Joint Task Offloading and Resource Allocation in Heterogeneous Edge Environments

Yu Liu, *Graduate Student Member, IEEE*, Yingling Mao, *Graduate Student Member, IEEE*, Zhenhua Liu, Fan Ye, *Senior Member, IEEE*, and Yuanyuan Yang, *Fellow, IEEE*

***Abstract*—Mobile edge computing has emerged as a prevalent computing paradigm to support applications that demand low latency and high computational capacity. Hardware reconfigurable accelerators exhibit high energy efficiency and low latency compared to general-purpose servers, making them ideal for integration into mobile edge computing systems. This article investigates the problem of joint task offloading, access point selection, and resource allocation in heterogeneous edge environments for latency minimization. Given the heterogeneity of edge computing devices and the interdependence of the decisions required for offloading, access point selection, and resource allocation, it is challenging to optimize over them simultaneously. We decomposed the proposed problem into two disjoint subproblems and developed algorithms for each of them. The first subproblem is to jointly determine access point selection and communication resource allocation decisions, for which we have proposed an algorithm with a provable approximation ratio of $2.62/(1-8\lambda)$, where λ is a tunable parameter balancing the approximation ratio and time complexity. Additionally, we offer a faster variant of the algorithm with an approximation ratio of $(\sqrt{3}+1)^2$. The second subproblem is to determine offloading and computing resource allocation decisions jointly and is NP-hard, where we developed algorithms based on relaxation and rounding. We conducted comprehensive numerical simulations to evaluate the proposed algorithms, and the results demonstrated that our algorithms outperformed existing baselines and achieved near-optimal performance across various settings.**

***Index Terms*—Heterogeneous edge environments, latency minimization, mobile edge computing.**

I. Introduction

THE advent of information technology has brought forth numerous novel applications that demand ultra-low response times, including augmented reality, virtual reality, the Internet of Things, and autonomous vehicles [1], [2], [3]. However, wireless devices face several constraints, such as limited computing capability and high energy consumption, that prevent them from fulfilling the low latency requirements of the applications, such as tasks that might include deep neural network inference, complex data analytics, or real-time video processing. Meanwhile, cloud computing cannot provide low latency either, given its susceptibility to network congestion and the substantial physical distances involved. To address this issue, offloading tasks from wireless devices to edge servers is gaining traction as a prevailing computing paradigm for these applications, owing to the proximity of edge servers to end-users and their potent computing power.

The development of Field Programmable Gate Arrays (FPGAs) has led to the widespread adoption of FPGA-based reconfigurable accelerators for diverse tasks [4]. Utilizing FPGAs for specific tasks has proven highly efficient in terms of computing time and energy consumption. For instance, an AlexNet accelerator with a 16-bit fixed point implemented on Xilinx Virtex-7 outperforms the ARM Cortex A15 by being 62 times faster and consuming 22 times less energy [5], [6]. Typically, FPGAs are coupled with a CPU in conventional computing systems, serving as powerful auxiliary components. Recently, a more advanced architecture has been proposed, where FPGAs can be connected to networks as standalone computing resources, adhering to existing Infrastructure as a Service (IaaS) mechanisms [7], [8]. This approach allows for greater scalability, flexibility, and ease of maintenance.

Previous research on task offloading and resource management in edge computing has mainly targeted homogeneous systems equipped with either general-purpose processors or FPGAs. In contrast, our study delves into a heterogeneous edge computing system encompassing both FPGAs and general-purpose servers. Wireless devices (WDs) communicate with computing devices through access points (APs) such as base stations. Four categories of decisions exist in such systems: offloading, access point selection, computing resource management, and communication resource management, all of which jointly determine the system's overall latency. The objective is to minimize the average latency for all wireless devices. The objective of the system is to minimize the average latency for all wireless devices.

Although edge computing and the incorporation of standalone FPGAs offer numerous advantages, simultaneously selecting offloading, AP selection, computing resource management, and communication resource management decisions presents significant challenges. First, the system's computing devices are heterogeneous, making them suitable for distinct tasks. For

Manuscript received 12 April 2023; revised 25 October 2023; accepted 10 November 2023. Date of publication 28 November 2023; date of current version 7 May 2024. This work was supported in part by the National Science Foundation under Grants 1730291, 1717731, 2230620, 2046444, 2146909, 2106027, and 2214980. Recommended for acceptance by C. Assi. *(Corresponding author: Yu Liu.)*

Yu Liu, Yingling Mao, Fan Ye, and Yuanyuan Yang are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794 USA (e-mail: yu.liu.3@stonybrook.edu; yingling.mao@stonybrook.edu; fan.ye@stonybrook.edu; yuanyuan.yang@stonybrook.edu).

Zhenhua Liu is with the Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794 USA (e-mail: zhenhua.liu@stonybrook.edu).

Digital Object Identifier 10.1109/TMC.2023.3335198

instance, some tasks achieve significant acceleration on FPGAs, while others result in less pronounced latency improvements. Moreover, tasks from different WDs demand varying resource amounts when executed on FPGAs. As a result, offloading decisions must be judiciously determined to optimally utilize the finite FPGAs and delegate tasks to the most suitable computing devices. Second, the computing resources of general-purpose servers are finite and necessitate allocation among multiple WDs [9]. Resource allocation among WDs must be balanced to guarantee the minimum overall latency. Third, WDs can communicate with edge computing devices via different APs, where the channel conditions between the APs and WDs may differ. As a result, the AP selections for WDs need to be orchestrated to minimize their collective latencies. Lastly, communication bandwidth is limited and must be distributed among WDs in an optimally cooperative manner. Existing methods are inadequate in providing a comprehensive solution to these challenges, while this work investigates the joint task offloading, AP selection, and resource management problem in heterogeneous edge environments.

Our main contributions are summarized as follows.

- We formulate the Joint task Offloading, AP selection, and resource Management problem (JOAM) in heterogeneous edge environments, which we demonstrate to be NP-hard. The JOAM problem can be divided into two separate NP-hard problems, namely Joint AP selection and communication resource Management problem (JAM) for minimizing communication latency and Joint task Offloading and computing resource Management problem (JOM) for minimizing computing latency, (see Section III-E for details).
- For the first subproblem (JAM), we derive the closed-form optimal communication resource allocation decision and introduce a game-theoretic algorithm, denoted as Congestion Game-Based Algorithm (*CGBA*), for making AP selection decisions. The algorithm has an approximation ratio of $\frac{2.62}{(1-8\lambda)}$ and converges in at most $O(\frac{I}{\lambda}\log(\frac{P_0}{P^*}))$ iterations, where λ is an adjustable parameter that balances time complexity and approximation ratio. That is, the communication latency under the proposed *CGBA* algorithm, with parameter λ, is no more than $\frac{2.62}{(1-8\lambda)}$ times the optimal latency. We also propose a faster variant of *CGBA*, named *FCGBA*, which converges in I iterations and exhibits a relatively higher approximation ratio of $(\sqrt{3}+1)^2$.
- For the second subproblem (JOM) minimizing computing latency, we first derive the optimal computing resource allocation decision under any given offloading decision. In addition, we show that there is no polynomial-time approximation algorithm for choosing offloading decisions. Consequently, we develop a semidefinite relaxation-based algorithm to make offloading decisions. We also propose a faster quadratic relaxation-based offloading algorithm.
- We evaluate the algorithms by extensive simulations. The results highlight that the proposed algorithms outperform popular baselines and are near-optimal, with an average latency only 1.5% higher than the optimal value.

The remainder of this paper is structured as follows: Section II reviews related works; Section III presents the problem formulation; Sections IV and V propose algorithms for problems JAM and JOM, respectively; Section VI demonstrates the performance evaluation of the proposed algorithms; and Section VII concludes the paper.

II. Related Works

A large number of works of task offloading with various purposes and settings in edge computing systems are well-studied, e.g., [9], [10], [11], [12], [13], [14]. In [11], Xu et al. consider a problem of service caching and task offloading to minimize computational latency under a time-average energy consumption constraint, which does not account for communication latency. The authors propose a Gibbs sampling-based algorithm to make decisions. In [13], Sun et al. formulate and investigate a task offloading problem and define a metric called computation efficiency. However, the paper does not consider the processing latency of tasks. The authors propose an iterative algorithm based on gradient descent methods, which efficiently minimizes computation efficiency. In [14], Shuwaili et al. investigate the joint uplink and downlink optimization problem of minimizing energy consumption under latency constraints, which does not account for system heterogeneity, and the proposed problem is solved using successive convex approximation techniques. In [15], Liu et al. focus on the problem of cost-aware online task offloading and resource management. They use the drift-plus-penalty approach to balance the energy cost and the system latency. In addition, some papers focus on offloading partial tasks [16], [17], [18], [19]. In [16], Pavlos et al. investigate a risk-aware heterogeneous multi-MEC system and propose a game-theoretic-based algorithm with low time complexity. In [17] and [18], Pavlos et al. focus on the problem of risk-aware data offloading in multi-server multi-access and propose a non-cooperative game-theoretic-based distributed low-complexity algorithm. In [19], Pavlos et al. study the satisfaction-aware data offloading problem in surveillance systems. None of the studies above studies integrate hardware reconfigurable accelerators into their systems.

Some works are based on deep reinforcement learning. In [12], Liu et al. formulate and study the multi-objective problem of task offloading and resource scheduling and propose a double deep Q network-based approach converging quickly. In [2], Shi et al. consider a vehicle-to-vehicle (V2V) task offloading problem in blockchain-enabled vehicular edge computing systems, and a deep reinforcement learning-based algorithm is proposed. In [1], Huang et al. consider the computation offloading problem in wireless-powered mobile edge computing networks and propose a deep reinforcement learning method to make binary offloading decisions. In [20], Wang et al. study the joint edge trans-coding and client enhancement problem in multi-tier mobile edge computing networks with the goal of maximizing the quality of experience of mobile users and a deep reinforcement learning method is proposed to make computing decisions. In [21], Liu et al. investigate the task offloading and resource allocation in edge environments, and a

deep learning-assisted approach is proposed. Again, none of the studies account for hardware reconfigurable accelerators in their systems. We can extend our algorithms proposed in this paper to an online scheme by incorporating the above deep reinforcement learning methods in the future.

Several studies employ game-theoretic-based algorithms in their research. For instance, two papers, [9] and [10], investigate task offloading and resource allocation, where the allocation of computing resources to wireless devices can be adjusted. In [9], Jošilo et al. tackle the problem of distributing wireless and computing resources to wireless devices within an edge computing system to minimize latency, proposing an algorithm with a provable approximation ratio. In [10], Jošilo et al. expand upon the model in [9] by incorporating network slicing. Both [9] and [10] allow adjustable computing resource allocation for wireless devices. However, this paper includes standalone FPGAs, and the amount of FPGA resources assigned to FPGAs cannot be altered to adjust processing latency (see Section III-D for details). The distinct characteristics of computing latencies for general-purpose servers and FPGAs contribute to the complexity of the latency minimization problem, rendering existing methods in the literature insufficient for addressing the problem considered in this paper. In [22], Liu et al. develop and analyze the joint service function chain deployment and resource allocation problem within edge environments to minimize overall system delay, without considering input/output data uploading/downloading latency. The authors present a congestion game-theoretic-based algorithm with a provable approximation ratio. The algorithm proposed in Section IV can resolve the issues in [9], [10], [22], where our algorithm converges in polynomial steps while their algorithms may require exponential steps. Some research assumes adjustable computing resources for tasks, such as in [9], [10] and [22], while other studies, like [11], consider fixed resource amounts for tasks. This paper, an extended version of our conference paper [23], examines a more general case in which the resource allocation for tasks on general-purpose servers is adjustable, and that for tasks on FPGAs remains fixed.

TABLE I
IMPORTANT NOTATIONS

$\mathcal{I}$	set of wireless devices
$\mathcal{N}$	set of general-purpose servers
$\mathcal{M}$	set of reconfigurable accelerators
$\mathcal{K}$	set of access points
$x_{i,n}$	offloading decision related to server n (binary)
$\mathbf{x}$	$\{x_{i,n} \mid i \in \mathcal{I}, n \in \mathcal{N}\}$
$y_{i,m}$	offloading decision related to FPGA m (binary)
$\mathbf{y}$	$\{y_{i,m} \mid i \in \mathcal{I}, m \in \mathcal{M}\}$
f_i	task size (FLOP) of WD i
F_n	computing capability (FLOP/s) of server n
$\mathbf{A}_m$	resource capacity (e.g., CLBs, memory) of FPGA m
$\mathbf{a}_{i,m}$	resource sizes required by computing task i on FPGA m
$\alpha_{i,n}$	computing resource allocation decision (continuous)
α	$\{\alpha_{i,n} \mid i \in \mathcal{I}, n \in \mathcal{N}\}$
$\delta_{i,n}$	suitability between WD i and server n
δ	$\{\delta_{i,n} \mid i \in \mathcal{I}, n \in \mathcal{N}\}$
$\overline{z}_{i,k}$	uplink AP selection decision (binary)
$\underline{z}_{i,k}$	downlink AP selection decision (binary)
$\overline{c}_i$	uploading data length (bit) of WD i
$\underline{c}_i$	downloading data length (bit) of WD i
$\mathbf{z}$	$\{\overline{z}_{i,k}, \underline{z}_{i,k} \mid i \in \mathcal{I}, k \in \mathcal{K}\}$
$\overline{\beta}_{i,k}$	uplink bandwidth allocation decision (continuous)
$\underline{\beta}_{i,k}$	downlink bandwidth allocation decision (continuous)
β	$\{\overline{\beta}_{i,k}, \underline{\beta}_{i,k} \mid i \in \mathcal{I}, k \in \mathcal{K}\}$
$\gamma_{i,k}$	wireless channel condition (bps/Hz)
γ	$\{\gamma_{i,k} \mid i \in \mathcal{I}, k \in \mathcal{K}\}$
T_i^P	processing latency of WD i (second)
T^P	overall processing latency (second)
T_i^C	communication latency of WD i (second)
T^C	overall communication latency (second)

III. System Model

In this section, we formulate the problem of joint task offloading, AP selection, and resource allocation in heterogeneous edge environments with the goal of minimizing the system latency. Table I presents some main notations.

A. Edge Task Offloading System

1) System Components: We consider an edge system consisting of wireless end devices (WDs), access points (APs), and edge computing devices. There are I WDs in the system, and $\mathcal{I} = [I] \triangleq \{1, 2, \ldots, I\}$ denotes the set of WDs. There are K APs in the system, and $\mathcal{K} = [K] \triangleq \{1, 2, \ldots, K\}$ represents the set of APs. For each AP $k \in \mathcal{K}$, it has an uplink bandwidth of $\overline{B}_k$ bits/s and a downlink bandwidth of $\underline{B}_k$ bits/s. There are two types of edge computing devices, namely general-purpose servers and FPGAs. We use $\mathcal{N} = [N] \triangleq \{1, 2, \ldots, N\}$ to denote the set of general-purpose servers where N is the number of servers. Similarly, $\mathcal{M} = [M]$ represents the set of FPGAs where M is the number of FPGAs. We use F_n to represent the computing capability of server n, such as the number of floating-point operations per second (FLOP/s). For each $m \in \mathcal{M}$, $\mathbf{A}_m = \{A_{m,1}, A_{m,2}, \ldots, A_{m,L}\}$ is the vector representing the amounts of L different resources of FPGA m, i.e., the number of configurable logic blocks (CLBs), Flip-Flops, DSPs, RAMs and so on [24]. $\mathcal{L} = \{1, 2, \ldots, L\}$ denotes the set the L types of resources. Multiple applications can share the resources of an FPGA simultaneously [25], [26].

