“©2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

Energy-based Proportional Fairness for Task Offloading and Resource Allocation in Edge Computing

Thai T. Vu[†‡], Dinh Thai Hoang[†], Khoa T. Phan[‡], Diep N. Nguyen[†], and Eryk Dutkiewicz[†]
[†]School of Electrical and Data Engineering, University of Technology Sydney, Australia
[‡]Department of Computer Science and Information Technology, La Trobe University, Melbourne Australia

***Abstract*—By executing offloaded tasks from mobile users, edge computing augments mobile devices with computing/communications resources from edge nodes (ENs), enabling new services/applications (e.g., real-time gaming, virtual/augmented reality). However, despite being more resourceful than mobile devices, allocating ENs' computing/communications resources to given favorable sets of users may block other devices from their service. This is often the case for most existing task offloading and resource allocation approaches that only aim to maximize the network social welfare (e.g., minimizing the total energy consumption) but not consider the computing/battery status of each mobile device. This work develops a proportional fair task offloading and resource allocation framework for a multi-layer cooperative edge computing network to serve all user equipment (UEs) while considering both their service requirements and individual energy/battery levels. The resulting optimization involves both binary (offloading decisions) and real variables (resource allocations), making it NP-hard. To tackle it, we leverage the fact that the relaxed problem is convex and propose a distributed algorithm, namely the dynamic branch-and-bound Benders decomposition (DBBD). DBBD decomposes the original problem into a master problem (MP) for the offloading decision and subproblems (SPs) for resource allocation. The SPs can either find their closed-form solutions or be solved in parallel at ENs, thus help reduce the complexity. The numerical results show that the DBBD returns the optimal solution of the problem maximizing the fairness between UEs. The DBBD has higher fairness indexes, i.e., Jain's index and min-max ratio, in comparing with the existing ones that minimize the total consumed energy.**

Keywords- Edge computing, offloading, resource allocation, fairness, MINLP, energy efficiency, branch-and-bound, Benders decomposition.

I. Introduction

Serving an ever-growing number of mobile user equipments (UEs) calls for an emerging network architecture, namely edge computing [1], [7]. In edge networks, edge nodes (ENs) (or computing nodes) are distributed closer to UEs to better serve high-demanding computing tasks, thus reducing the workload for backhaul links and enabling computing demanding and low-latency services/applications (e.g., real-time gaming, augmented/virtual reality) [2]. However, while cloud servers, e.g., Amazon Web Services, often possess huge computing resources, an EN can provide limited computation services toward users due to its limited computing resources [3]. As such, the collaboration among ENs as well as with cloud servers [4] is a very promising approach.

Moreover, offloading computing tasks from mobile devices to ENs is also not always an effective or even possible due to the energy consumption for two-way data transmissions between the UEs and the ENs [1] as well as tasks' security/QoS requirements. For that, the task offloading should be jointly optimized with the resource allocation. As aforementioned, despite of being more resourceful than mobile devices, allocating ENs' computing/communications resources to given favorable set of users may block one or other devices from their service. This is often the case for most existing task offloading and resource allocation approaches (as intensively surveyed in [9]) that only aim to maximize the network social welfare (e.g., optimizing the total consumed energy) but not consider the computing/battery status of each mobile device. Therefore, fairness should be considered along with efficiency in edge computing.

A few recent works considered the fairness in resource allocation and task offloading in edge networks. The min-max cost polices or max-min energy balance has been investigated in [5]. The work [6] accounts for the fairness amongst user vehicles and vehicle edge servers with a heuristic reward policy. Lately, a few works investigated the fairness [8], [10] using market equilibrium approaches. These papers relied on the game theory and market-based frameworks, which design the price for resources in a multiple edge node and budget-constrained buyer environment. However, the market-based framework is only applicable to the two-layer model only (i.e., UEs and edge node layers) [8], [10]. Second, these approaches can't capture the coupling among different types of resources (i.e., the task duration depends on the uplink, downlink, and computation resources).

Given the above, this work develops an energy-based proportional-fair framework to serve all user equipments (UEs) with multiple tasks while considering both their service requirements and individual energy/battery levels. Each UE, which may have multiple computing tasks, can connect to multiple nearby ENs to offload their tasks. The ENs can forward the tasks to a cloud server if they do not have sufficient resources to serve UEs. The edge computing and communications resources are to be jointly optimized with the task offloading decisions so as to fairly "share" the energy reduction/benefit to all UEs while taking into account the individual UEs' energy/battery levels. The energy/battery level at each UE is captured via a nonnegative weight factor. To the best of our knowledge, this is the first work in the literature to address the fairness of energy benefit among users in multi-layer edge computing system with multiple tasks.

The consequent problem for offloading tasks and allocating resources toward the tasks is a Mixed Integer Nonlinear Programming (MINLP), which is NP-hard [12]. To tackle it, we leverage the convexity of its relaxed problem to propose a distributed algorithm, namely the dynamic branch-and-bound Benders decomposition (DBBD). The DBBD decomposes the MINLP problem according to integer variables (offloading decisions) and real variables (resource allocations) into a master problem (MP) with integer variables and subproblems (SPs) with real variables at ENs. These sub-problems can be solved iteratively and parallelly at ENs. The theoretical proofs and the numerical results confirm that the DBBD can always

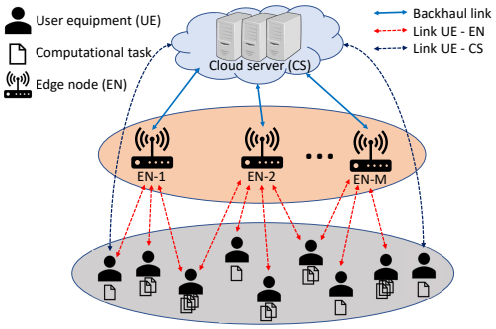

Fig. 1: Multi-layer edge computing system.

get the optimal solution maximizing the proportional fairness of the energy benefit among UEs, measured by Jain's index and the min-max ratio [13].

