 1, the FFBD-F-ROP, FFBD-F, FFBD-S-ROP, FFBD-S, IBBA-LFC and IBBA-LCF algorithms, respectively, have the average solving time of 186ms, 224ms, 382ms, 561ms, 2252ms, and 3714ms equivalent to an average of 3.5, 3.5, 9.4, 9.4, 17.9, and 29 times using the standard solver for the subproblems, which are either the intermediate relaxed problems in the IBBA-LFC/LCF or the subproblems of the form $(\mathbf{SP}_1)$ solved by the standard solver in the FFBD-S/F. For Scenario 2, the maximum number of master problem iterations is 20 for the FFBD-F/S, and average 8.63 (56.1%) of 15.38 subproblems are solved by the fast feasible method for the FFBD-F. From Fig. 9(a), 9(b) and Table II, 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 the joint offloading decision and resource allocation optimization framework for the multi-layer cooperative fog computing network. To find the optimal solution, we have developed three effective methods called IBBA with two variants IBBA-LFC/LCF (based on the Branch and bound), 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 relaxed problem). While the IBBA-LFC/LCF and FFBD-S/F can find the optimal solution, the ROP is a suboptimal method with error rates. The FFBD-F implemented the fast feasible detection method is the fasted algorithm in term of the computation time. Whereas, the IBBA-LFC/LCF algorithms with the optimal solution selection strategies can find the optimal solution with most tasks being offloaded to fog nodes and the cloud server, respectively. Numerical results have demonstrated the efficiency in terms of energy consumption reduction of the proposed methods.

Appendix A
Proof of Theorem 1

Proof. From Eqs. (2), (4), (6), (8), and (12), the objective function, $E = \mathbf{e}^\top \mathbf{x}$, is a linear expression of decision variables $\mathbf{x}$ because $\mathbf{e}$ is independent from $\mathbf{x}$, $\mathbf{r}$, $\mathbf{w}$, and $\mathbf{f}$. We need to show that all constraints in $(\mathbf{R}_0)$ and $(\widetilde{\mathbf{X}}_0)$ are convex functions. That is, from Eqs. (1), (3), (5), (7), and (10), the delay $T_i = \mathbf{h}_i^\top \mathbf{y}_i$ is the sum of functions, i.e., x_i^{l2}, $\frac{x_{ij}^{f2}}{r_{ij}^u}$, $\frac{x_{ij}^{f2}}{r_{ij}^d}$, $\frac{x_{ij}^{f2}}{r_{ij}^f}$, x_{ij}^{c2}, $\frac{x_{ij}^{c2}}{r_{ij}^u}$, $\frac{x_{ij}^{c2}}{r_{ij}^d}$, $\frac{x_{ij}^{c2}}{w_{ij}^c}$, and $\frac{x_{ij}^{c2}}{f_{ij}^c}$ $\forall j \in \mathbb{M}^*$, with positive coefficients, e.g., C_i, D_i^i, D_i^o, and $\left(D_i^i + D_i^o\right)$. Obviously, functions of the form x^2 are convex. We need to prove functions of the form $g(x,r) = \frac{x^2}{r}$ are convex. Let $\mathbf{H} = \nabla^2 g(x,r)$ is the Hessian of $g(x,r)$. Then, given an arbitrary vector $\mathbf{v} = (v_1, v_2)$, we have:

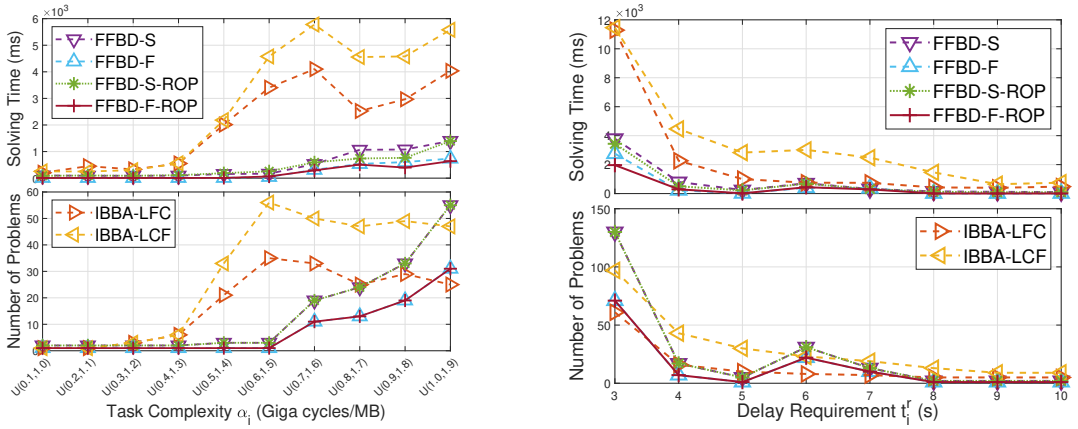
$$\mathbf{v}^\top \mathbf{H} \mathbf{v} = \mathbf{v}^\top \begin{bmatrix} \frac{\partial^2 g}{\partial^2 x} & \frac{\partial^2 g}{\partial x \partial r} \\ \frac{\partial^2 g}{\partial r \partial x} & \frac{\partial^2 g}{\partial^2 r} \end{bmatrix} \mathbf{v} = \frac{2}{r}\left(v_1 - v_2 \frac{x}{r}\right)^2. \quad (28)$$