2) Network Topology: WDs communicate with edge computing devices via APs. Each AP $k \in \mathcal{K}$ has a coverage area, and a WD can be covered by more than one AP. We use $\mathcal{K}_i$ to represent the set of APs covering the location of WD i, where $\mathcal{K}_i \subseteq \mathcal{K}$. APs communicate with edge computing devices by wired links [27], e.g., cellular base stations using fiber optic cables with a speed of up to 200 Gbps and wireless routers using twisted pair cables with a speed of up to 10 Gbps. Around 95 percent of buildings in the United States have fiber-optic infrastructures within 1.5 km, and most of the base stations are connected by fibers [27]. Therefore, compared with the

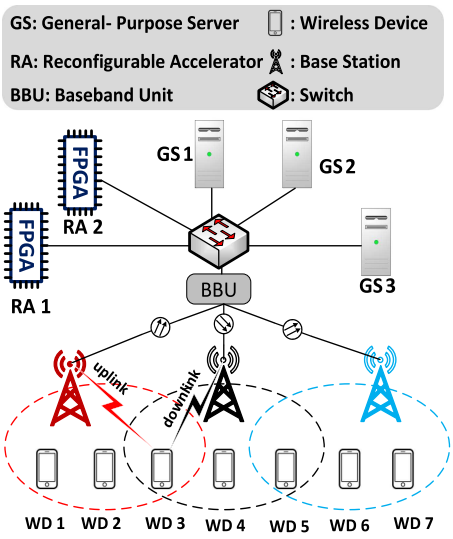


Fig. 1. Example of an edge computing system with $N = 3, M = 2, K = 3$ and $I = 7$. WD 3 uses AP 1 and AP 2 for uploading and downloading data, respectively.

wireless links between WDs and APs, the latency of the wired links between APs and edge computing devices is negligible, which is a typical assumption in the literature [9], [10]. In addition, the algorithms proposed in this paper can be adapted to accommodate situations where the latency between APs and computing devices is not negligible. Fig. 1 shows an example of the network topology.

3) Tasks: WDs generate computing tasks periodically under a given frequency [28]. Each task of WD i has an input data size of $\overline{c}_i$ bits and an output data size of $\underline{c}_i$ bits. Each WD i offloads its tasks to either a server or an FPGA. If WD i offloads its tasks on a server, it takes f_i FLOPs to complete a task. When WD i offloads its tasks on FPGA m, let $\mathbf{a}_{i,m} = \{a_{i,m,1}, \ldots, a_{i,m,L}\}$ represent the necessary resources to implement WD i's function, and the completion time is denoted by $t_{i,m}$. $t_{i,m}$ and $a_{i,m,l}$ can be different across FPGAs. Specifically, experiments in [29] demonstrated that function completion time and resource consumption vary across families of Xilinx, Altera, Actel, and Quick Logic FPGAs.

B. AP Selection and Wireless Resource Management

1) AP Selection Decisions: Each WD $i \in \mathcal{I}$ has to choose an AP $\overline{k}_i \in \mathcal{K}_i$ for uploading input data. Similarly, each WD $i \in \mathcal{I}$ has to choose an AP $\underline{k}_i \in \mathcal{K}_i$ for downloading output data. We use variable $\overline{z}_{i,k} \in \{0, 1\}$ and variable $\underline{z}_{i,k} \in \{0, 1\}$ to represent whether WD i choose AP k as its uploading AP and downloading AP, respectively. Let $\mathbf{z}_i = \{\overline{z}_{i,k} | k \in \mathcal{K}\} \cup \{\underline{z}_{i,k} | k \in \mathcal{K}\}$ be the collection of AP selection decisions of WD i. There are two constraints for the AP selection decisions of each WD i as follows:

$$\sum_{k \in \mathcal{K}_i} \overline{z}_{i,k} = 1 \text{ and } \sum_{k \in \mathcal{K}_i} \underline{z}_{i,k} = 1 \text{ for } i \in \mathcal{I}. \tag{1}$$

In addition, $\mathbf{z} = \{\overline{z}_{i,k} | i \in \mathcal{I}, k \in \mathcal{K}\} \cup \{\underline{z}_{i,k} | i \in \mathcal{I}, k \in \mathcal{K}\}$ is the set of all AP selection decisions. For each AP $k \in \mathcal{K}$, let $\overline{\mathcal{O}}_k(\mathbf{z})$ be the collection of WDs using AP k as their uploading AP under decision $\mathbf{z}$. WDs in $\overline{\mathcal{O}}_k(\mathbf{z})$ share the uplink bandwidth of AP k. Similarly, let $\underline{\mathcal{O}}_k(\mathbf{z})$ be the collection of WDs using AP k as their downloading AP.

2) Wireless Channel Conditions: For each $i \in \mathcal{I}$ and $k \in \mathcal{K}$, there is a bandwidth utilization, denoted by $\gamma_{i,k} \geq 0$, associated with WD i and AP k, which reflects the condition of the wireless condition between WD i and AP k. $\gamma_{i,k}$ is given in advance, which is affected by the distance between WD i and AP k, the noise power of the channel between WD i and AP k, and so on. In particular, we set $\gamma_{i,k} = 0$ if WD i is not covered by AP k, i.e., $\gamma_{i,k} = 0$ if $k \notin \mathcal{K}_i$.

3) Wireless Bandwidth Allocation: If WD $i \in \overline{\mathcal{O}}_k(\mathbf{z})$, there is a continuous variable, $\overline{\beta}_{i,k}$, representing the proportion of the uplink bandwidth of AP k allocated to WD i. Similarly, if WD $i \in \underline{\mathcal{O}}_k(\mathbf{z})$, $\underline{\beta}_{i,k}$ represents the proportion of the downlink bandwidth of AP k allocated to WD i. The system can employ the Orthogonal Frequency Division Multiplexing (OFDMA) scheme, where bandwidth allocation corresponds to the allocation of Resource Units (RUs). Since the total bandwidth allocated to WDs can not exceed the total bandwidth of AP k, we have two constraints as follows:

$$\begin{aligned} \sum_{i \in \mathcal{I}} \overline{z}_{i,k} \overline{\beta}_{i,k} &= \sum_{i \in \overline{\mathcal{O}}_k(\mathbf{z})} \overline{\beta}_{i,k} \leq 1 \\ \sum_{i \in \mathcal{I}} \underline{z}_{i,k} \underline{\beta}_{i,k} &= \sum_{i \in \underline{\mathcal{O}}_k(\mathbf{z})} \underline{\beta}_{i,k} \leq 1. \end{aligned} \tag{2}$$

For the sake of simplicity, we use β to denote the collection of all communication resource management variables, i.e., $\beta = \{\overline{\beta}_{i,k}, \underline{\beta}_{i,k} | i \in \mathcal{I}, k \in \mathcal{K}\}$.

We distinguish the uplink and downlink bandwidth to handle the case that uplink and downlink use different frequency bands, e.g., frequency division multiplexing (FDD) protocols [30]. Our algorithm can handle the case that there is no distinction between uplink and downlink bandwidth, i.e, each AP k has only one bandwidth constraint $\sum_{i \in \overline{\mathcal{O}}_k(\mathbf{z})} \overline{\beta}_{i,k} + \sum_{i \in \underline{\mathcal{O}}_k(\mathbf{z})} \underline{\beta}_{i,k} \leq 1$, which is a degenerated case.

C. Task Offloading and Computing Resource Management

1) Offloading to Server: For each WD $i \in \mathcal{I}$ and server $n \in \mathcal{N}$, there is a variable $x_{i,n} \in \{0, 1\}$. In particular, $x_{i,n} = 1$ if WD i offloads its tasks on server n, and $x_{i,n} = 0$ otherwise. $\mathbf{x}_n = (x_{1,n}; x_{2,n}; \ldots; x_{I,n})$ denotes the collection of $x_{i,n}$ related to server n. In addition, $\mathbf{x}$ denotes the collection of $x_{i,n}$ for $i \in \mathcal{I}, n \in \mathcal{N}$. $\mathcal{O}_n(\mathbf{x})$ is the set of WDs that place their tasks on server n, i.e., $x_{i,n} = 1$ if $i \in \mathcal{O}_n(\mathbf{x})$. If WD i offloads its tasks on server n, i.e., $i \in \mathcal{O}_n(\mathbf{x})$, we use $\alpha_{i,n}$ to denote the proportion of computing capability of server n allocated to WD i. Note that the server can be not only a CPU but also a GPU because the Multi-Process Service (MPS) scheme allows different tasks running on different address spaces of a GPU [31]. Specifically, MPS enables multiple CUDA applications to share a GPU simultaneously. In this setup, GPU resources, such as address space and memory, are allocated non-uniformly across these applications. There is a constraint limiting that the amount

of computing capability allocated to WDs in $\mathcal{O}_n(\mathbf{x})$ can not exceed the total computing capability of server n as follows:

$$\sum_{i\in\mathcal{I}} x_{i,n}\alpha_{i,n} = \sum_{i\in\mathcal{O}_n(\mathbf{x})} \alpha_{i,n} \leq 1. \tag{3}$$

For simplicity, we use $\alpha = \{\alpha_{i,n} | i \in \mathcal{I}, n \in \mathcal{N}\}$ to denote the collection of all computing resource management variables.

2) Suitability Between Servers and Tasks: Since different servers may be equipped with different amounts of CPUs, GPUs, etc., certain servers are more suitable for executing specific types of tasks. For each WD i and server n, there is a suitability $\delta_{i,n} \in [0,1]$ [22]. $\delta_{i,n} \in [0,1]$ depicts how well a server n is fitting for running tasks of WD i. The larger $\delta_{i,n}$, the better the suitability of offloading tasks of WD i to server n.

3) Offloading to FPGAs: For each WD $i \in \mathcal{I}$ and FPGA $m \in \mathcal{M}$, there is a decision variable $y_{i,m} \in \{0,1\}$. In particular, $y_{i,m} = 1$ if WD i places its tasks on FPGA m, and $y_{i,m} = 0$ otherwise. $\mathbf{y}$ is the collection of $y_{i,m}$ for $i \in \mathcal{I}, m \in \mathcal{M}$. Moreover, $\mathcal{O}_m(\mathbf{y})$ represents the set of WDs that place their tasks on FPGA m, i.e., $i \in \mathcal{O}_m(\mathbf{y})$ if $y_{i,m} = 1$. There is a constraint limiting the total amounts of resources required by WDs in $\mathcal{O}_m(\mathbf{y})$ can not exceed the resource amounts of FPGA m as follows:

$$\sum_{i\in\mathcal{I}} y_{i,m}a_{i,m,l} \triangleq \sum_{i\in\mathcal{O}_m(\mathbf{y})} a_{i,m,l} \leq A_{m,l} \text{ for } m \in \mathcal{M}, l \in \mathcal{L}. \tag{4}$$

4) Constraint of Offloading Decision: The collection of offloading decisions is $(\mathbf{x}, \mathbf{y})$. Since each WD i offloads its tasks on either a server $n \in \mathcal{N}$ or an FPGA $m \in \mathcal{M}$, we have the following constraint regarding $(\mathbf{x}, \mathbf{y})$:

$$\sum_{n\in\mathcal{N}} x_{i,n} + \sum_{m\in\mathcal{M}} y_{i,m} = 1 \text{ for } i \in \mathcal{I}. \tag{5}$$

D. Goal of the System

The goal of the system is to minimize the summation of latency of all WDs. The latency of each WD i consists of two parts, namely processing latency T_i^P and communication latency T_i^C.

1) Processing Latency: If WD i places its task on server n, the average processing latency can be expressed as a function of the amount of computing capability allocated to WD i [9], i.e.,

$$T_i^P = x_{i,n}\frac{f_i}{F_n\delta_{i,n}\alpha_{i,n}} \text{ if } i \in \mathcal{O}_n(\mathbf{x}) \tag{6}$$

where $\delta_{i,n}$ is a fixed parameter reflecting the suitability of running tasks of WD i on server n. For example, $\delta_{i,n}$ is different in two cases where the server is a CPU and a GPU. We can tune the above latency by varying $\alpha_{i,n}$ [9], [31]. On the other hand, if WD i offloads its tasks on FPGA m, the processing latency of WD i is $t_{i,m}$, i.e.,

$$T_i^P = y_{i,m}t_{i,m} \text{ if } i \in \mathcal{O}_m(\mathbf{y}). \tag{7}$$

Different from the latencies of tasks on servers, latencies of tasks on FPGAs can not be decreased by increasing the number of configurable logic blocks for the following reasons. First, the functions running on FPGAs are described by hardware description language in advance. Once the hardware description code is given and an FPGA is specified, the resource consumption and the latency are also fixed. Second, it is wasteful and time-consuming to develop different versions of FPGA implementation. Last and most importantly, varying the implementation (reprogramming FPGA) takes time, e.g., hundreds of ms up to tens of seconds, which is fatal to applications requiring extra-low latency.