II. System Model and Problem Formulation

A. System Model

Fig. 1 show a three-layer edge computing system, including edge layer with M edge nodes (ENs) $\mathbb{M} = \{1, \dots, M\}$, cloud layer with one cloud server (CS), and user layer with N user equipments (UEs) $\mathbb{N} = \{1, \dots, N\}$. The two former layers can provide computational services toward the user layer. Let $\mathbb{S} = \{1, \dots, S\}$ be the security levels of UEs, ENs and the CS, in which 1 and S respectively denotes the highest and lowest levels [3]. Let $\mathbb{Q} = \{1, \dots, Q\}$ be the application types of computational tasks. The security requirement of application type q is defined by a mapping $\Theta(q) \in \mathbb{S}$. UEs have a set of independent computational tasks, denoted by $\Phi = \cup_{n=1}^{N} \Phi_n$, in which Φ_n is the set of tasks at the UE n. These tasks can be executed locally at UEs, offloaded to either ENs or the CS. Each task I_i owned by UE $n \in \mathbb{N}$ can be defined by $I_i \left(L_i^u, L_i^d, w_i, t_i^r, s_i^r, q, n\right)$, in which L_i^u and L_i^d respectively are the input and output data size (in Mb), w_i is the number of required CPU Giga cycles per input data unit [14]. Thus, $L_i^u w_i$ is the required CPU Giga cycles of task I_i. The QoS of task I_i comprises the requirements of delay t_i^r and security $s_i^r = \Theta(q) \in \mathbb{S}$. The tasks can be processed only by UE n, ENs, or the CS satisfying their QoS.

1) Local Processing: The UE n has a security level $s_n^l \in \mathbb{S}$ and a CPU processing rate f_n^l. If the security requirement of task I_i can be met, i.e., $s_n^l \leq s_i^r$, then task I_i can be processed locally at UE n. As in [9], the chip architecture of UE n can define the CPU power consumption rate as $P_n^l = \alpha (f_n^l)^\gamma$ with its specific parameters α and γ. The energy consumption E_i^l and the necessary computing time T_i^l of the UE are given by

$$E_i^l = P_n^l T_i^l = \alpha (f_n^l)^{\gamma - 1} (L_i^u w_i) \text{ and } T_i^l = (L_i^u w_i)/f_n^l. \quad (1)$$

2) Edge Node Processing: Edge node j has capabilities defined by $(R_j^u, R_j^d, R_j^f, s_j^f, \Psi_j)$, where R_j^u, R_j^d, R_j^f, $s_j^f \in \mathbb{S}$, and $\Psi_j \subseteq \mathbb{Q}$ respectively are the total uplink, total downlink, the CPU cycle, its security level, and the set of applications supported by the EN.

If task I_i is offloaded and processed at EN j, then this node will allocate resources for the UE n, defined by $\mathbf{r}_{ij} = (r_{ij}^u, r_{ij}^d, r_{ij}^f)$, in which r_{ij}^u, r_{ij}^d are uplink/downlink rates for transmitting the input/output, and r_{ij}^f is the computing resource for executing the task. The UE n will consume an amount of energy for transmitting input data to and receiving output data from the EN j. The latency of task I_i comprises the time for transmission input/output and the task-execution time at EN j.

Let e_{ij}^u and e_{ij}^d be the consumed energy rates of transmitting and receiving data. Let ζ be the delay caused by multi-access. The UE have the consumed energy E_{ij}^f and the delay T_{ij}^f are given by

$$E_{ij}^f = e_{ij}^u L_i^u + e_{ij}^d L_i^d, \quad (2)$$

$$T_{ij}^f = L_i^u / r_{ij}^u + L_i^d / r_{ij}^d + (L_i^u w_i)/r_{ij}^f + \zeta. \quad (3)$$

3) Cloud Server Processing: Let $\mathcal{B} = \{\mathcal{B}_1, \dots, \mathcal{B}_M\} \in \mathbb{R}^M$ be the backhaul capacity between M ENs and the CS. All tasks offloaded to the CS via EN j will share the backhaul $\mathcal{B}_j$. Let $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_Q\} \in \mathbb{R}^Q$ be the processing rate the CS can allocate to each task of Q applications. Let s_q^c be the security level of the CS toward application q. If the security requirement is satisfied, i.e., $s_i^r \geq s_q^c$, then EN j can forward task I_i to the CS.

In this case, the EN j will allocate resources $\mathbf{r}_{ij} = (r_{ij}^u, r_{ij}^d, r_{ij}^f)$ for the UE n, where r_{ij}^u, r_{ij}^d are uplink/downlink rates for transmitting input/output data, and $r_{ij}^f = 0$ (since it does not process the task). Then, the CS will allocate backhaul rate b_{ij} to transmitting input/output data between EN j and the CS. Task i will be processed at the CS with computation rate $\mathcal{C}_Q$. The energy consumption E_{ij}^c at the UE includes the energy for transmitting/receiving input/output to and from EN j. The delay T_{ij}^c comprises the time for transmitting the input from the UE to the CS via FN j, the time for receiving the output from the CS via the EN j, and the time for executing the task at the CS. These metrics are given by

$$E_{ij}^c = E_{ij}^f = E_{ij}^u + E_{ij}^d. \quad (4)$$

$$T_{ij}^c = L_i^u / r_{ij}^u + L_i^d / r_{ij}^d + (L_i^u + L_i^d)/b_{ij} + (L_i^u w_i)/\mathcal{C}_q + \zeta. \quad (5)$$

B. Problem Formulation

The offloading decisions of task I_i can be modelled as $\mathbf{x}_i = (x_i^l, x_{i1}^f \dots, x_{iM}^f, x_{i1}^c \dots, x_{iM}^c)$, where either $x_i^l = 1$ or $x_{ij}^f = 1$ or $x_{ij}^c = 1$ determine that task I_i is exclusively executed at either the UE or EN j or the CS (via EN j). Equivalently, we have the delay and energy consumption, i.e., $\mathbf{h}_i = (T_i^l, T_{i1}^f, \dots, T_{iM}^f, T_{i1}^c, \dots, T_{iM}^c)$ and $\mathbf{e}_i = (E_i^l, E_{i1}^f, \dots, E_{iM}^f, E_{i1}^c, \dots, E_{iM}^c)$, as in Eqs. (1)–(5).

Due to the energy, computing constraints or security requirements, not all tasks can be processed locally (e.g., local processing cannot satisfy the delay requirement). For that, we classify the set of tasks Φ into two categories: $\widehat{\Phi}$ for tasks that can either be executed locally or offloaded and $\widetilde{\Phi}$ (for tasks that are unable to be executed locally but always need to be offloaded).