Since the resource allocation variables $r_{ij}^u, r_{ij}^d, r_{ij}^f, w_{ij}^c, f_{ij}^c \geq 0$ (The equality occurs only when $x_{ij}^f, x_{ij}^c = 0$), we have $r \geq 0$. Consequently, we have $\mathbf{v}^\top \mathbf{H} \mathbf{v} \geq 0$. This implies that $\mathbf{H}$ is a positive semidefinite matrix, and thus $g(x,r)$ is a convex function w.r.t. (x,r) [23]. Thus, T_i is a convex function since it is the nonnegative weighted sum of convex functions. In other words, the constraint $(\mathcal{C}_1)$ in $(\mathbf{R}_0)$ is a convex function w.r.t. $\mathbf{x}$, $\mathbf{r}$, $\mathbf{w}$, and $\mathbf{f}$. Besides, the constraints $(\mathcal{C}_2)$, $(\mathcal{C}_3)$, $(\mathcal{C}_4)$, $(\mathcal{C}_5)$, $(\mathcal{C}_6)$, and $(\mathcal{C}_7)$ in $(\mathbf{R}_0)$ and $(\widetilde{\mathbf{X}}_0)$ are linear functions.

Since the objective function in Eq. (12) is a linear function, and all constraints in $(\mathbf{R}_0)$ and $(\widetilde{\mathbf{X}}_0)$ are convex functions, the relaxed problem $(\widetilde{\mathbf{P}}_0)$ is a convex optimization problem [23]. ■

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 $\min_{\mathbf{x} \in \mathbf{X}_0} \{\mathbf{c}^\top \mathbf{x}\}$ s.t. $cuts^{(k)} \leq \min_{\mathbf{x} \in \mathbf{X}_0} \{\mathbf{c}^\top \mathbf{x}\}$ s.t. $cuts^{(k+1)}$. In other words, $\min_{\mathbf{x} \in \mathbf{X}_0} \{\mathbf{c}^\top \mathbf{x}\}$ s.t. $cuts^{(k)}$ is a function that does not decrease with iteration k. Therefore, the first found feasible solution $(\mathbf{x}, \mathbf{r}, \mathbf{w}, \mathbf{f})$ of $(\mathbf{MP}_0)$ and $(\mathbf{SP}_0)$ is the optimal solution of $(\mathbf{P}_0)$.



(a) When task complexity α_i is increased

(b) When delay requirement t_i^r is less strict.

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

TABLE II: Complexity and computation times

Scenario 1: Increasing complexity						
	FFBD-F-ROP	**FFBD-F**	**FFBD-S-ROP**	**FFBD-S**	**IBBA-LFC**	**IBBA-LCF**
Min. time	7ms	8ms	88ms	90ms	204ms	250ms
Max. time	640	735ms	1395	1400ms	4092ms	5780ms
Average time	194ms	228ms	433ms	477ms	2059ms	2862ms
Average num. of standard solve	3.5 (35%)	3.5 (35%)	10.0 (100%)	10.0 (100%)	17.9 (100%)	29.3 (100%)
Average num. of fast solve	6.5 (65%)	6.5 (65%)	NA (0%)	NA (0%)	NA (0%)	NA (0%)
Max. MP Iterations	16	16	16	16	NA	NA
Average MP Iterations	4.5	4.5	4.5	4.5	NA	NA
Scenario 2: Vary the task delay requirements						
	FFBD-F-ROP	**FFBD-F**	**FFBD-S-ROP**	**FFBD-S**	**IBBA-LFC**	**IBBA-LCF**
Min. time	6ms	7ms	87ms	100ms	398ms	646ms
Max. time	1966ms	2756ms	3439ms	3789ms	11286ms	11458ms
Average time	378ms	475ms	682ms	779ms	2169ms	3390ms
Average num. of standard solve	8 (42.1%)	8 (42.1%)	19 (100%)	19 (100%)	14.6 (100%)	30.4 (100%)
Average num. of fast solve	11 (57.9%)	11 (57.9%)	NA (0%)	NA (0%)	NA (0%)	NA (0%)
Max. MP Iterations	26	26	26	26	NA	NA
Average MP Iterations	6.25	6.25	6.25	6.25	NA	NA