From (6) and (7), the processing latency is a function of $(\mathbf{x}, \mathbf{y}, \alpha)$, and we use $T^P(\mathbf{x}, \mathbf{y}, \alpha)$ to denote the summation of processing latency of all WDs, i.e.,

$$T^P(\mathbf{x}, \mathbf{y}, \alpha) = \sum_{i\in\mathcal{I}} \left(\sum_{n\in\mathcal{N}} \frac{x_{i,n}f_i}{F_n\delta_{i,n}\alpha_{i,n}} + \sum_{m\in\mathcal{M}} y_{i,m}t_{i,m} \right). \tag{8}$$

2) Communication Latency: Since we focus on edge computing systems, the WDs and APs are in close vicinity; therefore, the propagation delay is negligible, and we only need to consider the transmission delay. The transmission latency of WD i consists of input data uploading latency and output data downloading latency. The uploading transmission delay of WD i is denoted by $\overline{T}_i^C$. In particular,

$$\overline{T}_i^C = \sum_{k\in\mathcal{K}} \frac{\overline{z}_{i,k}\overline{c}_i}{\gamma_{i,k}\cdot\overline{B}_k\cdot\overline{\beta}_{i,k}}. \tag{9}$$

Similarly, the downloading transmission delay of WD i is denoted by $\underline{T}_i^C$, and we have

$$\underline{T}_i^C = \sum_{k\in\mathcal{K}} \frac{\underline{z}_{i,k}\underline{c}_i}{\gamma_{i,k}\cdot\underline{B}_k\cdot\underline{\beta}_{i,k}}. \tag{10}$$

Let T_i^C be the average communication latency of WD i, i.e., $T_i^C = \overline{T}_i^C + \underline{T}_i^C$. The summation of communication latencies of all WDs is a function of $(\mathbf{z}, \beta)$ as follows:

$$T^C(\mathbf{z}, \beta) = \sum_{i\in\mathcal{I}} T_i^C(\mathbf{z}, \beta) = \sum_{i\in\mathcal{I}} \left(\overline{T}_i^C + \underline{T}_i^C \right). \tag{11}$$

E. Problem Formulation

Next, we formally state the problem that we formulated above as an optimization problem, and we refer to the problem as JOAM which is short for Joint task Offloading, AP selection, and resource Management. JOAM is as follows:

$$\min_{\mathbf{x},\mathbf{y},\mathbf{z},\alpha,\beta} \quad T^P(\mathbf{x}, \mathbf{y}, \alpha) + T^C(\mathbf{z}, \beta) \tag{JOAM}$$

$$\text{s.t.} \quad (1)\text{–}(5)$$

$$x_{i,n} \in \{0,1\} \text{ for } i \in \mathcal{I} \text{ and } n \in \mathcal{N} \tag{12}$$

$$y_{i,m} \in \{0,1\} \text{ for } i \in \mathcal{I} \text{ and } m \in \mathcal{M} \tag{13}$$

$$\alpha_{i,n} \in [0,1] \text{ for } i \in \mathcal{O}_n(\mathbf{x}) \text{ and } n \in \mathcal{N} \tag{14}$$

$$\overline{z}_{i,k}, \underline{z}_{i,k} \in \{0,1\} \text{ for } i \in \mathcal{I} \text{ and } k \in \mathcal{K} \tag{15}$$

$$\overline{\beta}_{i,k} \in [0,1] \text{ for } i \in \overline{\mathcal{O}}_k(\mathbf{z}) \text{ and } k \in \mathcal{K} \tag{16}$$

$$\underline{\beta}_{i,k} \in [0,1] \text{ for } i \in \underline{\mathcal{O}}_k(\mathbf{z}) \text{ and } k \in \mathcal{K}. \tag{17}$$

The decision variables of JOAM can be partitioned into two sets, namely $(\mathbf{x}, \mathbf{y}, \alpha)$ and $(\mathbf{z}, \beta)$. There is no coupling between $(\mathbf{x}, \mathbf{y}, \alpha)$ and $(\mathbf{z}, \beta)$ in the constraints. In addition, communication latency T^C is merely determined by $(\mathbf{z}, \beta)$, and processing latency T^P is merely determined by $(\mathbf{x}, \mathbf{y}, \alpha)$. Therefore, JOAM can be divided into two disjoint subproblems, namely the Joint task Offloading and computing resource Management problem (JOM) of minimizing T^P over $(\mathbf{x}, \mathbf{y}, \alpha)$ and the Joint AP selection and communication resource Management problem (JAM) of minimizing T^C over variables $(\mathbf{z}, \beta)$. The two disjoint subproblems are as follows:

$$\min_{\mathbf{x},\mathbf{y},\alpha} \quad T^P(\mathbf{x}, \mathbf{y}, \alpha) \qquad \text{(JOM)}$$

$$\text{s.t.} \quad (3)\text{–}(5), (12)\text{–}(14)$$

$$\min_{\mathbf{z},\beta} \quad T^C(\mathbf{z}, \beta) \qquad \text{(JAM)}$$

$$\text{s.t.} \quad (1)\text{–}(2), (15)\text{–}(17)$$

In what follows, we show the NP-hardness of JOAM. Actually, both JOM and JAM are NP-hard.

Theorem 1: JOAM is NP-hard.

Proof: JOM is a special version of JOAM. To be more specific, JOM is equivalent to JOAM if there is only one AP covering all WDs and having infinite uplink and downlink bandwidth. JOAM is NP-hard because JOM, a special version of JOM, is NP-hard. The NP-hardness of JOM is shown in Theorem 6. □

IV. Algorithm Design for AP Selection and Communication Resource Management

First, we show the hardness of JAM in Theorem 2.

Theorem 2: The JAM problem is NP-hard.

The proof of Theorem 2 is similar to that of Theorem 1 in [10], so we omit it.

A. Communication Resource Management

We first consider finding the continuous communication resource management variables β under any given AP selection decision $\mathbf{z}$. If $\mathbf{z}$ is given, JAM is equivalent to the following problem.

$$\begin{aligned}
\min_{\beta} \quad & \sum_{i\in\mathcal{I}} \left(\overline{T}_i^C(\mathbf{z}, \beta) + \underline{T}_i^C(\mathbf{z}, \beta)\right) \\
\text{s.t.} \quad & \sum_{i\in\overline{\mathcal{O}}_k(\mathbf{z})} \overline{\beta}_{i,k} \le 1, \sum_{i\in\underline{\mathcal{O}}_k(\mathbf{z})} \underline{\beta}_{i,k} \le 1 \text{ for } k \in \mathcal{K} \\
& \overline{\beta}_{i,k} \in [0,1] \text{ for } i \in \overline{\mathcal{O}}_k(\mathbf{z}) \text{ and } k \in \mathcal{K} \\
& \underline{\beta}_{i,k} \in [0,1] \text{ for } i \in \underline{\mathcal{O}}_k(\mathbf{z}) \text{ and } k \in \mathcal{K}. \qquad (18)
\end{aligned}$$

We use $\beta^*(\mathbf{z}) = \{\overline{\beta}^*_{i,k} | k \in \mathcal{K}, i \in \overline{\mathcal{O}}_k(\mathbf{z})\} \cup \{\underline{\beta}^*_{i,k} | k \in \mathcal{K}, i \in \underline{\mathcal{O}}_k(\mathbf{z})\}$ to denote the optimal solution of (18) under AP selection decision $\mathbf{z}$. The optimal communication resource management decision $\beta^*(\mathbf{z})$ is shown in Lemma 1.

Lemma 1: For any feasible $\mathbf{z}$, optimal communication resource management decision $\beta^*(\mathbf{z})$ is as follows:

$$\overline{\beta}^*_{i,k} = \frac{\sqrt{\frac{\overline{c}_i}{\gamma_{i,k}}}}{\sum_{j\in\overline{\mathcal{O}}_k(\mathbf{z})} \sqrt{\frac{\overline{c}_j}{\gamma_{j,k}}}} \text{ for } k \in \mathcal{K}, i \in \overline{\mathcal{O}}_k(\mathbf{z}), \qquad (19)$$

$$\underline{\beta}^*_{i,k} = \frac{\sqrt{\frac{\underline{c}_i}{\gamma_{i,k}}}}{\sum_{j\in\underline{\mathcal{O}}_k(\mathbf{z})} \sqrt{\frac{\underline{c}_j}{\gamma_{j,k}}}} \text{ for } k \in \mathcal{K}, i \in \underline{\mathcal{O}}_k(\mathbf{z}). \qquad (20)$$

The proof of Lemma 1 is omitted due to space limitations. The main idea of Lemma 1 is to derive the optimal solution by exploiting KKT conditions.

Substituting β^* into (18), the optimal communication latency under any feasible AP selection decision $\mathbf{z}$ is equal to

$$\begin{aligned}
\sum_{i\in\mathcal{I}} \left(\overline{T}_i^C + \underline{T}_i^C\right) &= \sum_{k\in\mathcal{K}} \sum_{i\in\overline{\mathcal{O}}_k(\mathbf{z})} \overline{T}_i^C + \sum_{k\in\mathcal{K}} \sum_{i\in\underline{\mathcal{O}}_k(\mathbf{z})} \underline{T}_i^C \\
&= \sum_{k\in\mathcal{K}} \frac{1}{\overline{B}_k} \sum_{i\in\overline{\mathcal{O}}_k(\mathbf{z})} \sqrt{\frac{\overline{c}_i}{\gamma_{i,k}}} \left(\sum_{j\in\overline{\mathcal{O}}_k(\mathbf{z})} \sqrt{\frac{\overline{c}_j}{\gamma_{j,k}}}\right) \\
&\quad + \sum_{k\in\mathcal{K}} \frac{1}{\underline{B}_k} \sum_{i\in\underline{\mathcal{O}}_k(\mathbf{z})} \sqrt{\frac{\underline{c}_i}{\gamma_{i,k}}} \left(\sum_{j\in\underline{\mathcal{O}}_k(\mathbf{z})} \sqrt{\frac{\underline{c}_j}{\gamma_{j,k}}}\right). \qquad (21)
\end{aligned}$$

B. Algorithm Design for AP Selection

For convenience, we introduce some short-formed terms and interpret the problem as a weighted congestion game. Let $\mathcal{R} \triangleq \{(k, upload), (k, download) | k \in \mathcal{K}\}$, where each element in set $\mathcal{R}$ is a tuple representing a kind of communication resource. For example, $(k, upload)$ and $(k, download)$ represent the uploading and downloading bandwidth resources of AP k, respectively. For each resource $r \in \mathcal{R}$, there is a weight m_r associated with it. In particular, $m_r = \frac{1}{\overline{B}_k}$ if $r = (k, upload)$ and $m_r = \frac{1}{\underline{B}_k}$ if $r = (k, download)$. For each WD i, let $\mathcal{Z}_i$ be the set of all feasible $\mathbf{z}_i$. In particular, from the constraints of JAM, we have

$$\mathcal{Z}_i = \left\{ \mathbf{z}_i \,\middle|\, \sum_{k\in\mathcal{K}} \overline{z}_{i,k} = 1, \sum_{k\in\mathcal{K}} \underline{z}_{i,k} = 1, \text{and } \overline{z}_{i,k}, \underline{z}_{i,k} \in \{0,1\} \right\}. \qquad (22)$$

For any given $\mathbf{z}_i \in \mathcal{Z}_i$, $\mathbf{z}_i$ decides the uploading AP and the downloading AP of WD i, and we use $\mathcal{R}_i(\mathbf{z}_i)$ to denote resources that WD i chooses. For example, if WD i chooses AP k_1 and AP k_2 as its uploading and downloading APs respectively, we have $\mathcal{R}_i(\mathbf{z}_i) = \{(k_1, upload), (k_2, download)\}$. Let $p_r^i(\mathbf{z}_i)$ be a value corresponding with the pair of WD i and resources r, which represents the congestion value that WD i contributed to resource r under decision $\mathbf{z}_i$. In particular, for

TABLE II
NOTATION TABLE

$r \in \mathcal{R}$	resource $r \in \mathcal{R} \triangleq \{(k, upload), (k, download) \vert k \in \mathcal{K}\}$
$p_r^i(\mathbf{z}_i)$	congestion value that player i contributed to resource r
$p_r(\mathbf{z})$	congestion value of resource r
$\mathcal{R}_i(\mathbf{z}_i)$	set of resources that player (WD) i chooses

$r \in \{(k, upload) \vert k \in \mathcal{K}\}$,

$$p_r^i(\mathbf{z}_i) = \begin{cases} \sqrt{\overline{c}_i/\gamma_{i,k}}, & \text{if } \overline{z}_{i,k} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

Similarly, for $r = (k, downloading)$ and $k \in \mathcal{K}$,

$$p_r^i(\mathbf{z}_i) = \begin{cases} \sqrt{\underline{c}_i/\gamma_{i,k}}, & \text{if } \underline{z}_{i,k} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

In addition, for each $r \in \mathcal{R}$ and each feasible $\mathbf{z}$, we define the congestion value of resource r under decision $\mathbf{z}$, denoted by $p_r(\mathbf{z})$, which is a function of $\mathbf{z}$ as follows:

$$p_r(\mathbf{z}) = \sum_{i \in \mathcal{I}} p_r^i(\mathbf{z}_i). \tag{25}$$

We summarize the notation defined above in Table II.

Next, by substituting (21) and the terms defined above into JAM, we have that JAM is equivalent to P1 as follows:

$$\min_{\mathbf{z}} \quad \sum_{i \in \mathcal{I}} T_i^C(\mathbf{z}) = \sum_{i \in \mathcal{I}} \sum_{r \in \mathcal{R}_i(\mathbf{z}_i)} m_r p_r^i(\mathbf{z}_i) p_r(\mathbf{z})$$
$$\text{s.t.} \quad \mathbf{z}_i \in \mathcal{Z}_i, i \in \mathcal{I}. \tag{P1}$$

P1 can be interpreted as a weighted congestion game as follows, where $\mathcal{I}$ is the set of players, and $T_i^C(\mathbf{z}) = \sum_{r \in \mathcal{R}_i(\mathbf{z}_i)} m_r p_r^i(\mathbf{z}_i) p_r(\mathbf{z})$ is the cost of player i under decision $\mathbf{z}$. The problem is equivalent to finding the strategy profile of all players, which minimizes the total cost of all players.

Next, we propose an algorithm, called Congestion Game Based Algorithm (*CGBA*), for P1 in Algorithm 1. *CGBA* has a parameter $\lambda \geq 0$ that we can tune. We use *CGBA*(λ) to denote *CGBA* with parameter λ. *CGBA*(λ) starts by randomly selecting a feasible decision for each WD i (Line 1). At the beginning of each iteration, the algorithm checks whether there exists a WD i that can reduce its latency to less than $(1-\lambda)$ times its previous latency by changing its own decision. In particular, let $\Delta_i(\mathbf{z}_i) = T_i^C(\mathbf{z}_i, \mathbf{z}_{-i}) - \min_{\bar{\mathbf{z}}_i \in \mathcal{Z}_i} T_i^C(\bar{\mathbf{z}}_i, \mathbf{z}_{-i})$ represent the maximum decrease in the cost of WD i that can be achieved by changing only $\mathbf{z}_i$. $\mathcal{J}$ is defined as the set of WDs such that $i \in \mathcal{J}$ if and only if WD i can decrease its cost by changing only $\mathbf{z}_i$ to a value that is less than $(1-\lambda)T_i^C(\mathbf{z})$. In addition, $\Delta_{\mathcal{J}}$ represents $\sum_{i \in \mathcal{J}} \Delta_i$. Line 4–Line 20 compute $\mathcal{J}$, Δ_i and $\Delta_{\mathcal{J}}$. If $\mathcal{J}$ is empty or $\Delta_{\mathcal{J}}/T^C(\mathbf{z})$ is less than parameter λ, the algorithm terminates. If $\mathcal{J}$ is not empty, the algorithm selects the WD $i^\star$ (Line 24), where $i^\star = \arg\max \Delta_i$. Afterward, WD $i^\star$ updates its decision by selecting the option that minimizes its cost.