Let E_i^{base} denote the total energy consumption required for the *baseline solution* to execute task I_i, depending on which category the task belongs to. Specifically:

$$E_i^{base} = \begin{cases} E_i^l, & I_i \in \widehat{\Phi}, \\ \max\limits_{j \leq M} \{E_{ij}^f, E_{ij}^c\} = \max\limits_{j \leq M} \{E_{ij}^f\}, & I_i \in \widetilde{\Phi}. \end{cases} \quad (6)$$

Then, the energy benefit/saving Δ_i of the UE and the task-execution delay T_i are given by

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

Finally, we can define the utility function of the UE n with its set of computational tasks Φ_n as follows.

$$u_n = \sum_{I_i \in \Phi_n} \Delta_i. \quad (8)$$

Without considering the fairness in energy reduction/benefit for users in the task offloading decision, one can simply optimize the total of all individual users' utility function u_i. Unlike these works in the literature, this work addresses a problem of jointly task-offloading ($\mathbf{x}$) and resource-allocating $(\mathbf{r}, \mathbf{b}) = (\{\mathbf{r}_{ij}\}, \{b_{ij}\})$ so that all UEs can achieve their proportionally fair share of energy benefit/saving, considering their delay, security, application compatibility requirements as well as their battery/energy status. Let $\rho_n \in [0, 1]$ with $\sum_1^N \rho_n = 1$ be the weight of the UE n that captures the user's battery status/priority level. According to the definition of proportional fairness in [15], the proportionally fair joint offloading and resource allocation solution can be obtained by maximizing of the utility function $\sum_{n=1}^{N} \rho_n \ln(u_n)$ over offloading decisions ($\mathbf{x}$) and resource allocation $(\mathbf{r}, \mathbf{b})$ toward all tasks in $\Phi = \widetilde{\Phi} \cup \widehat{\Phi}$. The equivalent optimization problem considering tasks' QoS requirements and edge nodes' resource constraints is formally formulated as follows.

$$(\mathbf{P}_0) \quad \max_{\mathbf{x},\mathbf{r},\mathbf{b}} \sum_{n=1}^{N} \rho_n \ln(u_n), \text{ s.t.} \tag{9}$$

$$(\mathbf{R}_0) \begin{cases} (\mathcal{C}_1)\ T_i \le t_i^r, \forall i \in \Phi, \\ (\mathcal{C}_2)\ \sum\limits_{i \in \Phi} r_{ij}^f \le R_j^f, \forall j \in \mathbb{M}, \\ (\mathcal{C}_3)\ \sum\limits_{i \in \Phi} r_{ij}^u \le R_j^u, \forall j \in \mathbb{M}, \\ (\mathcal{C}_4)\ \sum\limits_{i \in \Phi} r_{ij}^d \le R_j^d, \forall j \in \mathbb{M}, \\ (\mathcal{C}_5)\ \sum\limits_{i \in \Phi} b_{ij} \le \mathcal{B}_j, \forall j \in \mathbb{M}, \\ r_{ij}^u, r_{ij}^d, r_{ij}^f, b_{ij} \ge 0, \forall (i,j) \in \Phi \times \mathbb{M}, \end{cases} \tag{10}$$

and

$$(\mathbf{X}_0) \begin{cases} (\mathcal{C}_6)\ x_i^l + \sum\limits_{j=1}^{M} x_{ij}^f + \sum\limits_{j=1}^{M} x_{ij}^c = 1, \forall i \in \Phi, \\ (\mathcal{C}_7)\ x_i^l s_i^l + \sum\limits_{j=1}^{M} x_{ij}^f s_j^f + \sum\limits_{j=1}^{M} x_{ij}^c s_q^c \le s_i^r, \forall i \in \Phi, \\ (\mathcal{C}_8)\ x_{ij}^f = 0, \forall (i,j) \in \Phi \times \overline{G}(q), \\ x_i^l, x_{ij}^f, x_{ij}^c \in \{0,1\}, \forall (i,j) \in \Phi \times \mathbb{M}, \end{cases} \tag{11}$$

where $(\mathcal{C}_1)$, $(\mathcal{C}_7)$, and $(\mathcal{C}_8)$ capture tasks' QoS requirements, i.e., the delay, security, and application compatibility, $(\mathcal{C}_2)$, $(\mathcal{C}_3)$, $(\mathcal{C}_4)$, and $(\mathcal{C}_5)$ capture ENs' resource bounds, and $(\mathcal{C}_6)$ constrains tasks' offloading decisions. $\overline{\mathbb{G}}(q)$ is the set of all edge nodes that do not support the application type q.

III. Proposed Optimal Solutions

As aforementioned, the MINLP optimization problem $(\mathbf{P}_0)$ can be shown to be NP-hard (its proof is omitted due to space limitation). In general, it is intractable to find its optimal solution. However, by relaxing the integer variables to real numbers $x_i^l, x_{ij}^f, x_{ij}^c \in [0,1], \forall (i,j) \in \Phi \times \mathbb{M}$, the resulting relaxation of $(\mathbf{P}_0)$ becomes a convex optimization problem [12] (the proof is omitted here for brevity but can be found in [9]). In the sequel, we leverage this feature to develop an effective algorithm to find the optimal solution of $(\mathbf{P}_0)$.

A. Dynamic Branch-and-Bound Benders Decomposition

We introduce a dynamic Branch-and-Bound Benders Decomposition Algorithm, namely DBBD as in Fig. 2. In DBBD,

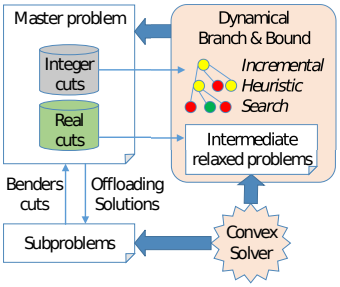

Fig. 2: Dynamic Branch-and-Bound Benders Decomposition.