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 3

Proof. Let's find a feasible solution of $(\mathbf{SP}_1)$. Task I_i will be allocated with resources r_{ij}^u, r_{ij}^d, r_{ij}^f, w_{ij}^c and f_{ij}^c as $r_{ij}^u = \frac{D_i^{i'}}{\beta_{bal}^u}$, $r_{ij}^d = \frac{D_i^{o'}}{\beta_{bal}^d}$, $r_{ij}^f = \frac{C_i'}{\beta_{bal}^f}$, $w_{ij}^c = \frac{D_i^{c'}}{\beta_{bal}^w}$, and $f_{ij}^c = \frac{C_i^{c'}}{\beta_{bal}^c}$. We have $\beta_i = \left(\frac{D_i^{i'}}{r_{ij}^u} + \frac{D_i^{o'}}{r_{ij}^d} + \frac{C_i'}{r_{ij}^f} + \frac{D_i^{c'}}{w_{ij}^c} + \frac{C_i^{c'}}{f_{ij}^c}\right) = \left(\beta_{bal}^u + \beta_{bal}^d + \beta_{bal}^f + \beta_{bal}^w + \beta_{bal}^c\right)$. Here, $r_{ij}^f = 0$, $\frac{C_i'}{r_{ij}^f} = 0, \forall i \in \mathbb{N}_j^s$, and $w_{ij}^c = 0$, $f_{ij}^c = 0$, $\frac{D_i^{c'}}{w_{ij}^c} = 0$, $\frac{C_i^{o'}}{f_{ij}^c} = 0, \forall i \in \mathbb{N}_j^f$. Thus, $\beta_i = \beta_{bal} \leq 1, \forall i \in \mathbb{N}_j^{t+s}$.

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

APPENDIX E
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 $\max\limits_{i\in\mathbb{N}_j^{t+s}}\{\frac{D_i^{i'}}{r_{ij}^u}\} \geq \frac{\sum_{i\in\mathbb{N}_j^{t+s}} D_i^{i'}}{\sum_{i\in\mathbb{N}_j^{t+s}} r_{ij}^u}$. According to resource allocation conditions, $\sum_{i\in\mathbb{N}_j^{t+s}} r_{ij}^u \leq R_j^u$, we have $\max\limits_{i\in\mathbb{N}_j^{t+s}}\{\frac{D_i^{i'}}{r_{ij}^u}\} \geq \frac{\sum_{i\in\mathbb{N}_j^{t+s}} D_i^{i'}}{R_j^u}$. Therefore, $\max\limits_{i\in\mathbb{N}_j^{t+s}}\{\frac{D_i^{i'}}{r_{ij}^u}\} > 1$. Without loss of generality, we assume $\exists i* \in \mathbb{N}_j^{t+s}, \frac{D_{i*}^{i'}}{r_{i*j}^u} = \max\limits_{i\in\mathbb{N}_j^{t+s}}\{\frac{D_i^{i'}}{r_{ij}^u}\} > 1$. 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. (26). In conclusion, the problem $(\mathbf{SP}_1)$ is infeasible.

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

REFERENCES

[1] 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

[2] 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. 59–66.

[3] 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17302121

[4] 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.

[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] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, April 2010.

[8] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C. Lin, "Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2018.

[9] 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.

[10] 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.

[11] 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.

[12] J. Du, L. Zhao, X. Chu, F. R. Yu, J. Feng, and I. C, "Enabling low-latency applications in lte-a based mixed fog/cloud computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1757–1771, 2019.

[13] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for d2d-enabled mobile-edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4193–4207, 2019.

[14] C. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132–4150, 2019.

[15] 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.

[16] J. Wang, K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[17] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An admm framework," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2763–2776, 2019.

[18] Y. Huang, Y. Liu, and F. Chen, "Noma-aided mobile edge computing via user cooperation," *IEEE Transactions on Communications*, vol. 68, no. 4, pp. 2221–2235, 2020.

[19] 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.

[20] 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.

[21] C. Shuguang, A. J. Goldsmith, and A. Bahai, "Energy-efficiency of mimo and cooperative mimo techniques in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1089–1098, 2004.

[22] D. N. Nguyen and M. Krunz, "A cooperative clustering protocol for energy constrained networks," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2011, Conference Proceedings, pp. 574–582.