We use $\hat{\mathbf{z}}$ to denote the AP selection decision made by Algorithm 1, and use $\mathbf{z}^* = (\mathbf{z}_1^*, \ldots, \mathbf{z}_I^*)$ to denote the optimal AP selection decision. In addition, $\mathbf{z}_{-i}$ represents the decision of all WDs except WD i, i.e., $\mathbf{z} = (\mathbf{z}_{-i}, \mathbf{z}_i)$.

Algorithm 1: *CGBA*(λ)

Input: $\overline{B}_k, \underline{B}_k$ for $k \in \mathcal{K}$, $\overline{c}_i, \underline{c}_i$ for $i \in \mathcal{I}$, $\gamma_{i,k}$ for $i \in \mathcal{I}, k \in \mathcal{K}$
Output: A feasible solution to P1: $\hat{\mathbf{z}}$

1 Initialization: choose $\mathbf{z}_i$ from $\mathcal{Z}_i$ randomly for $k \in \mathcal{K}$;
2 **while** *True* **do**
3 $\mathcal{J} = \emptyset$, $T = 0$, $\Delta_{\mathcal{J}} = 0$;
4 **for** $i \in \mathcal{I}$ **do**
5 $T_i^{old} := T_i^C(\mathbf{z}_i, \mathbf{z}_{-i})$;
6 $T := T + T_i^{old}$;
7 $T_i^{new} := \infty$;
8 **for** $\bar{\mathbf{z}}_i \in \mathcal{Z}$ **do**
9 $T_{temp} := T_i^C(\bar{\mathbf{z}}_i, \mathbf{z}_{-i})$;
10 **if** $T_{temp} < T_i^{new}$ **then**
11 $T_i^{new} := T_{temp}$;
12 $\mathbf{z}_i' := \bar{\mathbf{z}}_i$;
13 **end**
14 **end**
15 $\Delta_i := T_i^{old} - T_i^{new}$;
16 **if** $\Delta_i / T_i^{old} > \lambda$ **then**
17 $\mathcal{J} := \mathcal{J} \cup \{i\}$;
18 $\Delta_{\mathcal{J}} = \Delta_{\mathcal{J}} + \Delta_i$
19 **end**
20 **end**
21 **if** $(\mathcal{J} == \emptyset)$ || $(\Delta_{\mathcal{J}}/T < \lambda)$ **then**
22 break;
23 **else**
24 $i^\star := \arg\max_{i \in \mathcal{I}} \Delta_i$;
25 $\mathbf{z}_{i^\star} := \mathbf{z}_{i^\star}'$;
26 $\mathbf{z} := (\mathbf{z}_{i^\star}, \mathbf{z}_{-i^\star})$;
27 **end**
28 **end**
29 Return $\hat{\mathbf{z}} = \mathbf{z}$;

In what follows, we analyze the performance of *CGBA*(λ). First, we show that P1 is an exact potential game, as shown in Lemma 2.

Lemma 2: P1 is an exact potential game where the potential function is

$$P(\mathbf{z}) \triangleq \sum_{i \in \mathcal{I}} \sum_{r \in \mathcal{R}_i(\mathbf{z}_i)} \left(m_r \sum_{j=1}^{i} p_r^j(\mathbf{z}_j) \right) p_r^i(\mathbf{z}_i). \tag{26}$$

For all feasible $\mathbf{z}_i$, $\mathbf{z}_i'$, and $\mathbf{z}_{-i}$, we have

$$T_i^C(\mathbf{z}_i, \mathbf{z}_{-i}) - T_i^C(\mathbf{z}_i', \mathbf{z}_{-i}) = P(\mathbf{z}_i, \mathbf{z}_{-i}) - P(\mathbf{z}_i', \mathbf{z}_{-i}). \tag{27}$$

In addition, $T^C(\mathbf{z}) > P(\mathbf{z}) > 0$ holds for all feasible $\mathbf{z}$.

The proof of Lemma 2 can be found in [32], [33], so we omit it. Based on Lemma 2, we have the performance guarantee for *CGBA*(0) as shown in the following theorem.

Theorem 3: *CGBA*(0) terminates at a decision $\hat{\mathbf{z}}$ with $2.62 \cdot T^C(\mathbf{z}^*) \geq T^C(\hat{\mathbf{z}})$ after a finite number of iterations, where $\mathbf{z}^*$ is the optimal decision.

Proof: Part A: Proof of CGBA(0) terminates in finite steps Since Lemma 2 holds, the value of the potential function will keep decreasing under *CGBA*(0). That is, there is no cycle in the path of $\mathbf{z}$ under *CGBA*(0). Also, the number of feasible $\mathbf{z}$ is finite.

Therefore, *CGBA*(0) will terminate after a finite number of iterations. Assume *CGBA*(0) terminates in $\hat{\mathbf{z}} = (\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_K)$. From Line 2 of *CGBA*(0), there is no i with $T_i^C(\mathbf{z}_i, \mathbf{z}_{-i}) > \min_{\bar{\mathbf{z}}_i \in \mathcal{Z}_i} T_i^C(\bar{\mathbf{z}}_i, \mathbf{z}_{-i})$. That is, each i in $\mathcal{I}$ can not decrease its own cost by changing only its own strategy, which is the definition of Nash equilibrium.

Part B: Proof of CGBA(0)'s Approximation Ratio

$\hat{\mathbf{z}} = (\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_I)$ represents the AP selection decision got by *CGBA*(0). $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_I)$ can be any feasible AP selection decision. From the part A, we have that $\hat{\mathbf{z}}$ is a pure Nash equilibrium. For any $i \in \mathcal{I}$, because $\hat{\mathbf{z}}$ is a Nash equilibrium, we have

$$\begin{aligned} T_i^C(\hat{\mathbf{z}}_i, \hat{\mathbf{z}}_{-i}) &\le T_i^C(\mathbf{z}_i, \hat{\mathbf{z}}_{-i}) \\ &= \sum_{r \in \mathcal{R}_i(\mathbf{z}_i)} m_r p_r^i(\mathbf{z}_i) p_r(\mathbf{z}_i, \hat{\mathbf{z}}_{-i}) \\ &\le \sum_{r \in \mathcal{R}_i(\mathbf{z}_i)} m_r p_r^i(\mathbf{z}_i) \left(p_r(\hat{\mathbf{z}}) + p_r^i(\mathbf{z}_i)\right). \end{aligned} \tag{28}$$

Summing (28) up for $i \in \mathcal{I}$, we have

$$T^C(\hat{\mathbf{z}}) \le \sum_{i \in \mathcal{I}} \sum_{r \in \mathcal{R}_i(\mathbf{z}_i)} m_r p_r^i(\mathbf{z}_i) \left(p_r(\hat{\mathbf{z}}) + p_r^i(\mathbf{z}_i)\right). \tag{29}$$

Interchanging the order of the double summation in (29), we have

$$\begin{aligned} T^C(\hat{\mathbf{z}}) &\le \sum_{i \in \mathcal{I}} \sum_{r \in \mathcal{R}_i(\mathbf{z}_i)} m_r p_r^i(\mathbf{z}_i) \left(p_r(\hat{\mathbf{z}}) + p_r^i(\mathbf{z}_i)\right) \\ &= \sum_{r \in \mathcal{R}} m_r \sum_{i \in \mathcal{I}} p_r^i(\mathbf{z}_i) \left(p_r(\hat{\mathbf{z}}) + p_r^i(\mathbf{z}_i)\right). \end{aligned} \tag{30}$$

The equation in (30) holds because we have $p_r^i(\mathbf{z}_i) = 0$ if $r \notin \mathcal{R}_i(\mathbf{z}_i)$ from the definition of $p_r^i(\mathbf{z}_i)$. Since $\sum_{i \in \mathcal{I}} (p_r^i(\mathbf{z}_i))^2 \le (\sum_{i \in \mathcal{I}} p_r^i(\mathbf{z}_i))^2 = (p_r(\mathbf{z}))^2$ and (30) holds, we have

$$\begin{aligned} T^C(\hat{\mathbf{z}}) &\le \sum_{r \in \mathcal{R}} m_r \sum_{i \in \mathcal{I}} p_r^i(\mathbf{z}_i) p_r(\hat{\mathbf{z}}) + \sum_{r \in \mathcal{R}} m_r \left(p_r(\mathbf{z})\right)^2 \\ &= \sum_{r \in \mathcal{R}} m_r \sum_{i \in \mathcal{I}} p_r^i(\mathbf{z}_i) p_r(\hat{\mathbf{z}}) + T^C(\mathbf{z}) \\ &= \sum_{r \in \mathcal{R}} m_r p_r(\mathbf{z}) p_r(\hat{\mathbf{z}}) + T^C(\mathbf{z}). \end{aligned} \tag{31}$$

Then, applying Cauchy–Schwarz inequality to (31), we have

$$\begin{aligned} T^C(\hat{\mathbf{z}}) &\le \sum_{r \in \mathcal{R}} m_r p_r(\mathbf{z}) p_r(\hat{\mathbf{z}}) + T^C(\mathbf{z}) \\ &\le \sqrt{\sum_{r \in \mathcal{R}} m_r (p_r(\mathbf{z}))^2 \sum_{r \in \mathcal{R}} m_r \left(p_r(\hat{\mathbf{z}})\right)^2} + T^C(\mathbf{z}) \\ &= \sqrt{T^C(\mathbf{z}) T^C(\hat{\mathbf{z}})} + T^C(\mathbf{z}). \end{aligned} \tag{32}$$

From (32), we have $\frac{T^C(\hat{\mathbf{z}})}{T^C(\mathbf{z})} \le \sqrt{\frac{T^C(\hat{\mathbf{z}})}{T^C(\mathbf{z})}} + 1$. Solving the above inequation, we have $\frac{T^C(\hat{\mathbf{z}})}{T^C(\mathbf{z})} \le \frac{3+\sqrt{5}}{2} \approx 2.62$. Since $\mathbf{z}$ can be any feasible strategy profile, the approximation ratio of *CGBA*(0) is 2.62, which proves Theorem 3. □

Theorem 3 shows that *CGBA*(0) is a 2.62-approximation algorithm. Simulation results show that the time complexity of *CGBA*(0) is linear to I, and the average cost under *CGBA*(0) is around $1.013\times$ the optimum.

In what follows, we show the main result of this section, i.e., the performance guarantee for *CGBA*(λ), in Theorem 4. Theorem 4 indicates that *CGBA*(λ) with $\lambda \in (0, 1/8)$ can generate an approximate solution in polynomial time. In addition, we can tune parameter λ to balance the approximation ratio and time complexity.

Theorem 4: For $\lambda \in (0, \frac{1}{8})$, *CGBA*($\lambda$) terminates at a decision $\hat{\mathbf{z}}$ with $T^C(\hat{\mathbf{z}}) \le \frac{2.62}{1-8\lambda} T^C(\mathbf{z}^*)$ in at most $\mathcal{O}(\frac{I}{\lambda} \log(\frac{P_0}{P^*}))$ iterations, and each iteration takes $O(I^2\, K^2)$ steps.

Proof: The prove the theorem, we first prove Claim 1 as follows:

Claim 1: Let $\mathbf{z}^*$ be the optimal decision. For any feasible decision $\mathbf{z}$, we have $\sum_{i \in \mathcal{I}} T_i^C(\mathbf{z}_i^*, \mathbf{z}_{-i}) \le \sqrt{T^C(\mathbf{z}) T^C(\mathbf{z}^*)} + T^C(\mathbf{z}^*)$.

The proof of Claim 1 is as follows. For each $i \in \mathcal{I}$, we have

$$\begin{aligned} T_i^C(\mathbf{z}_i^*, \mathbf{z}_{-i}) &= \sum_{r \in \mathcal{R}_i(\mathbf{z}_i^*)} m_r p_r^i(\mathbf{z}_i^*) p_r(\mathbf{z}_i^*, \mathbf{z}_{-i}) \\ &\le \sum_{r \in \mathcal{R}_i(\mathbf{z}_i^*)} m_r p_r^i(\mathbf{z}_i^*) (p_r(\mathbf{z}) + p_r^i(\mathbf{z}_i^*)) \\ &= \sum_{r \in \mathcal{R}_i(\mathbf{z}_i^*)} m_r p_r^i(\mathbf{z}_i^*) p_r(\mathbf{z}) + \sum_{r \in \mathcal{R}_i(\mathbf{z}_i^*)} m_r (p_r^i(\mathbf{z}_i^*))^2. \end{aligned} \tag{33}$$

Summing the above equation up for $i \in \mathcal{I}$, we have

$$\begin{aligned} &\sum_{i \in \mathcal{I}} T_i^C(\mathbf{z}_i^*, \mathbf{z}_{-i}) \\ &\le \sum_{i \in \mathcal{I}} \sum_{r \in \mathcal{R}_i(\mathbf{z}_i^*)} m_r p_r^i(\mathbf{z}_i^*) p_r(\mathbf{z}) + \sum_{i \in \mathcal{I}} \sum_{r \in \mathcal{R}_i(\mathbf{z}_i^*)} m_r (p_r^i(\mathbf{z}_i^*))^2 \\ &= \sum_{r \in \mathcal{R}} m_r p_r(\mathbf{z}^*) p_r(\mathbf{z}) + \sum_{r \in \mathcal{R}} m_r \sum_{i \in \mathcal{I}} (p_r^i(\mathbf{z}_i^*))^2 \\ &\le \sqrt{\sum_{r \in \mathcal{R}} m_r (p_r(\mathbf{z}^*))^2 \sum_{r \in \mathcal{R}} m_r (p_r(\mathbf{z}))^2} \\ &\quad + \sum_{r \in \mathcal{R}} m_r (p_r(\mathbf{z}))^2 \\ &= \sqrt{T^C(\mathbf{z}) T^C(\mathbf{z}^*)} + T^C(\mathbf{z}^*), \end{aligned} \tag{34}$$

which proves Claim 1.