$(\mathbf{P}_0)$ is first decomposed according to integer variables (offloading decisions) and real variables (resource allocations) into a master problem $(\mathbf{MP}_0)$ with the integer variables and subproblems $(\mathbf{SP}_0)$ with the real variables. Then, we develop a dynamical Branch and Bound (DBB), which is equipped an incremental heuristic search, to quickly find the optimal offloading solution of the $(\mathbf{MP}_0)$ [9][7]. The *Benders cuts* that eliminate useless solutions of the $(\mathbf{SP}_0)$ are generated and updated in the $(\mathbf{MP}_0)$. The DBBD finds the optimal solution of $(\mathbf{P}_0)$ by iteratively solving $(\mathbf{MP}_0)$ and $(\mathbf{SP}_0)$.

$$(\mathbf{MP}_0) \quad \mathbf{x}^{(k)} = \underset{\mathbf{x} \in \mathbf{X}_0}{\operatorname{argmax}} \{\sum_{n=1}^{N} \rho_n \ln(u_n)\} \text{ s.t. } cuts^{(k)}, \tag{12}$$

$$(\mathbf{SP}_0) \quad \min_{\mathbf{r},\mathbf{b} \in \mathbf{R}_0} \{0\}, \tag{13}$$

where $cuts^{(k)}$ is the set of Benders cuts generated in previous iterations $(1, \ldots, (k-1))$ as in Section III-C, $\{0\}$ is the constant zero. Here, $cuts^{(k)}$ are constraints on offloading variables $\mathbf{x}$ of $(\mathbf{MP}_0)$ at iteration (k).

In DBBD, at iteration (k), $(\mathbf{MP}_0)$ first is solved to find the offloading solution, i.e., $\mathbf{x}^{(k)}$, of $(\mathbf{MP}_0)$. Then, the $(\mathbf{SP}_0)$ is solved to find the resource allocation, i.e., $(\mathbf{r}, \mathbf{b})$, toward the offloaded tasks that are determined by $\mathbf{x}^{(k)}$ of $(\mathbf{MP}_0)$. According to Theorem 1, DBBD can terminate the iteration if either $(\mathbf{MP}_0)$ is infeasible or it returns a solution $(\mathbf{x}, \mathbf{r}, \mathbf{b})$.

Theorem 1. *At iteration (k), if a solution $(\mathbf{x})$ of $(\mathbf{MP}_0)$ leads to a solution $(\mathbf{r}, \mathbf{b})$ of $(\mathbf{SP}_0)$, then $(\mathbf{x}, \mathbf{r}, \mathbf{b})$ is the optimal one of $(\mathbf{P}_0)$. In addition, at iteration (k), if $(\mathbf{MP}_0)$ is infeasible, then $(\mathbf{P}_0)$ is infeasible.*

Proof: The detailed proof is presented in [9].

B. Distributed Solving Subproblems

The offloading decision $\mathbf{x}^{(k)}$ helps to break down $(\mathbf{SP}_0)$ to M smaller independent problems $(\mathbf{SP}_1)$ at M ENs. The resource allocation problem $(\mathbf{SP}_1)$ at EN j is for tasks that are offloaded to EN j (denoted Φ_j^t) and to the CS via EN j (denoted Φ_j^s). Equivalently, these sets are captured by $\mathbf{x}_j^{f(k)} = (x_{1j}^f, \ldots, x_{|\Phi|j}^f)^{(k)}$ and $\mathbf{x}_j^{c(k)} = (x_{1j}^c, \ldots, x_{|\Phi|j}^c)^{(k)}$ in $\mathbf{x}^{(k)}$. Thus, we can define $\Phi_j^t = \{1, \ldots t\}$, $\Phi_j^s = \{t+1, \ldots t+s\}$, and $\Phi_j^{t+s} = \Phi_j^t \cup \Phi_j^s = \{1, \ldots, t+s\}$ is captured by $\mathbf{x}_j^{(k)} = (\mathbf{x}_j^{f(k)}, \mathbf{x}_j^{c(k)})$. Variables $\mathbf{r}_j = (\mathbf{r}_{1j}, \ldots \mathbf{r}_{(t+s)j})$ and $\mathbf{b}_j = (\mathbf{b}_{1j}, \ldots \mathbf{b}_{(t+s)j})$ denote resource allocation of EN j towards the set of tasks Φ_j^{t+s}. The problem $(\mathbf{SP}_1)$ at EN j can be defined as

$$(\mathbf{SP}_1) \quad \min_{\mathbf{r}_j,\mathbf{b}_j \in \mathbf{R}_j} \{0\}, \tag{14}$$

in which $\mathbf{R}_j$ is the feasible region at EN j as follows.

$$(\mathbf{R}_j)\begin{cases} (\mathcal{C}_{1j})\ T_i \le t_i^r, \forall i \in \Phi_j^{t+s}, \\ (\mathcal{C}_{2j})\ \sum_{i\in\Phi_j^t} r_{ij}^f \le R_j^f, \\ (\mathcal{C}_{3j})\ \sum_{i\in\Phi_j^{t+s}} r_{ij}^u \le R_j^u, \\ (\mathcal{C}_{4j})\ \sum_{i\in\Phi_j^{t+s}} r_{ij}^d \le R_j^d, \\ (\mathcal{C}_{5j})\ \sum_{i\in\Phi_j^{t+s}} b_{ij} \le \mathcal{B}_j, \\ r_{ij}^f, r_{ij}^u, r_{ij}^d, b_{ij} \ge 0, \forall i \in \Phi_j^{t+s}, \\ r_{ij}^f = 0, \forall i \in \Phi_j^s,\ b_{ij} = 0, \forall i \in \Phi_j^t. \end{cases} \quad (15)$$

Instead of directly solving $(\mathbf{SP}_0)$, the resource allocation solution can be found by parallely solving M smaller problems $(\mathbf{SP}_1)$ at M ENs in concurrence with the CS for $(\mathbf{MP}_0)$.

At iteration (k), if M subproblems $(\mathbf{SP}_1)$ are feasible at all M ENs, then $\mathbf{x}^{(k)}$, $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_M)$, and $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_M)$ are the optimal solutions of $(\mathbf{P}_0)$. Otherwise, for each infeasible $(\mathbf{SP}_1)$ at EN j, a new Benders cut $c_j^{(k)}$ will be added to the cutting-plane set of $(\mathbf{MP}_0)$ for the next iteration, i.e., $cuts^{(k+1)} = cuts^{(k)} \cup c_j^{(j)}$, that are designed in Section III-C. To advance the efficiency of DBBD, we develops theoretical study in the section below.