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

[24] A. M. Geoffrion, "Generalized benders decomposition," *Journal of Optimization Theory and Applications*, vol. 10, no. 4, pp. 237–260, 1972. [Online]. Available: https://doi.org/10.1007/BF00934810

[25] 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.

[26] 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.

[27] 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221716301126

[28] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962. [Online]. Available: https://doi.org/10.1007/BF01386316

[29] A. Grothey, S. Leyffer, and K. McKinnon, "A note on feasibility in benders decomposition," *Numerical Analysis Report NA/188, Dundee University*, 1999.

[30] Z. C. A Taşkin, *Benders Decomposition*, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470400531.eorms0104

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

[32] F. Liu, E. Bala, E. Erkip, M. C. Beluri, and R. Yang, "Small-cell traffic balancing over licensed and unlicensed bands," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5850–5865, 2015.

[33] S. K. Saha, P. Deshpande, P. P. Inamdar, R. K. Sheshadri, and D. Koutsonikolas, "Power-throughput tradeoffs of 802.11n/ac in smartphones," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, Conference Proceedings, pp. 100–108.

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

Thai T. VU received his B.S. and M.S. degrees in CS from VNU University of Engineering and Technology, Hanoi, Vietnam in 2006 and 2014, respectively. He is currently a Ph.D. student at University of Technology Sydney (UTS) as well as a research staff at the School of Engineering and Mathematical Sciences, La Trobe University, Australia. Before joining UTS, he was a lecturer at the Faculty of Computer Science and Engineering, Thuyloi University, Vietnam. His research interests include fog/cloud computing, Internet of Things, machine learning, and learning algorithms, with an emphasis on energy efficiency, low latency, fairness, and security/privacy awareness.

DIEP N. NGUYEN (Senior Member, IEEE) received the M.E. and PhD degrees in electrical and computer engineering from University of California San Diego (UCSD) and The University of Arizona (UA), respectively. He is a Faculty Member of the Faculty of Engineering and Information Technology, University of Technology Sydney (UTS). Before joining the UTS, he was a DECRA Research Fellow at Macquarie University, and a Member of Technical Staff at Broadcom, CA, USA, and ARCON Corporation, Boston, consulting the Federal Administration of Aviation on turning detection of UAVs and aircraft, U.S. Air Force Research Lab on anti-jamming. His current research interests include computer networking, wireless communications, and machine learning applications, with an emphasis on systems' performance and security/privacy. He has received several awards from LG Electronics, UCSD, UA, U.S. National Science Foundation, and Australian Research Council. He is an Editor, Associate Editor of the Transactions on Mobile Computing, IEEE Access, Sensors journal, and IEEE Open Journal of the Communications Society (OJ-COMS).

Dinh Thai Hoang (M'16) is currently a faculty member at the School of Electrical and Data Engineering, University of Technology Sydney, Australia. He received his Ph.D. in Computer Science and Engineering from the Nanyang Technological University, Singapore, in 2016. His research interests include emerging topics in wireless communications and networking such as machine learning, ambient backscatter communications, IRS, mobile edge intelligence, cybersecurity, IoT, and 5G/6G networks. He has received several awards including Australian Research Council. Currently, he is an Editor of IEEE Wireless Communications Letters and IEEE Transactions on Cognitive Communications and Networking.

Eryk Dutkiewicz received his B.E. degree in Electrical and Electronic Engineering from the University of Adelaide in 1988, his M.Sc. degree in Applied Mathematics from the University of Adelaide in 1992 and his PhD in Telecommunications from the University of Wollongong in 1996. His industry experience includes management of the Wireless Research Laboratory at Motorola in early 2000's. Prof. Dutkiewicz is currently the Head of School of Electrical and Data Engineering at the University of Technology Sydney, Australia. He is a Senior Member of IEEE. He also holds a professorial appointment at Hokkaido University in Japan. His current research interests cover 5G/6G and IoT networks.

THUY V. NGUYEN received the B.Sc. degree from the Hanoi University of Science and Technology (HUST), Hanoi, Vietnam, the M.Sc. degree from New Mexico State University, Las Cruces, NM, USA, and the Ph.D. degree from The University of Texas at Dallas, Richardson, TX, USA, all in electrical engineering. He was a member of Technical Staff with Flash Channel Architecture, Seagate, Fremont, CA, USA. He is currently a Lecturer with the Faculty of Information Technology, Posts and Telecommunications Institute of Technology (PTIT), Hanoi. His research interest includes wireless communications, coding theory, and machine learning applications in next-generation communication systems.