Let $\mathbf{z}_i'$ be the best response of WD i given profile $\mathbf{z}_{-i}$, then we have

$$\sum_{i \in \mathcal{I}} T_i^C(\mathbf{z}_i', \mathbf{z}_{-i}) \le \sum_{i \in \mathcal{I}} T_i^C(\mathbf{z}_i^*, \mathbf{z}_{-i}). \tag{35}$$

Thus, we have

$$T^C(\mathbf{z}) - \sum_{i \in \mathcal{I}} T_i^C(\mathbf{z}_i^*, \mathbf{z}_{-i}) \le T^C(\mathbf{z}) - \sum_{i \in \mathcal{I}} T_i^C(\mathbf{z}_i', \mathbf{z}_{-i}). \tag{36}$$

Combining the above equation and Claim 1, we have

$$T^C(\mathbf{z}) \leq \sqrt{T^C(\mathbf{z})T^C(\mathbf{z}^*)} + T^C(\mathbf{z}^*) + \left(T^C(\mathbf{z}) - \sum_{i\in\mathcal{I}} T^C(\mathbf{z}'_i, \mathbf{z}_{-i})\right). \quad (37)$$

Let $x^2 = \frac{T^C(\mathbf{z})}{T^C(\mathbf{z}^*)}$ and $y = \frac{(T^C(\mathbf{z}) - \sum_{i\in\mathcal{I}} T^C(\mathbf{z}'_i, \mathbf{z}_{-i}))}{T^C(\mathbf{z}^*)}$. The above equation can be rewritten as $x^2 \leq x + 1 + y$. For coordinates (x,y) with $x > 0$ and $y > 0$, the area defined by the inequality $x^2 \leq x + 1 + y$ is a subset of the area defined by $x^2 \leq 2.62 + 2y$. Thus, we have $x^2 \leq 2.62 + 2y$. That is, we have

$$T^C(\mathbf{z}) \leq 2.62\, T^C(\mathbf{z}^*) + 2\left(T^C(\mathbf{z}) - \sum_{i\in\mathcal{I}} T^C(\mathbf{z}'_i, \mathbf{z}_{-i})\right). \quad (38)$$

For any profile $\mathbf{z}$, let $\Delta_i(\mathbf{z}) \triangleq T_i^C(\mathbf{z}) - T_i^C(\mathbf{z}'_i, \mathbf{z}_{-i})$ and $\Delta(\mathbf{z}) \triangleq \sum_{i\in\mathcal{I}} \Delta_i(\mathbf{z})$ where $\mathbf{z}'_i$ is the best response of WD i. Also, for any $\mathcal{I}' \in \mathcal{I}$, $\Delta_{\mathcal{I}'}(\mathbf{z}) \triangleq \sum_{i\in\mathcal{I}'} \Delta_i(\mathbf{z})$. Define $\mathbf{z}$ is a ϵ-approximate λ-equilibria if $\Delta_{\mathcal{J}}(\mathbf{z}) \leq (\lambda + \epsilon)T^C(\mathbf{z})$, where $\mathcal{J}$ is the set of WDs with $(1-\lambda)T_i^C(\mathbf{z}_i, \mathbf{z}_{-i}) > \min_{\bar{\mathbf{z}}_i \in \mathcal{Z}_i} T_i^C(\bar{\mathbf{z}}_i, \mathbf{z}_{-i})$. Then, we have

$$\Delta_{\bar{\mathcal{J}}} \leq \lambda \sum_{i\in\bar{\mathcal{J}}} T_i^C(\mathbf{z}) \qquad \text{for } i \in \bar{\mathcal{J}} \triangleq \mathcal{I} \setminus \mathcal{J} \quad (39)$$

$$\Delta_{\mathcal{J}}(\mathbf{z}) \leq (\lambda + \epsilon)T^C(\mathbf{z}) \qquad \text{for } i \in \mathcal{J}. \quad (40)$$

Thus, if $\mathbf{z}$ is a ϵ-approximate λ-equilibria, we have

$$\Delta(\mathbf{z}) = \Delta_{\bar{\mathcal{J}}}(\mathbf{z}) + \Delta_{\mathcal{J}}(\mathbf{z}) \leq 2(\lambda + \epsilon)T^C(\mathbf{z}). \quad (41)$$

For any state $\mathbf{z}$, let $\epsilon(\mathbf{z}) = \frac{\Delta_{\mathcal{J}}(\mathbf{z})}{T^C(\mathbf{z})}$. From the definition of ϵ-approximate λ-equilibria, $\mathbf{z}$ is $\epsilon(\mathbf{z})$-approximate λ-equilibria because

$$\Delta_{\mathcal{J}}(\mathbf{z}) = \frac{\Delta_{\mathcal{J}}(\mathbf{z})}{T^C(\mathbf{z})} T^C(\mathbf{z}) = \epsilon(\mathbf{z})T^C(\mathbf{z}) \leq (\epsilon(\mathbf{z}) + \lambda)T^C(\mathbf{z}). \quad (42)$$

Next, let $\mathbf{z}^t$ be the $\mathbf{z}$ at the t-th iteration of *CGBA*(λ), we consider two cases as follows.

Case 1: $\epsilon(\mathbf{z}^t) \leq \lambda$. From (41), we have

$$\Delta(\mathbf{z}^t) = \Delta_{\bar{\mathcal{J}}}(\mathbf{z}^t) + \Delta_{\mathcal{J}}(\mathbf{z}^t) \leq 2(\lambda + \epsilon(\mathbf{z}^t))T^C(\mathbf{z}^t) \leq 4\lambda T^C(\mathbf{z}^t). \quad (43)$$

Then, from (38) and (43), we have

$$T^C(\mathbf{z}^t) \leq 2.62\, T^C(\mathbf{z}^*) + 2\Delta(\mathbf{z}^t) \leq 2.62\, T^C(\mathbf{z}^*) + 8\lambda T^C(\mathbf{z}^t). \quad (44)$$

That is,

$$T^C(\mathbf{z}^t) \leq \frac{2.62\, T^C(\mathbf{z}^*)}{(1 - 8\lambda)}. \quad (45)$$

Case 2: $\epsilon(\mathbf{z}^t) > \lambda$. Under Case 2, we have $\Delta_{\mathcal{J}}(\mathbf{z}^t) > \lambda T^C(\mathbf{z}^t)$. That is, there exists i such that

$$\Delta_i(\mathbf{z}^t) > \frac{\lambda}{I} T^C(\mathbf{z}^t) > \frac{\lambda}{I} P(\mathbf{z}^t). \quad (46)$$

Let $i^\star$ be the player that changes its decision at iteration t. That is, $\Delta_{i^\star}(\mathbf{z}^t) = \max_{i\in I} \Delta_i(\mathbf{z}^t) > \frac{\lambda}{I} P(\mathbf{z}^t)$. Let P_t be the potential value at iteration t. From Lemma 2, we have $P_t - P_{t+1} = \Delta_{i^\star}(\mathbf{z}^t) \geq \frac{\lambda}{I} P(\mathbf{z}^t) = \frac{\lambda}{I} P_t$.

Let P_0 and P^* be the initial and the minimum value of the potential function $P(\mathbf{z})$, respectively, and both P_0 and P^* are finite positive values. Let

$$t' = \frac{\log(P^*/P_0)}{\log(1 - \lambda/I)} = O\left(\frac{I}{\lambda} \log\left(\frac{P_0}{P^*}\right)\right). \quad (47)$$

If there exists $t \leq t'$ such $\mathbf{z}^t$ is of case 1, the theorem holds. Otherwise, $\mathbf{z}_t$ is of case 2 for $t \leq t'$. Then, we have $P_{t'} < P_0(1 - \frac{\lambda}{I})^{t'} = P^*$, which contradicts to the fact that P^* is the minimum value of the potential function. Thus, the algorithm will terminates in at most $\mathcal{O}(\frac{I}{\lambda} \log(\frac{P_0}{P^*}))$ iterations.

We now analyze the time complexity of each iteration (from Line 2 to Line 28). From the objective function of P1, computing $T_i^C(\mathbf{z})$ involves a double summation of total $|\mathcal{R}_i(\mathbf{z}_i)| \times I$ terms. As each WD i chooses two resources, i.e., $|\mathcal{R}_i(\mathbf{z}_i)| = 2$, the time complexity of computing $T_i^C(\mathbf{z})$ is $\mathcal{O}(I)$. Next, we focus on the time complexity of the for loop (from Line 4 to Line 20), which iterates for I times. In addition, the time complexity of Line 5 to Line 7 is $\mathcal{O}(I)$. From the definition of $\mathcal{Z}_i$, we know that set $\mathcal{Z}_i$ has K^2 members. Therefore, the time complexity of the for loop (from Line 8 to Line 14) is $\mathcal{O}(I) \times K^2 = \mathcal{O}(IK^2)$. The time complexity of Line 15 to Line 19 is O(1). Thus, the time complexity of the for loop (Line 4-Line 20) is $I \times (\mathcal{O}(IK) + \mathcal{O}(IK^2) + \mathcal{O}(1)) = \mathcal{O}(I^2\, K^2)$. In addition, the time complexity of Line 21 to Line 27 is dominated by that of Line 24, which is equivalent to $\mathcal{O}(I)$. As a result, the time complexity of each iteration of the while loop is equal to the time complexity of Line 4-Line 20 plus the time complexity of Line 21-Line 27, i.e., $\mathcal{O}(I^2\, K^2) + \mathcal{O}(I) = \mathcal{O}(I^2\, K^2)$. □

Next we consider a variant of *CGBA*(λ) named *FCGBA*, representing fast *CGBA*. *FCGBA* converges in I iteration. At iteration i, WD i chooses its best responses without considering WDs $i+1, i+2, \ldots, I$. *FCGBA* is formally stated in Algorithm 2.

Theorem 5: *FCGBA* has an approximation ratio of $(\sqrt{3}+1)^2$ and converges in $\mathcal{O}(I^2\, K^2)$ steps.

Proof: Use $\bar{\mathbf{z}}_i$ to denote the decision of WD i made at the i-th iteration, where $i \in \mathcal{I}$. Let $\bar{\mathbf{z}}^i = (\bar{\mathbf{z}}_1, \ldots, \bar{\mathbf{z}}_i, \emptyset, \ldots, \emptyset)$ be the decision profile of WDs after i iterations. The communication latency of users in $[i]$ under decision $\bar{\mathbf{z}}^i$ is denoted as $T_{\leq i}^C(\bar{\mathbf{z}}^i)$. Let $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_I)$ be any feasible decision. Then, we have

$$T_{\leq i}^C(\bar{\mathbf{z}}^i) = \sum_{j=1}^{i} \sum_{r\in\mathcal{R}_i(\bar{\mathbf{z}}_i)} \left(m_r \sum_{j=1}^{i} p_r^j(\bar{\mathbf{z}}_j)\right) p_r^i(\bar{\mathbf{z}}_i) = \sum_{r\in\mathcal{R}} m_r p_r(\bar{\mathbf{z}}^i) p_r(\bar{\mathbf{z}}^i). \quad (48)$$

Algorithm 2: *FCGBA*

Input: $\overline{B}_k, \underline{B}_k$ for $k \in \mathcal{K}$, $\overline{c}_i, \underline{c}_i$ for $i \in \mathcal{I}$, $\gamma_{i,k}$ for $i \in \mathcal{I}, k \in \mathcal{K}$
Output: A feasible solution to P1: $\bar{\mathbf{z}}$

```
1  Initialization: set z̄_i := ∅ for k ∈ K;
2  for i ∈ I do
3      T_i^new = ∞;
4      for ẑ_i ∈ Z do
5          T_temp := T_i^C(ẑ_i, z_{-i});
6          if T_temp < T_i^new then
7              T_i^new := T_temp;
8              z'_i := ẑ_i;
9          end
10     end
11     z̄_i := z'_i;
12 end
13 Return z̄;
```

By partitioning set $\mathcal{R}$ to $\mathcal{R} \setminus \mathcal{R}_i(\bar{\mathbf{z}}_i)$ and $\mathcal{R}_i(\bar{\mathbf{z}}_i)$, we have

$$\begin{aligned} T^C_{\le i}(\bar{\mathbf{z}}^i) &= \sum_{r\in\mathcal{R}\setminus\mathcal{R}_i(\bar{\mathbf{z}}_i)} m_r p_r(\bar{\mathbf{z}}^{i-1}) p_r(\bar{\mathbf{z}}^{i-1}) \\ &\quad + \sum_{r\in\mathcal{R}_i(\bar{\mathbf{z}}_i)} m_r (p_r(\bar{\mathbf{z}}^{i-1}) + p^i_r(\bar{\mathbf{z}}_i))^2 \\ &\le \sum_{r\in\mathcal{R}} m_r p_r(\bar{\mathbf{z}}^{i-1}) p_r(\bar{\mathbf{z}}^{i-1}) \\ &\quad + \sum_{r\in\mathcal{R}_i(\bar{\mathbf{z}}_i)} 2 m_r p^i_r(\bar{\mathbf{z}}_i)(p_r(\bar{\mathbf{z}}^{i-1}) + p^i_r(\bar{\mathbf{z}}_i)) \\ &\overset{(a)}{\le} \sum_{r\in R} m_r p_r(\bar{\mathbf{z}}^{i-1}) p_r(\bar{\mathbf{z}}^{i-1}) \\ &\quad + \sum_{r\in\mathcal{R}_i(\mathbf{z}_i)} 2 m_r p^i_r(\mathbf{z}_i)(p_r(\bar{\mathbf{z}}^{i-1}) + p^i_r(\mathbf{z}_i)) \\ &= T^C_{\le i-1}(\bar{\mathbf{z}}^{i-1}) \\ &\quad + \sum_{r\in\mathcal{R}_i(\mathbf{z}_i)} 2 m_r p^i_r(\mathbf{z}_i)(p_r(\bar{\mathbf{z}}^{i-1}) + p^i_r(\mathbf{z}_i)). \end{aligned} \tag{49}$$

Inequality (a) holds because $\bar{\mathbf{z}}_i$ minimizes the latency of WD i, i.e., $\sum_{r\in\mathcal{R}_i(\mathbf{z}_i)} m_r p^i_r(\mathbf{z}_i)(p_r(\bar{\mathbf{z}}^{i-1}) + p^i_r(\mathbf{z}_i))$. From (49), we have

$$\begin{aligned} T^C(\bar{\mathbf{z}}^I) = T^C_{\le I}(\bar{\mathbf{z}}^I) &\le T^C_{\le I-1}(\bar{\mathbf{z}}^I) \\ &\quad + \sum_{r\in\mathcal{R}_I(\mathbf{z}_I)} 2 m_r p^I_r(\mathbf{z}_I)(p_r(\bar{\mathbf{z}}^{I-1}) + p^I_r(\mathbf{z}_I)) \\ &\le \cdots \le \sum_{i=1}^{I} \sum_{r\in\mathcal{R}_i(\mathbf{z}_i)} 2 m_r p^i_r(\mathbf{z}_i)(p_r(\bar{\mathbf{z}}^{i-1}) + p^i_r(\mathbf{z}_i)). \end{aligned} \tag{50}$$

Then, we have

$$\begin{aligned} T^C(\bar{\mathbf{z}}^I) &\le 2 \sum_{i\in\mathcal{I}} \sum_{r\in\mathcal{R}_i(\mathbf{z}_i)} m_r p^i_r(\mathbf{z}_i)(p_r(\bar{\mathbf{z}}^I) + p^i_r(\mathbf{z}_i)) \\ &= 2 \sum_{r\in\mathcal{R}} \sum_{i\in\mathcal{I}} m_r p^i_r(\mathbf{z}_i)(p_r(\bar{\mathbf{z}}^I) + p^i_r(\mathbf{z}_i)) \\ &\le 2 \sum_{r\in\mathcal{R}} m_r p_r(\bar{\mathbf{z}}^I) p_r(\mathbf{z}) + 2 \sum_{r\in\mathcal{R}} m_r p_r(\mathbf{z}) p_r(\mathbf{z}) \\ &\le 2 \sqrt{\sum\nolimits_{r\in\mathcal{R}} m_r (p_r(\bar{\mathbf{z}}^I))^2 \sum\nolimits_{r\in\mathcal{R}} m_r \left(p_r(\mathbf{z})\right)^2} \\ &\quad + 2\, T^C(\mathbf{z}) \\ &= 2 \sqrt{T^C(\bar{\mathbf{z}}^I) T^C(\mathbf{z})} + 2\, T^C(\mathbf{z}). \end{aligned} \tag{51}$$

From (51), we have

$$\frac{T^C(\bar{\mathbf{z}}^I)}{T^C(\mathbf{z})} \le 2 \sqrt{\frac{T^C(\bar{\mathbf{z}}^I)}{T^C(\mathbf{z})}} + 2. \tag{52}$$

Solving the above inequation, we have

$$T^C(\bar{\mathbf{z}}^I) \le (1+\sqrt{3})^2 T^C(\mathbf{z}), \tag{53}$$

which proves the approximation ratio part of the theorem.