1) Feasibility and Infeasibility Detection: Replacing Eqs. (3) and (5) into $(\mathcal{C}_{1j})$ $T_i \le t_i^r$ in $(\mathbf{R}_j)$ of $(\mathbf{SP}_1)$, this delay constraint can be transformed as

$$\begin{cases} \left(\frac{L_i^u}{r_{ij}^u} + \frac{L_i^d}{r_{ij}^d} + \frac{L_i^u w_i}{r_{ij}^f}\right) \le t_i^r - \zeta, & \forall i \in \Phi_j^t, \\ \left(\frac{L_i^u}{r_{ij}^u} + \frac{L_i^d}{r_{ij}^d}\right) + \left(\frac{L_i^u + L_i^d}{b_{ij}}\right) \le t_i^r - \frac{L_i^u w_i}{\mathcal{C}_q} - \zeta, & \forall i \in \Phi_j^s. \end{cases} \quad (16)$$

Remarkably, $(t_i^r - \zeta)$ and $\left(t_i^r - \frac{L_i^u w_i}{\mathcal{C}_q} - \zeta\right)$ are constant components. If $\exists i \in \Phi_j^{t+s}, (t_i^r - \zeta) \le 0$ or $\left(t_i^r - \frac{L_i^u w_i}{\mathcal{C}_q} - \zeta\right) \le 0$, then offloading task I_i to either EN j or the CS does not meet the delay requirement, i.e., $T_i \le t_i^r$, leading to the infeasibility of $(\mathbf{SP}_1)$. In this case, a new cutting-plane is directly generated to prevent offloading task I_i. In other , if $(t_i^r - \zeta) > 0, \forall i \in \Phi_j^t$ and $\left(t_i^r - \frac{L_i^u w_i}{\mathcal{C}_q} - \zeta\right) > 0, \forall i \in \Phi_j^s$, then the relative size, i.e., $(L_i^{u'}, L_i^{d'}, w_i', L_i^{c'})$, of task I_i is defined as

$$\begin{cases} \left(\frac{L_i^u}{t_i^r - \zeta}, \frac{L_i^d}{t_i^r - \zeta}, w_i, 0\right), & \forall i \in \Phi_j^t \\ \left(\frac{L_i^u}{t_i^r - \frac{L_i^u w_i}{\mathcal{C}_q} - \zeta}, \frac{L_i^d}{t_i^r - \frac{L_i^u w_i}{\mathcal{C}_q} - \zeta}, 0, \frac{L_i^u + L_i^d}{t_i^r - \frac{L_i^u w_i}{\mathcal{C}_q} - \zeta}\right), & \forall i \in \Phi_j^s. \end{cases} \quad (17)$$

For task I_i, let $\beta_i = \left(\frac{L_i^{u'}}{r_{ij}^u} + \frac{L_i^{d'}}{r_{ij}^d} + \frac{L_i^{u'} w_i'}{r_{ij}^f} + \frac{L_i^{c'}}{b_{ij}}\right)$. Then, the delay constraint in Eq. (16) becomes

$$\beta_i = \left(\frac{L_i^{u'}}{r_{ij}^u} + \frac{L_i^{d'}}{r_{ij}^d} + \frac{L_i^{u'} w_i'}{r_{ij}^f} + \frac{L_i^{c'}}{b_{ij}}\right) \le 1, \forall i \in \Phi_j^{t+s}. \quad (18)$$

Based on the relative size concepts, Theorems 2 and 3 below can detect the feasibility as well as the infeasibility of $(\mathbf{SP}_1)$.

THEOREM 2. *Let* $\beta_{bal}^u = \frac{\sum_{i\in\Phi_j^{t+s}} L_i^{u'}}{R_j^u}$, $\beta_{bal}^d = \frac{\sum_{i\in\Phi_j^{t+s}} L_i^{d'}}{R_j^d}$, $\beta_{bal}^f = \frac{\sum_{i\in\Phi_j^{t+s}} L_i^{u'} w_i'}{R_j^f}$, *and* $\beta_{bal}^b = \frac{\sum_{i\in\Phi_j^{t+s}} L_i^{c'}}{\mathcal{B}_j}$. *If* $\beta_{bal} = \beta_{bal}^u + \beta_{bal}^d + \beta_{bal}^f + \beta_{bal}^b \le 1$, *then* $(\mathbf{SP}_1)$ *is feasible and* $\mathbf{r}_{ij} = (r_{ij}^u, r_{ij}^d, r_{ij}^f, b_{ij}) = (\frac{L_i^{u'}}{\beta_{bal}^u}, \frac{L_i^{d'}}{\beta_{bal}^d}, \frac{L_i^{u'} w_i'}{\beta_{bal}^f}, \frac{L_i^{c'}}{\beta_{bal}^b}), \forall i \in \Phi_j^{t+s}$, *is a resource allocation solution.*

Proof: The detailed proof is presented in [9].

THEOREM 3. *If* $\frac{\sum_{i\in\Phi_j^{t+s}} L_i^{u'}}{R_j^u} > 1$ *or* $\frac{\sum_{i\in\Phi_j^{t+s}} L_i^{d'}}{R_j^d} > 1$ *or* $\frac{\sum_{i\in\Phi_j^{t+s}} L_i^{u'} w_i'}{R_j^f} > 1$ *or* $\frac{\sum_{i\in\Phi_j^{t+s}} L_i^{c'}}{\mathcal{B}_j} > 1$, *then* $(\mathbf{SP}_1)$ *is infeasible.*

Proof: The detailed proof is presented in [9].

C. Benders Cut Generation

This section develops three types of Benders cuts that will be added into the constraints of $(\mathbf{MP}_0)$.

1) Subproblem Benders Cut: At iteration (k), the problem $(\mathbf{SP}_1)$ at EN j is determined by $\mathbf{x}_j^{(k)} = (\mathbf{x}_j^{f(k)}, \mathbf{x}_j^{c(k)})$. If $(\mathbf{SP}_1)$ is infeasible, then a new Benders cut $c_j^{(k)}$ will be added into the $cuts$ set of $(\mathbf{MP}_0)$ to prevent offloading Φ_j^{t+s} in next iterations.

$$c_j^{(k)} = \{\mathbf{x}_j^{f(k)\top}\mathbf{x}_j^f + \mathbf{x}_j^{c(k)\top}\mathbf{x}_j^c \le t + s - 1\}. \quad (19)$$

2) Resource Benders Cut: To make a feasible problem $(\mathbf{SP}_1)$ at EN j, the set $\Phi_j^{t+s} \subseteq \Phi$ must not violate any resource constraints at EN j as presented in Theorem 3.