We now analyze the time complexity of *FCGBA*. The first for loop (Line 2–Line 12) of Algorithm 2 runs for I times. From the proof of Theorem 4, the time complexity of second for loop (Line 4–Line 10) is $\mathcal{O}(IK^2)$. Thus, we have the time complexity of *FCGBA* is $I \times \mathcal{O}(IK^2) = \mathcal{O}(I^2\, K^2)$. □

V. Algorithm Design for Task Offloading and Computing Resource Management

This section focuses on designing algorithms for JOM. We first show the hardness of JOM.

Theorem 6: JOM is NP-hard, and no polynomial-time approximation algorithm is possible unless there are some additional assumptions.

The main idea of the proof is to show a special version of JOM is equivalent to the Generalized Assignment Problem (GAP) defined in Chapter 48 of [34]. Under the special version of JOM, there is no general-purpose server and only one type of resource constraint, i.e., $N = 0$ and $L = 1$. Detailed proof of Theorem 6 is omitted due to space limitations.

A. Computing Resource Management

Next, we address the problem of determining the optimal computing resource management decision α under any given offloading decision $(\mathbf{x}, \mathbf{y})$. For WDs placing their tasks on reconfigurable accelerators, their latencies are fixed. Therefore, if $(\mathbf{x}, \mathbf{y})$ is given, JOM is equivalent to minimizing the summation of T^P_i for $i \in \bigcup_{n\in\mathcal{N}} \mathcal{O}_n(\mathbf{x})$ over α as follows:

$$\begin{aligned} \min_{\alpha} \quad & \sum_{n\in\mathcal{N}} \sum_{i\in\mathcal{O}_n(\mathbf{x})} \frac{f_i}{F_n \delta_{i,n} \alpha_{i,n}} \\ \text{s.t.} \quad & \alpha_{i,n} \in [0,1] \text{ for } n \in \mathcal{N}, i \in \mathcal{O}_n(\mathbf{x}) \\ & \sum_{i\in\mathcal{O}_n(\mathbf{x})} \alpha_{i,n} \le 1 \text{ for } n \in \mathcal{N}. \end{aligned} \tag{54}$$

We use $\alpha^*(\mathbf{x}) = \{\alpha^*_{i,n} | n \in \mathcal{N}, i \in \mathcal{O}_n(\mathbf{x})\}$ to denote the optimal solution of (54) under offloading decision $\mathbf{x}$. The optimal computing resource management decision $\alpha^*(\mathbf{x})$ is shown in Lemma 3.

Lemma 3: For any given $\mathbf{x}$, $\alpha^*(\mathbf{x})$ is as follows:

$$\alpha^*_{i,n} = \frac{\sqrt{\frac{f_i}{\delta_{i,n}}}}{\sum_{j \in \mathcal{O}_n(\mathbf{x})} \sqrt{\frac{f_j}{\delta_{j,n}}}} \text{ for } n \in \mathcal{N}, i \in \mathcal{O}_n(\mathbf{x}). \tag{55}$$

The proof of Lemma 3 is omitted due to space limitations. The main idea of the proof is to exploit the KKT conditions. Substituting α^* into (8) and changing the order in the double sum, the optimal processing latency under offloading decision $(\mathbf{x}, \mathbf{y})$ is equal to

$$\begin{aligned} T^P &= \sum_{i \in \mathcal{I}} \sum_{n \in \mathcal{N}} x_{i,n} \frac{f_i}{F_n \cdot \alpha^*_{i,n}} + \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{M}} y_{i,m} t_i \\ &= \sum_{i \in \mathcal{I}} \sum_{n \in \mathcal{N}} x_{i,n} \frac{\sqrt{f_i} \sum_{j \in \mathcal{O}_n(\mathbf{x})} \sqrt{f_j}}{F_n} + \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{M}} y_{i,m} t_i \\ &= \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \frac{x_{i,n} x_{j,n}}{F_n} \sqrt{\frac{f_i f_j}{\delta_{i,n} \delta_{j,n}}} + \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{M}} y_{i,m} t_{i,m}. \end{aligned} \tag{56}$$

That is, by substituting α^* into JOM, JOM is equivalent to P2 as follows:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & \sum_{n \in \mathcal{N}} \frac{1}{F_n} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} x_{i,n} x_{j,n} \sqrt{\frac{f_i f_j}{\delta_{i,n} \delta_{j,n}}} + \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{M}} y_{i,m} t_{i,m} \\ \text{s.t.} \quad & \sum_{i \in \mathcal{O}_m(\mathbf{y})} a_{i,m,l} \le A_{m,l} \text{ for } m \in \mathcal{M}, l \in \mathcal{L} \\ & \sum_{n \in \mathcal{N}} x_{i,n} + \sum_{m \in \mathcal{M}} y_{i,m} = 1 \text{ for } i \in \mathcal{I} \\ & x_{i,n}, y_{i,m} \in \{0,1\} \text{ for } i \in \mathcal{I}, n \in \mathcal{N}, m \in \mathcal{M}. \end{aligned} \tag{P2}$$

We use $(\mathbf{x}^*, \mathbf{y}^*)$ to denote the optimal solution of P2. Accordingly, $(\mathbf{x}^*, \mathbf{y}^*, \alpha^*(\mathbf{x}^*))$ is the optimal solution of JOM. Since we have the optimal solution $\alpha^*(\mathbf{x})$ under any given $(\mathbf{x}, \mathbf{y})$, we then focus on choosing $(\mathbf{x}, \mathbf{y})$, i.e., solving P2.

B. Semidefinite Programming Relaxation for Task Offloading

Despite eliminating variable α, P2 remains NP-hard, and no polynomial-time approximation algorithm exists for it. The proof is akin to that of Theorem 6, and we omit it. Semidefinite relaxation for P2 provides a practical method to find a feasible solution close to the optimal one in polynomial time. Consequently, we concentrate on developing a semidefinite programming (SDP) relaxation approach for P2.

First, we restructure P2 into the standard binary quadratic programming (BQP) form. Let u be the column vector of all variables of P2, i.e., $\mathbf{x}$ and $\mathbf{y}$. In particular, we have $u = (\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_N; \mathbf{y}_1; \mathbf{y}_2; \ldots; \mathbf{y}_M)$ where $\mathbf{x}_n = (x_{1,n}; \ldots; x_{I,n})$ and $\mathbf{y}_n = (y_{1,m}; y_{2,m}; \ldots; y_{I,m})$. $J = (M + N)I$ is the number of binary variables of P2. By introducing a new variable $U = u \cdot u^T$, we can reformulate P2 as:

$$\begin{aligned} \min_{u,U} \quad & \mathbf{Tr}(PU) + q^T u \\ \text{s.t.} \quad & h_i^T u = 1 \text{ for } i \in \mathcal{I} \\ & d_{m,l}^T u \le A_{m,l} \text{ for } m \in \mathcal{M}, l \in \mathcal{L} \\ & U = u \cdot u^T. \end{aligned} \tag{57}$$

P is a $J \times J$ matrix corresponding to the quadratic terms in the objective function of P2. $q, h_i, d_{m,l}$ are column vectors of length J. Since P can be written in the form of $\sum_i Q_i Q_i^T$, P is positive semidefinite, i.e., $P \succcurlyeq 0$. The only non-convex constraint in (57) is $U = u \cdot u^T$. By relaxing $U = u \cdot u^T$ to $U \succcurlyeq u \cdot u^T$ and adding constraint $u_j(1 - u_j) \ge 0$, (57) can be relaxed to a convex optimization problem as follows:

$$\begin{aligned} \min_{u,U} \quad & \mathbf{Tr}(PU) + q^T u \\ \text{s.t.} \quad & h_i^T u = 1 \text{ for } i \in \mathcal{I}, \\ & d_{m,l}^T u \le A_{m,l} \text{ for } m \in \mathcal{M}, l \in \mathcal{L}, \\ & \mathbf{diag}(U) \le u, \\ & \begin{bmatrix} U & u \\ u^T & 1 \end{bmatrix} \succcurlyeq 0. \end{aligned} \tag{58}$$

The last constraint in (58) holds if and only if $U \succcurlyeq 0$ and $U - u \cdot u^T \succcurlyeq 0$, which can be proved by exploiting the Schur complement. Let (u^*, U^*) be the optimal solution to (58). Since (58) is a relaxation of P2, the optimal objective value of (58) is a natural lower bound of the minimum value of P2.

In what follows, we provide a physical interpretation of the semidefinite relaxation in (58). Let v be a J-dimensional joint normal random vector. Let μ and Σ be the mean and covariance matrix of v, respectively. That is, v is drawn from distribution $\mathcal{N}(\mu, \Sigma)$. Then, from [35], if constraint $\mathbf{diag}(U) \le u$ is not in (58), $\mu = u^*$ and $\Sigma = U^* - \mu \cdot \mu^T$ minimizes the following problem:

$$\begin{aligned} \min \quad & \mathbb{E}\left[v^T P v + q^T v\right] \\ \text{s.t.} \quad & \mathbb{E}\left[h_i^T v\right] = 1 \text{ for } i \in \mathcal{I} \\ & \mathbb{E}\left[d_{m,l}^T v\right] \le A_{m,l} \text{ for } m \in \mathcal{M}, l \in \mathcal{L}. \end{aligned} \tag{59}$$

Intuitively, v drawn from $\mathcal{N}(\mu, \Sigma)$ under $\mu = u^*$ and $\Sigma = U^* - \mu \cdot \mu^T$ has a cost close to the optimal objective value of (58). We can draw a number of samples from $\mathcal{N}(\mu, \Sigma)$ with $\mu = u^*$ and $\Sigma = U^* - \mu \cdot \mu^T$, round each of them to a feasible decision, and choose the feasible decision with the lowest cost.

TABLE III
NOTATION TABLE

u	vector representing the collection of variables $\mathbf{x}$ and $\mathbf{y}$
Q	matrix related to quadratic terms in the objective function of P2
q	vector related to linear terms in the objective function of P2
h_i	vector related to the single selection constraint of WD i
$d_{m,l}$	vector of resource sizes related to the resource l of RA m

Algorithm 3: SDPR.

Input: $F_n, n \in \mathcal{N}$, $f_i, i \in \mathcal{I}$, $\mathbf{A}_m, m \in \mathcal{M}$, $\mathbf{a}_i, i \in \mathcal{I}$, $t_i, i \in \mathcal{I}$
Parameter: IterNum
Output: A feasible solution to P2: $(\mathbf{x}^\star, \mathbf{y}^\star)$

Calculate P, q, c_i for $i \in \mathcal{I}$, and d_m for $m \in \mathcal{M}$;
Solve SDP (58) to get u^* and U^*;
Initialize $(\mathbf{x}^\star, \mathbf{y}^\star) = \mathbf{round}(u^*)$;
Initialize $L^\star = T^P(\mathbf{x}^\star, \mathbf{y}^\star, \alpha^*(\mathbf{x}^\star, \mathbf{y}^\star))$;
Set $\mu = u^*$, $\Sigma = U^* - u^* \cdot u^{*T}$, and $l = 0$;
while $l <$ ***IterNum*** **do**
  Randomly sample $v^{(l)}$ from $\mathcal{N}(\mu, \Sigma)$;
  Calculate $(\mathbf{x}^{(l)}, \mathbf{y}^l) = \mathbf{round}(v^l)$;
  Calculate $L^{(l)} = T^P\left(\mathbf{x}^{(l)}, \mathbf{y}^{(l)}, \alpha^*(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\right)$;
  if $L^{(l)} < L^\star$ **then**
    $L^\star := L^{(l)}$;
    $(\mathbf{x}^\star, \mathbf{y}^\star) := (\mathbf{x}^{(l)}, \mathbf{y}^{(l)})$;
  end
  $l := l + 1$;
end
Return $(\mathbf{x}^\star, \mathbf{y}^\star)$;

Let *round* be the operator rounding u to a feasible solution to P2. A specific rounding algorithm is proposed in Section V-D. We then formally state the proposed algorithm, named SDPR, for P2 in Algorithm 3.

In step 2 of Algorithm 3 (*SDPR*), we can solve (58) by the interior point method with the time complexity of $O(J^7 \log(\epsilon^{-1}))$ [36], where ϵ represents the desired level of precision. The time complexity for solving (58) plus the time complexity of the rounding process is the time complexity of Algorithm 3. The time complexity of the rounding process is much lower than solving (58), which is validated by numerical simulations.

C. Linearly Constrained Quadratic Programming Relaxation for Task Offloading

In Section V-B, we relax P2 to an SDP problem. In particular, the nonconvex constraint $U = u \cdot u^T$ is substituted with the SDP constraint, i.e., the last constraint in (58). This relaxation, however, results in a longer numerical solution time. In this section, we relax P2 to a linearly constrained quadratic programming (QP) problem, which can be solved more efficiently. Similar to that in Section V-B, we use u to denote the set of all decision variables. Then, P2 can be rewritten as follows:

$$\begin{aligned} \min_{u} \quad & u^T P u + q^T u \\ \text{s.t.} \quad & h_i^T u = 1 \text{ for } i \in \mathcal{I} \\ & d_{m,l}^T u \le A_{m,l} \text{ for } m \in \mathcal{M}, l \in \mathcal{L} \\ & u \in \{0,1\}^J. \end{aligned} \tag{60}$$

The last constraint in (60) is non-convex. By relaxing it directly to $u \in \{0,1\}^J$, we have the following linearly constrained quadratic programming (QP) problem:

$$\begin{aligned} \min_{u} \quad & u^T P u + q^T u \\ \text{s.t.} \quad & h_i^T u = 1 \text{ for } i \in \mathcal{I} \\ & d_{m,l}^T u \le A_{m,l} \text{ for } m \in \mathcal{M}, l \in \mathcal{L} \\ & u \in [0,1]^J. \end{aligned} \tag{61}$$

Next, we introduce an algorithm similar to *SDPR*, which we name the Quadratic Programming Relaxation and Rounding

Algorithm 4: Round (Rounding Algorithm).