Let $\mathbf{c}_j^{u(edge)} = (L_1^{u'}, \ldots, L_N^{u'})/R_j^u$, $\mathbf{c}_j^{d(edge)} = (L_1^{d'}, \ldots, L_N^{d'})/R_j^d$ and $\mathbf{c}_j^{f(edge)} = (L_1^{u'} w_i', \ldots, L_N^{u'} w_i')/R_j^f$. Here, $(L_i^{u'}, L_i^{d'}, w_i')$ is defined as in Eq. (17) for $i \in \Phi_j^t$.

Let $\mathbf{c}_j^{u(cloud)} = (L_1^{u'}, \ldots, L_N^{u'})/R_j^u$, $\mathbf{c}_j^{d(cloud)} = (L_1^{d'}, \ldots, L_N^{d'})/R_j^d$, and $\mathbf{c}_j^{b(cloud)} = (L_1^{c'}, \ldots, L_N^{c'})/\mathcal{B}_j$. Here, $(L_i^{u'}, L_i^{d'}, L_i^{c'})$ is defined as in Eq. (17) for $i \in \Phi_j^s$.

To avoid the infeasible conditions in Theorem 3, we need to add the cutting-planes below into the $cuts$ set of $(\mathbf{MP}_0)$.

$c_j^u = \{\mathbf{c}_j^{u(edge)\top}\mathbf{x}_j^f + \mathbf{c}_j^{u(cloud)\top}\mathbf{x}_j^c \le 1\}$,
$c_j^d = \{\mathbf{c}_j^{d(edge)\top}\mathbf{x}_j^f + \mathbf{c}_j^{d(cloud)\top}\mathbf{x}_j^c \le 1\}$,
$c_j^f = \{\mathbf{c}_j^{f(edge)\top}\mathbf{x}_j^f \le 1\}$, and $c_j^b = \{\mathbf{c}_j^{b(cloud)\top}\mathbf{x}_j^c \le 1\}$.

3) Prefixed-Decision Benders Cut: As aforementioned in Section III-B1, if $(t_i^r - \zeta) \le 0$, then task I_i is unable to be offloaded to ENs, if $\left(t_i^r - \frac{L_i^u w_i}{\mathcal{C}_q} - \zeta\right) \le 0$, then task I_i is unable to be offloaded to the CS. Thus, the suitable cutting-planes can be generated and updated in the $cuts$ set of $(\mathbf{MP}_0)$.

D. DBBD Algorithm

DBBD finds the optimal solution by iteratively solving $(\mathbf{MP}_0)$ and $(\mathbf{SP}_1)$. An optimization solver can be used to find resource allocation solution of the convex problem $(\mathbf{SP}_1)$. However, we need to use a branch-and-bound method to solve the integer problem $(\mathbf{MP}_0)$. Thus, to support DBBD, we develop a dynamic branch and bound algorithm, namely DBB, which can effectively solve $(\mathbf{MP}_0)$ considering the balance between the users' demand and available resources at ENs. The details of the DBB is presented in [9]. The DBBD is presented in Algorithm 1. Initially, initialize the iterator $k = 1$ and set $cuts^{(k)}$ in $(\mathbf{MP}_0)$ with $4M$ resource Benders cuts as in Section III-C. At iteration (k), the DBB algorithm finds $\mathbf{x}^{(k)} \in X_0$ of $(\mathbf{MP}_0)$ satisfying $cuts^{(k)}$. With $\mathbf{x}^{(k)}$, M problems of the form $(\mathbf{SP}_1)$ at M ENs are defined. Then, EN j solves $(\mathbf{SP}_1)$ to find a resource allocation solution toward Φ_j^{t+s}. Here, Theorem 2 can determine the feasibility

of $(\mathbf{SP}_1)$. If $(\mathbf{SP}_1)$ has no solution, a new Benders cut $c_j^{(k)}$ as in Section III-C will be updated into $cuts^{(k+1)}$ of $(\mathbf{MP}_0)$ for the later iterations. If $\mathbf{x}^{(k)}$ of $(\mathbf{MP}_0)$ does not exist, then DBBD can conclude the infeasibility of $(\mathbf{P}_0)$. With $\mathbf{x}^{(k)}$ of $(\mathbf{MP}_0)$, if M problems of the form $(\mathbf{SP}_1)$ have solutions $(\mathbf{r}, \mathbf{b}) = (\{\mathbf{r}_j\}, \{\mathbf{b}_j\})$, then DBBD can conclude $(\mathbf{x}^{(k)}, \mathbf{r}, \mathbf{b})$ is the optimal solution of $(\mathbf{P}_0)$.

Algorithm 1: DBBD Algorithm

Input : Set Φ of tasks $I_i \left(L_i^u, L_i^d, w_i, t_i^r, s_i^r, q, n\right)$
Set of M edge nodes $\{(R_j^u, R_j^d, R_j^f, s_j^f, \Psi_j)\}$
Set $\mathbb{N}$ of UEs, Security levels $\mathbb{S}$
Application types $\mathbb{Q}$, Cloud server $(\mathcal{B}, \mathcal{C})$
Output: Optimal $(\mathbf{x}, \mathbf{r}, \mathbf{b})$ of $(\mathbf{P}_0)$

begin
  $k \leftarrow (k+1)$, $cuts^{(k)} \leftarrow \bigcup_{j=1}^{M} \{c_j^u, c_j^d, c_j^f, c_j^b\}$.
  while solution $(\mathbf{x}, \mathbf{r}, \mathbf{b})$ has not been found **do**
    $\mathbf{x} \leftarrow$ **DBB algorithm** solve $(\mathbf{MP}_0)$ with $cuts^{(k)}$. ▷ $\mathbf{x}$ store $\mathbf{x}^{(k)}$ at iteration k
    if $\mathbf{x}$ is found **then**
      Solution $\mathbf{x}$ defines M problems $(\mathbf{SP}_1)$ with asigned tasks $\Phi_1^{t+s}, \ldots, \Phi_M^{t+s}$.
    else **Return** Problem $(\mathbf{P}_0)$ is infeasible.
    for $(j = 1;\ j < M+1;\ j = j+1)$ **do**
      $(\mathbf{r}_j, \mathbf{b}_j) \leftarrow$ Solver solves $(\mathbf{SP}_1)$ at EN j with assigned tasks Φ_j^{t+s}.
      if $(\mathbf{r}_j, \mathbf{b}_j)$ is found **then**
        Update new cut $c_j^{(k)}$ into $cuts^{(k+1)}$.
    end
    if $(\mathbf{r}, \mathbf{b}) = (\{\mathbf{r}_j\}, \{\mathbf{b}_j\})$ is found **then**
      Optimal $(\mathbf{x}, \mathbf{r}, \mathbf{b})$ has been found.
    $k \leftarrow (k+1)$ ▷ For next iteration
  end
  Return $(\mathbf{x}, \mathbf{r}, \mathbf{b})$
end

E. Complexity Analysis

With $|\Phi|$ tasks and M ENs, there are $M^{|\Phi|+1}$ possible problems of the form $(\mathbf{SP}_1)$ with the task numbers increasing from 0 to $|\Phi|$ and $M^{|\Phi|}$ master problems $(\mathbf{MP}_0)$. Thus, in the worst situation, the DBBD algorithm has complexity in the order of $O(M^{|\Phi|})$. However, with the support of Benders cut generations, most of the useless subproblems are excluded. Consequently, in practice, the solving time is far more less than that of the worst case. This is also because $(\mathbf{MP}_0)$ and $(\mathbf{SP}_1)$ have the linear sizes over the number of tasks.

IV. Performance Evaluation

We study how the numbers of UEs/tasks affect the fairness, energy benefit, and total consumed energy of all UEs. To capture the fairness, we define and use the **Jain's index** $= \frac{\left(\sum_{n=1}^{N} u_n\right)^2}{N \sum_{n=1}^{N} u_n^2}$ and **Min-Max ratio** $= \frac{\min\limits_{n \le N}\{u_n\}}{\max\limits_{n \le N}\{u_n\}}$ where u_n is defined as in Eq. (8) [13].

At first, the experiments are designed with 5 applications $\mathcal{Q} = \{1, \ldots, 5\}$ and 3 security levels $\mathcal{S} = \{1-\text{High}, 2-\text{Medium}, 3-\text{Low}\}$. Each EN can randomly support 3 applications, and the CS can support all applications in $\mathcal{Q}$. All UEs have the same weight $\rho_n = 1$. We then generate $|\Phi| = 24$ tasks $I_i \left(L_i^u, L_i^d, w_i, t_i^r, s_i^r, q, n\right)$ $(I_i \in \Phi)$ in which $L_i^u = 1$ MB, $L_i^d = 0.1$ MB, $w_i = 5$ Giga cycles/Mb, $t_i^r = 5$s, $s_i^r \in \mathcal{S}$, and $q \in \mathcal{Q}$. Three edge nodes with WLAN connection are configured with total resources $(\sum R_j^u,\ \sum R_j^d,\ \sum R_j^f)$ = (108 Mbps, 108 Mbps, 15 Giga cycles/s) so that they can process $50\% \times 24 = 12$ tasks. Then, we vary the number of UEs N from 2 to 12 with the uplink/downlink energy consumption units raising by 0.01 J/Mb from 0.071 J/Mb to 0.213 J/Mb. Each UE has an equal demand with $\frac{24}{N}$ tasks. Other parameters are presented in Table I and clarified in [9].

TABLE I: Experimental parameters

Parameters	Value
Number of mobile devices N	2 – 12
Number of edge nodes M	3
Number of computation tasks $\lvert\Phi\rvert$	24
CPU rate of mobile devices f_i^l	1 Giga cycles/s
Security levels of mobile device s_i^l	1(High)
Energy consumption model of devices (α, γ)	$(10^{-11}\text{Watt/cycle}^2, 2)$
Unit transmitting energy consumption e_{ij}^u	0.071 – 0.213 J/Mb
Unit receiving energy consumption e_{ij}^d	0.071 – 0.213 J/Mb
Security level of each edge node $s_j^f \in \mathbb{S}$	1(High)
CPU rate of the cloud server $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_Q\}$	$\{10, \ldots, 10\}$ Giga cycles/s
Backhaul capacity between FNs and the cloud $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$	$\{100, \ldots, 100\}$ Mbps
Upper bound of backhaul rate for each task	$b_{ij} \le 5$ Mbps
Security level of clouds towards application q: $s_q^c \in \mathbb{S}$	1(High)
Multi-access delay ζ	20ms

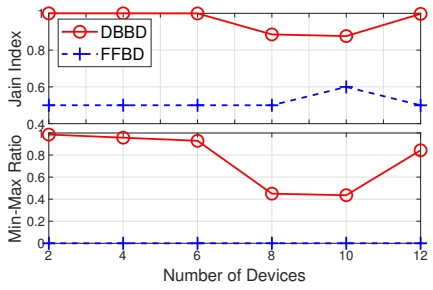

Fig. 3: Jain's index and Min-Max Ratio of energy benefits as the number of devices N is increased.

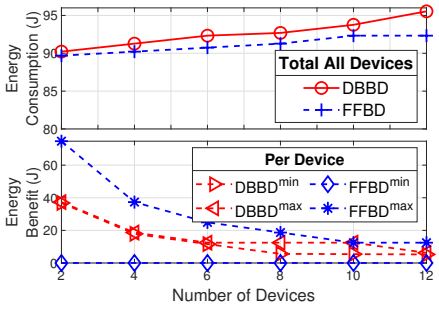

Fig. 4: Total consumed energy and energy benefits as the number of devices N is increased.

Our proposed framework, i.e., DBBD, is compared with the total utility maximization framework, e.g., FFBD [7] where the total energy consumption is minimized without considering the fairness. This policy is also called the social welfare maximization scheme (SWM) in [10]. The DBBD and FFBD are implemented using the **MOSEK** Optimizer [16]. In the DBBD and FFBD, the minimum and maximum energy benefit of UEs are denoted by $\text{DBBD}^{\min}$, $\text{DBBD}^{\max}$ and $\text{FFBD}^{\min}$, $\text{FFBD}^{\max}$, respectively.