Input: $F_n, n \in \mathcal{N}$, $f_i, i \in \mathcal{I}$, $\mathbf{A}_m, m \in \mathcal{M}$, $\mathbf{a}_i, i \in \mathcal{I}$, $t_i, i \in \mathcal{I}$, $\mathbf{x}^c$, and $\mathbf{y}^c$
Parameter: IterNum
Output: A feasible solution to P2: $(\mathbf{x}^b, \mathbf{y}^b)$

$\mathcal{I}_F := \emptyset$, $\mathcal{I}_S := \mathcal{I}$, $l := 1$,$\mathbf{x}^b = 0$,$\mathbf{y}^b = 0$;
while $\exists j \in \mathcal{I}, m \in \mathcal{M}$ *such that* $\mathbf{a}_j \le \mathbf{A}_m$ **do**
  $i := \arg\max_{j \in \mathcal{I}_S} \max\{y_{j,1}^c, \cdots, y_{j,M}^c\}$;
  $m^* := \arg\max_{m \in \mathcal{M}} y_{i,m}^c$;
  if $\mathbf{a}_i \le \mathbf{A}_{m^*}$ **then**
    $y_{i,m}^b := 0$ for $m \in \mathcal{M} \setminus \{m^*\}$, $y_{i,m^*}^b := 1$;
    $\mathbf{A}_{m^*} := \mathbf{A}_{m^*} - \mathbf{a}_i$;
    $\mathbf{a}_i := \infty$;
    $y_{i,m}^c := 0$ for $i \in \mathcal{I}$;
    $\mathcal{I}_F = \mathcal{I}_F \cup \{i\}$ and $\mathcal{I}_S = \mathcal{I}_S \setminus \{i\}$;
  else
    $y_{i,m^*}^c := 0$;
  end
end
$x_{i,n}^b := 0$ for $i \in \mathcal{I}$ and $n \in \mathcal{N}$;
$x_{i,n^*}^b := 1$ for $i \in \mathcal{I}_S$ and $n^* = \arg\max_n x_{i,n}^c$;
for *iter* $\in \{1, 2, \cdots, IterNum\}$ **do**
  fag $= 0$;
  for $i \in \mathcal{I}_S$ **do**
    cost^{old} $= C_i(\mathbf{x}^b)$;
    cost^{new} $= \infty$;
    for $\bar{\mathbf{x}}_i \in \mathcal{X}_i$ **do**
      cost = $C_i(\bar{\mathbf{x}}_i, \mathbf{x}_{-i}^b)$;
      if *cost*<*cost*^{new} **then**
        $\mathbf{x}_i' = \bar{\mathbf{x}}_i$;
        cost^{new} = cost;
      end
    end
    if *cost*^{new} $<$ *cost*^{old} **then**
      flag = 1;
      break;
    end
  end
  if *flag == 1* **then**
    $\mathbf{x}_i^b = \mathbf{x}_i'$;
  else
    break;
  end
end

(abbreviated as *QPR*) algorithm. The main idea of *QPR* is to first solve (61), obtaining a continuous decision, and subsequently rounding this decision to its binary form using the *round* algorithm in Section V-D. For brevity, we will not delve into the detailed description of the *QPR* algorithm.

D. Rounding Algorithm Design

Next, we propose a rounding algorithm that rounds vector u of length J to a feasible solution $(\mathbf{x}, \mathbf{y})$ for P2, where u is the continuous solution of Problem (58) or (61), and $(\mathbf{x}, \mathbf{y})$ is binary. Before the rounding process, we unzip u to $(\mathbf{x}^c, \mathbf{y}^c)$. In particular, $x^c_{i,n} = v_{(n-1)\cdot I+i}$ for $i \in \mathcal{I}$ and $n \in \mathcal{N}$ and $y^c_{i,m} = v_{(N+m-1)\cdot I+i}$ for $i \in \mathcal{I}$ and $m \in \mathcal{M}$. The rounding algorithm is formally stated in Algorithm 4.

We first consider rounding $\mathbf{y}^c$ to a feasible $\mathbf{y}$, where $\mathbf{y}$ is an I-by-M binary matrix, $\mathbf{y}_m$ is the m^{th} column of $\mathbf{y}$. We sort WDs $\mathcal{I}$ by the value of $\max_m y^c_{i,m}$ in descending order. WDs in the order take turns to round $\{y^c_{i,m} \in [0,1]\}_{m\in\mathcal{M}}$ to $\{y_{i,m} \in \{0,1\}\}_{m\in\mathcal{M}}$. In particular, WD i set $y_{i,m} = 1$ if $m = \arg\max_{m'} y^c_{i,m'}$ and $\mathbf{a}_i$ is no greater then the available space of FPGA m, and $y_{i,m} = 0$ otherwise. Let $\mathcal{I}_F$ be the set of WDs offloading its tasks to $\mathcal{M}$, i.e., $i \in \mathcal{I}_F$ if and only if $\sum_{m\in\mathcal{M}} y_{i,m} = 1$. Then, since we have the constraint that $\sum_{m\in\mathcal{M}} y_{i,m} + \sum_{n\in\mathcal{N}} x_{i,n} = 1$, we have $x_{i,n} = 0$ for $i \in \mathcal{I}_F$. That is, we only need to consider $x_{i,n}$ for $i \in \mathcal{I}_S \triangleq \mathcal{I} \setminus \mathcal{I}_F$ in the following.

In what follows, we consider rounding $x^c_{i,n}$ to binary for $i \in \mathcal{I}_S$ and $n \in \mathcal{N}$. First, let $\mathbf{x}_i \triangleq \{x_{i,n} | n \in \mathcal{N}\}$ and define function $C_i(\mathbf{x})$ as follows:

$$C_i(\mathbf{x}) = \sum_{n\in\mathcal{N}} \frac{x_{i,n}}{F_n} \sqrt{\frac{f_i}{\delta_{i,n}}} \left(\sum_{j\in\mathcal{I}_S} x_{j,n} \sqrt{\frac{f_j}{\delta_{j,n}}} \right). \tag{62}$$

If $\mathbf{y}$ is fixed, P2 is equivalent to the optimization problem as follows:

$$\begin{aligned} \min_{\mathbf{x}_i, i\in\mathcal{I}_S} \quad & \sum_{i\in\mathcal{I}_S} C_i(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x}_i \in \mathcal{X}_i, \quad i \in \mathcal{I}_S \\ & \mathcal{X}_i \triangleq \left\{ \mathbf{x}_i \in \{0,1\}^N | \sum_{n\in\mathcal{N}} x_{i,n} = 1 \right\}. \end{aligned} \tag{63}$$

Problem (63) can be interpreted as a congestion game similar to P1. Then, we can choose $\mathbf{x}$ by an algorithm similar to Algorithm 1 as follows. At the beginning, each WD i sets $x_{i,n}$ to 1 if $n = \arg\max_{n'} x^c_{i,n'}$ and sets $x_{i,n}$ to 0 otherwise. Then, WDs in $\mathcal{I}_S$ take turns adjusting their decisions. To be more specific, if $x_{i,n} = 1$ and there exists $\bar{n} \in \mathcal{N}$ such that moving WD i from n to $\bar{n}$ can lower the latency of WD i given other WDs' decisions, WD i resets $x_{i,\bar{n}} = 1$ and $x_{i,n} = 0$ for $n \neq \bar{n}$. We can set a maximum number of iterations for WDs to adjust their decisions. In fact, if there is no limit for maximum iteration, it can be proved that the decision adjustment process for WDs in $\mathcal{I}_S$ will terminate after a finite number of iterations, as shown in the following theorem.

Theorem 7. If IterNum is set to ∞, Algorithm 4 terminates in finite steps, and $T^P(\mathbf{x}^b, \mathbf{y}^b) \leq 2.62 \cdot T^P(\mathbf{x}, \mathbf{y}^b)$ for any feasible $\mathbf{x}$.

The proof of Theorem 7 is omitted because the main idea of the proof is similar to that of Theorem 3. Note that Theorem 7 does not mean the proposed algorithm has an approximation ratio. Theorem 7 means that $T^P(\mathbf{x}^b, \mathbf{y}^b) \leq 2.62 \cdot T^P(\mathbf{x}, \mathbf{y}^b)$ for any $\mathbf{x}$, where $\mathbf{y}^b$ may not necessarily be the optimal $\mathbf{y}$. The time complexity of the rounding algorithm is polynomial. Since the proof is standard, we omit it.

VI. Numerical Evaluation

In this section, we assess the performance of the proposed algorithms across various scenarios. Our simulations are implemented using Python 3.10 on a DELL Alienware desktop equipped with 32 GB RAM and an AMD Ryzen7 2700X Eight-Core Processor running Windows 10 OS.

A. Simulation Setup

The computing capability of each server is measured by floating-point operations per second (FLOPS), and the capacity of each server i is set to the real-world FLOPS of EC2 instances from [37]. We consider a system with $N = 10$ general-purpose servers (servers) and $M = 5$ FPGAs. Task size of WDs $f_i, i \in \mathcal{I}$ are drawn from the real-world computing complexity (in terms of floating-point operations (FLOPs)) of 6 neural network models, the first six entries of Table V in [38]. Similar to [22], $\delta_{i,n}$ is drawn from $[0.5, 1]$. We set the numbers of CLBs and DSP slices are the two bottleneck resources, i.e., $L = 2$, and set $A_{m,1}, m \in \mathcal{M}$ and $A_{m,2}, m \in \mathcal{M}$ to real-world values of different types of FPGAs [39]. The number of CLBs required for implementing a fuzzy neural network varies with network size, where the numbers of CLBs required for implementing fuzzy neural networks with 60 input neurons, 10 to 18 neurons in the hidden layer, and three output neurons are in the range from 5000 to 6000 [40]. We drew the number of CLBs required of WDs from $[4000, 8000]$ and the number of DSP slices of WDs from $[20, 40]$. From [5], [6], $t_{i,m}, i \in \mathcal{I}, m \in \mathcal{M}$ are set to 10 to 60 times faster than the optimal average latencies of WDs on servers under the case that $I = 100$ and $M = 0$. The number of WDs varies in different simulations and will be specified in the following.

We assume WDs are located in a square area of $1\text{km} \times 1$ km, similar to that used in [9]. We divide the $1\text{km} \times 1$ km area into six $1/3\text{km} \times 1/2$ km subareas, and there are 6 5 G base stations (APs) located in the center of the 6 subareas. WD i and an AP k can communicate if the distance between them, denoted by $d_{i,k}$, is less than 0.5 km. For convenience, we set the bandwidth of APs to the maximum achievable speed rather than its true physical bandwidth. We randomly draw the uplink bandwidth $\overline{B}_k$ and the downlink bandwidth $\underline{B}_k$ from the set of 50–100 MHz. $\overline{c}_i, i \in [N]$ and $\underline{c}_i, i \in [N]$ are randomly drawn from $[0.4, 1]$ and $[0.2, 0.5]$ Megabits, respectively [10]. The parameter of

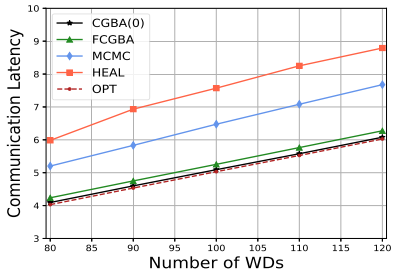


Fig. 2. Communication latency versus the number of WDs.

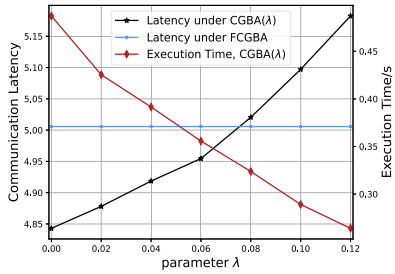


Fig. 3. Communication latency and execution time versus parameter λ.

bandwidth utilization ratio $\gamma_{i,k}$ corresponding to WD i and AP k is randomly chosen 15-50 bps/Hz [41].

B. Baselines

We use three baselines for comparison with *SDPR*. The first baseline, named by *MCMC*, which is similar to the algorithm proposed in [42]. *MCMC* is short for Monte Carlo Markov Chain technique and is similar to the simulated annealing technique. *MCMC* randomly chooses initial states $(\mathbf{x}, \mathbf{y})$ and $\mathbf{z}$ for JOM and JAM, respectively. Then, at each iteration, *MCMC* chooses a neighbor of the previous decision and moves to the neighbor with a probability related to the cost difference of the decisions. Details of *MCMC* can be found in Algorithm 1 of [42]. The second baseline is named *HEAL* (short for HEuristic ALgorithm), similar to the baseline used in [22]. For JOM, *HEAL* first chooses $\mathbf{y}$ by a greedy algorithm similar to the greedy algorithm [43] for the knapsack problem. In particular, *HEAL* sorts WDs in ascending order of $t_{i,m}$, and WDs in the order take turns to be placed on an FPGA until there is no sufficient space available. Then, *HEAL* chooses $\mathbf{x}$ for WDs that are not placed on $\mathcal{M}$. In particular, *HEAL* chooses the best server n for each i under the assumption that there is no other WD. *HEAL* chooses the optimal computing resource allocation decisions under the selected $(\mathbf{x}, \mathbf{y})$. For JAM, *HEAL* chooses the best uplink and downlink AP for each i under the assumption that there is no other WD and chooses the optimal communication resource allocation decisions under the selected $\mathbf{z}$. Moreover, we also compare the performance of our algorithms and that of the third baseline, i.e., the optimal solution by the commercial Gurobi solver.

C. Simulation Results

We first evaluate the performance of the proposed *CGBA*(λ) and *FCGBA* for P1. Fig. 2 shows the communication latency (in ms) of *CGBA*(0), *FCGBA*, *MCMC*, *HEAL* and the optimal latency under the number of WDs $I = \{80, 90, \ldots, 120\}$. The results shown in Fig. 2 represent the average values obtained from 10 independent simulations. As we can see from Fig. 2, the communication latency of *CGBA*(0) (Algorithm 1) is lower than that of *FCGBA*, *MCMC* and *HEAL* under all different settings of I. In addition, the communication latency of *CGBA*(0) is around 1.013 times the optimal latency on average, and the communication latency of *FCGBA* is around 1.05 times the optimal latency

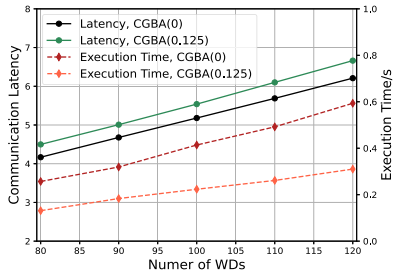


Fig. 4. Communication latency and execution time versus number of WDs.