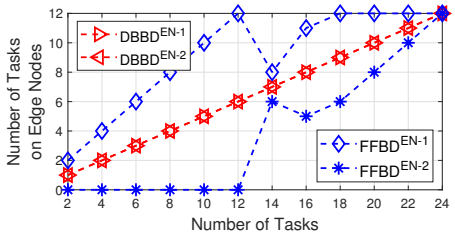

Fig. 5: Number of tasks offloaded to each edge node as the number of tasks is increased.

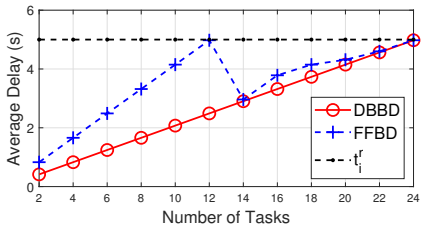

Fig. 6: Average delay of tasks as the number of tasks is increased.

Figs. 3 and 4, respectively, show the fairness indices and the energy benefit of UEs for the proposed methods when the number of UEs N is increased from 2 to 12. In Fig. 3, as expected, both the Jain's index and min-max ratio in the DBBD are much higher than those in the benchmark FFBD method. Especially, the both indices are close to their maximum value 1 for the cases of 2, 4, 6 and 12 UEs in the DBBD. This is because the DBBD aims to allocate the energy benefit in a proportionally fair manner to all UEs. Consequently, each UE can get its fair share of offloaded tasks, i.e., 6, 3, 2 and 1 (for 12 tasks described in the above).

The FFBD minimizes the total consumed energy, equivalently maximizing the total energy benefit of all UEs. Besides, the transceiving energy units of UE $n+1$ are 0.01 J/Mb higher than those of UE n as in the experiment setup. Consequently, in the FFBD, 12 tasks of UEs with the less energy consumption are offloaded, whereas 12 tasks of UEs with the higher energy consumption are processed locally. Thus, the Jain's index of the FFBD is mostly close to 0.5 and the min-max ratio is 0 for all experiments.

The fairness and efficiency of two schemes are also demonstrated in Fig. 4 in terms of the total energy consumption of all UEs and the energy benefit per UE. One can observe that the FFBD's consumption is a little lower than that of the DBBD. This is because the FFBD tries to minimize the total consumed energy, whereas the DBBD aims to maximize the proportionally fair energy benefit but the energy traded off for the fairness is not significant. The energy benefit of each UE also matches the trends of fairness indexes in both methods. Especially, the gap between the minimum ($\text{FFBD}^{\min}$) and maximum ($\text{FFBD}^{\max}$) energy benefits of the FFBD is lager than that of the DBBD ($\text{DBBD}^{\min}$ and $\text{DBBD}^{\max}$). The zero value of the minimum energy benefit ($\text{FFBD}^{\min}$) shows that all the tasks of some UEs are processed locally in the FFBD method.

To investigate the effect of the number of tasks on the number of tasks offloaded to each edge node and the average delay of all tasks, we set up two devices with an equal number of tasks $|\Phi|/2$. We then vary the total number of tasks $|\Phi|$ from 2 to 24. These devices have WLAN connections to ENs 1 and 2, and the 3G near connections to EN 3. From Fig. 5, the DBBD offloads tasks equally to ENs 1 and 2 (labelled $\text{DBBD}^{\text{EN-1}}$ and $\text{DBBD}^{\text{EN-2}}$). This is because while the DBBD always returns offloading decisions that balance the load amongst ENs, the FFBD returns arbitrary ones. Here, tasks are not offloaded to EN 3 due to the higher energy consumption of the 3G connection. As a result, the DBBD has a lower average delay than the FFBD has as in Fig. 6.

V. Conclusion

We considered the fairness among users in the joint task offloading and resource allocation problem for the multi-layer cooperative edge computing network. To that end, we formulated a proportional fairness maximization problem that turns out to be NP-hard. To find its optimal solution, we have developed a dynamic Branch-and-Bound Benders Decomposition algorithm, namely DBBD, to decompose the original problem into subproblems that can be solved parallelly in a distributed at edge nodes. Numerical results showed that the DBBD always returns the optimal solution, which achieves the Jain's index and Min-Max ratio, i.e., respectively around 1.0 and 0.8 in most cases. Simulations were also used to compare DBBD with other task offloading and resource allocation frameworks that only aim to minimize the energy consumption. Our proposed framework ensures that UEs' tasks and edges' resources can be offloaded and allocated with respect to UEs' energy/priority levels.

References

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

[2] Y. Mao, *et al.*, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[3] H. El-Sayed, *et al.*, "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.

[4] J. Wang, *et al.*, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Trans. Mobile Comput.*, pp. 1–1, 2019.

[5] J. Liu, *et al.*, 'Max-min energy balance in wireless-powered hierarchical fog-cloud computing networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7064–7080, 2020.

[6] H. Liao, *et al.*, 'Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing," *IEEE Trans. Wireless Commun.*, vol. 22, no. 7, pp. 4051–4063, 2021.

[7] T. T. Vu, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, and T. V. Nguyen, "Optimal energy efficiency with delay constraints for multi-layer cooperative fog computing networks," *IEEE Trans. Commun.*, vol. 69, no. 6, pp. 3911–3929, 2021.

[8] D. T. Nguyen, *et al.*, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Trans. Cloud Comput.*, vol. 9, no. 1, pp. 302–317, 2021.

[9] T. T. Vu, D. T. Hoang, Khoa T. Phan, D. N. Nguyen, and E. Dutkiewicz, "Proportional Fairness Edge Computing Resource Allocation using Dynamic Branch-and-Bound Benders Decomposition Algorithm," *Technical report*, 2021. [Online]. Available: https://arxiv.org

[10] D. T. Nguyen, *et al.*, "A market-based framework for multi-resource allocation in fog computing," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1151–1164, 2019.

[11] Y. Wang, *et al.*, "Cooperative task offloading in three-tier mobile computing networks: An admm framework," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2763–2776, 2019.

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

[13] R. K. Jain, *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984.

[14] J. Du, *et al.*, "Enabling low-latency applications in lte-a based mixed fog/cloud computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1757–1771, 2019.

[15] F. P. Kelly, *et al.*, "Rate control for communication networks: shadow prices, proportional fairness and stability," *JORS*, vol. 49, no. 3, pp. 237–252, 1998.

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