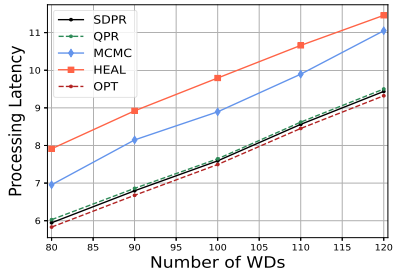


Fig. 5. Processing latency versus the number of WDs.

on average. As the number of WDs I increases, the average communication latencies under all the algorithms increase due to congestion. Then, we evaluate the performance of *CGBA*(λ) under different λ. Fig. 3 shows the performance of *CGBA*(λ) under $\lambda \in \{0, 0.02, \ldots, 0.12\}$ and that of *FCGBA*. There is a trade-off between the time complexity and the objective value (communication latency) of $CGBA(\lambda)$. As λ increases, the time complexity decreases, and the communication latency increases, which matches the statement in Theorem 4. From Theorem 4 and Theorem 5, the time complexity of *FCGBA* is equivalent to one iteration of $CGBA(\lambda)$. Although $CGBA(\lambda)$ has higher time complexity, it outperforms *FCGBA* in terms of objective value for $\lambda \in \{0, 0.02, 0.04, 0.06\}$. *FCGBA* outperforms $CGBA(\lambda)$ in both time complexity and approximation ratio for $\lambda \in \{0.08, 0.1, 0.12\}$. Fig. 4 shows the communication latency and time complexity of *CGBA*(0) and *CGBA*(0.125). As the system scale increases, both the latencies and time complexities under *CGBA*(0) and *CGBA*(0.125) increase linearly.

We then evaluate the performance of the proposed *SDPR* and *QPR* for P2. Fig. 5 shows the average processing latency (in ms) of *SDPR* and *QPR*, *MCMC*, *HEAL* and the optimal latency

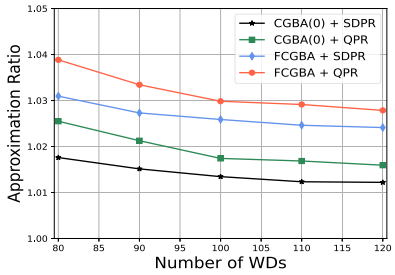


Fig. 6. Overall approximation ratio versus the number of WDs.

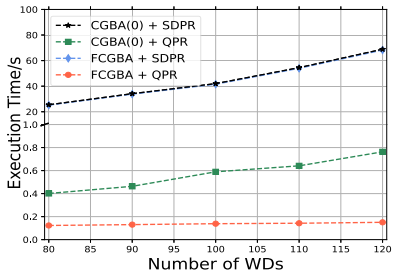


Fig. 7. Overall execution time versus the number of WDs

under the number of WDs $I = \{80, 90, \ldots, 120\}$. We set the maximum iteration number to be 20, i.e., the IterNum of *SDPR* is 20. In addition, since Theorem 7 holds, the maximum iteration number of the rounding algorithm (Algorithm 4) is set to ∞. The proposed *SDPR* and *QPR* outperform *MCMC* and *HEAL* under all the settings of I. From Fig. 5, the average ratio of the processing latencies under *SDPR* and *QPR* to the optimal value are around 1.017 and 1.021, respectively. As the number of WDs I increases, the average processing latency increases due to congestion. In addition, simulation results show that the execution time of *SDPR* and *QPR* is linear to I, demonstrating the good scalability of the proposed algorithms.

Next, we evaluate the total system latency and execution time of the proposed algorithms. Figs. 6 and 7 respectively illustrate the approximation ratio and execution time using various combinations of the proposed algorithms. Different combinations of the proposed algorithms yield varying approximation ratios and execution times, allowing users to select based on their precision and delay requirements, e.g., applications requiring low execution time can use the combination of *FCGBA* and *QPR* with the lowest time complexity, while applications requiring high precision can choose the combination of *CGBA*(0) and *SDPR* with the lowest approximation ratio. The execution time for the combinations of the proposed algorithms increases linearly with I, demonstrating the good scalability of our proposed approach.

VII. Conclusion

In this article, we have studied the joint task offloading, AP selection, and resource allocation problem (JOAM) in heterogeneous edge environments to minimize the overall system latency. We decomposed JOAM into two subproblems, namely JOM and JAM. We proposed a 2.62-approximation algorithm named *CGBA* for JAM. We demonstrated a trade-off between the approximation ratio and time complexity of *CGBA* and additionally offered a faster variant of the algorithm. In addition, we designed two algorithms, *SDPR* and *QPR*, for JOM based on convex relaxation and rounding. Simulation results have shown that the proposed algorithms outperform the popular baselines and are near-optimal. In particular, the average processing latency under *SDPR* is around 1.017 times the optimum, and the average communication latency under *CGBA* is around 1.013 times the optimum. In our future work, we aim to explore energy-aware online task offloading, AP selection, and resource allocation. Our emphasis will be on reducing system latency while adhering to energy consumption constraints, and on the development of algorithms optimized for swift decision-making.

References

[1] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[2] J. Shi, J. Du, Y. Shen, J. Wang, J. Yuan, and Z. Han, "DRL-based V2V computation offloading for blockchain-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3882–3897, Jul. 2023.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[4] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA-based neural network accelerator," 2017, *arXiv: 1712.08934*.

[5] Y. Shen, M. Ferdman, and P. Milder, "Maximizing CNN accelerator efficiency through resource partitioning," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit.*, 2017, pp. 535–547.

[6] P. Milder, "ESE507 advanced digital system design and generation," *ESE507 lecture notes*, 2021.

[7] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Enabling FPGAs in hyperscale data centers," in *Proc. IEEE 12th Int. Conf. Ubiquitous Intell. Comput. 12th Int. Conf. Autonomic Trusted Comput. IEEE 15th Int. Conf. Scalable Comput. Commun. Assoc. Workshops*, 2015, pp. 1078–1086.

[8] J. Weerasinghe, R. Polig, F. Abel, and C. Hagleitner, "Network-attached FPGAs for data center applications," in *Proc. Int. Conf. Field- Program. Technol.*, 2016, pp. 36–43.

[9] S. Jošilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2467–2475.

[10] S. Jošilo and G. Dán, "Joint wireless and edge computing resource management with dynamic network slice selection," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1865–1878, Aug. 2022.

[11] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.

[12] T. Liu, Y. Zhang, Y. Zhu, W. Tong, and Y. Yang, "Online computation offloading and resource scheduling in mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6649–6664, Apr. 2021.

[13] H. Sun, F. Zhou, and R. Q. Hu, "Joint offloading and computation energy efficiency maximization in a mobile edge computing system," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 3052–3056, Mar. 2019.

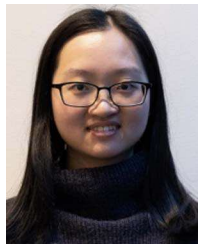
[14] A. Al-Shuwaili, O. Simeone, A. Bagheri, and G. Scutari, "Joint uplink/downlink optimization for backhaul-limited mobile cloud computing with user scheduling," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 4, pp. 787–802, Dec. 2017.

[15] Y. Liu, Y. Mao, X. Shang, Z. Liu, and Y. Yang, "Energy-aware online task offloading and resource allocation for mobile edge computing," in *Proc. 43rd Int. Conf. Distrib. Comput. Syst.*, 2023, pp. 339–349.

[16] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 175–190, Jan. 2023.

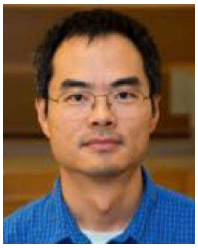
[17] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1405–1418, Jun. 2020.
[18] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware social cloud computing based on serverless computing model," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.
[19] P. A. Apostolopoulos, M. Torres, and E. E. Tsiropoulou, "Satisfaction-aware data offloading in surveillance systems," in *Proc. 14th Workshop Challenged Netw.*, New York, NY, USA, 2019, pp. 21–26. [Online]. Available: https://doi.org/10.1145/3349625.3355437
[20] S. Wang, J. Yang, and S. Bi, "Adaptive video streaming in multi-tier computing networks: Joint edge transcoding and client enhancement," *IEEE Trans. Mobile Comput.*, early access, Mar. 30, 2023, doi: 10.1109/TMC.2023.3263046.
[21] Y. Liu, Y. Mao, Z. Liu, and Y. Yang, "Deep learning-assisted online task offloading for latency minimization in heterogeneous mobile edge," *IEEE Trans. Mobile Comput.*, early access, Jun. 20, 2023, doi: 10.1109/TMC.2023.3285882.
[22] Y. Liu, X. Shang, and Y. Yang, "Joint SFC deployment and resource management in heterogeneous edge for latency minimization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 2131–2143, Aug. 2021.
[23] Y. Liu, Y. Mao, Z. Liu, F. Ye, and Y. Yang, "Joint task offloading and resource allocation in heterogeneous edge environments," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
[24] S. I. Venieris and C.-S. Bouganis, "f-CNNx: A toolflow for mapping multiple convolutional neural networks on FPGAs," in *Proc. 28th Int. Conf. Field Program. Log. Appl.*, 2018, pp. 381–3817.
[25] Y. Zha and J. Li, "Virtualizing FPGAs in the cloud," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2020, pp. 845–858.
[26] O. Knodel, P. Lehmann, and R. G. Spallek, "RC3E: Reconfigurable accelerators in data centres and their provision by adapted service models," in *Proc. IEEE 9th Int. Conf. Cloud Comput.*, 2016, pp. 19–26.
[27] P. Chanclou and e. Pizzinat, "Optical fiber solution for mobile fronthaul to achieve cloud radio access network," in *Proc. Future Netw. Mobile Summit*, 2013, pp. 1–11.
[28] S. Jošilo and G. Dán, "Decentralized scheduling for offloading of periodic tasks in mobile edge computing," in *Proc. IFIP Netw. Conf. Workshops*, 2018, pp. 1–9.
[29] R. K. James, K. P. Jacob, and S. Sasi, "Performance analysis of double digit decimal multiplier on various FPGA logic families," in *Proc. 5th Southern Conf. Program. Log.*, 2009, pp. 165–170.
[30] Wikipedia contributors, "5G NR frequency bands—Wikipedia, the free encyclopedia," 2023. Accessed: Nov. 27, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=5G_NR_frequency_bands&oldid=1185466352
[31] Nvidia, "Multi-process service (MPS)." Accessed: Nov. 25, 2023. [Online]. Available: https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf
[32] T. Harks, M. Klimm, and R. H. Möhring, "Characterizing the existence of potential functions in weighted congestion games," *Theory Comput. Syst.*, vol. 49, no. 1, pp. 46–70, 2011.
[33] B. Awerbuch, Y. Azar, A. Epstein, V. S. Mirrokni, and A. Skopalik, "Fast convergence to nearly optimal solutions in potential games," in *Proc. 9th ACM Conf. Electron. Commerce*, 2008, pp. 264–273.
[34] T. F. Gonzalez, *Handbook of Approximation Algorithms and Metaheuristics*. London, U.K.: Chapman and Hall/CRC, 2007.
[35] P. Wang, C. Shen, A. v. d. Hengel, and P. H. S. Torr, "Large-scale binary quadratic optimization using semidefinite relaxation and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 470–485, Mar. 2017.
[36] H. Jiang, T. Kathuria, Y. T. Lee, S. Padmanabhan, and Z. Song, "A faster interior point method for semidefinite programming," in *Proc. IEEE 61st Annu. Symp. Found. Comput. Sci.*, 2020, pp. 910–918.
[37] J. Emeras, S. Varrette, V. Plugaru, and P. Bouvry, "Amazon elastic compute cloud (EC2) versus in-house HPC platform: A cost analysis," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 456–468, Apr./Jun. 2019.
[38] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.
[39] Virtex, "Virtex-5 FPGA feature summary." Accessed: Nov. 25, 2023. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ds100
[40] S. R. Chowdhury and H. Saha, "Development of a FPGA based fuzzy neural network system for early diagnosis of critical health condition of a patient," *Comput. Biol. Med.*, vol. 40, no. 2, pp. 190–200, 2010.
[41] Y. Huo, X. Dong, and W. Xu, "5G cellular user equipment: From theory to practical hardware design," *IEEE Access*, vol. 5, pp. 13992–14010, 2017.
[42] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2076–2085.
[43] G. Dantzig, 26. *Discrete-Variable Extremum Problems*. Princeton, NJ, USA: Princeton Univ. Press, 2016.

Yu Liu (Graduate Student Member, IEEE) received the BEng degree in telecommunication engineering from Xidian University, Xi'an, China. He is currently working the PhD degree with the Department of Electrical and Computer Engineering, Stony Brook University. His research interests encompass edge computing and networks, low-earth orbit satellite networks, and distributed quantum computing. His primary focus is on service function placement and resource management in edge environments, as well as ensuring the reliability of service functions.



Yingling Mao (Graduate Student Member, IEEE) received the BS degree in mathematics and applied mathematics from Zhiyuan College, Shanghai Jiao Tong University, Shanghai, China, in 2018. She is currently working toward the PhD degree with the Department of Electrical and Computer Engineering, Stony Brook University. Her research interests include network function virtualization, software-defined networks, and cloud computing.

Zhenhua Liu received the BE degree in measurement control with honor from Tsinghua University, in 2006, the BS degree in economics from Peking University, in 2009, the MS degree in computer science technology with honor from Tsinghua University, in 2009, and the PhD degree in computer science from the California Institute of Technology, where he was co-advised by Dr. Adam Wierman and Prof. Steven Low. He is currently assistant professor with the Department of Applied Mathematics and Statistics, also affiliated with the Department of Computer Science and Smart Energy Technology Cluster, since August 2014. During the year 2014-2015, he is on leave for the ITRI-Rosenfeld Fellowship in the Energy and Environmental Technology Division, Lawrence Berkeley National Laboratory.



Fan Ye (Senior Member, IEEE) received the BE and MS degrees from Tsinghua University, Beijing, China, and the PhD degree from the Computer Science Department, The University of California at Los Angeles, Los Angeles, CA, USA. He is an assistant professor with the ECE Department, Stony Brook University, Stony Brook, NY, USA. He has published more than 60 peer-reviewed papers that have received more than 8000 citations according to Google Scholar. He has 21 granted/pending U.S. and international patents/applications. He was the co-chair of the Mobile Computing Professional Interests Community, IBM Watson for two years. His current research interests include mobile sensing platforms, systems and applications, Internet of Things, indoor location sensing, wireless, and sensor networks. He received IBM Research Division Award, five Invention Achievement Plateau Awards, and the Best Paper Award for International Conference on Parallel Computing 2008.

Yuanyuan Yang (Fellow, IEEE) received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a SUNY distinguished professor of computer engineering and computer science with Stony Brook University, New York, and is currently on leave with the National Science Foundation as a program director. Her research interests include edge computing, data center networks, cloud computing and wireless networks. She has published more than 460 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She is currently the editor-in-chief for *IEEE Transactions on Cloud Computing* and an associate editor for *IEEE Transactions on Parallel and Distributed Systems* and *ACM Computing Surveys*. She has served as an associate editor-in-chief for *IEEE Transactions on Cloud Computing*, associate editor-in-chief and associated editor for *IEEE Transactions on Computers*, and associate editor for *IEEE Transactions on Parallel and Distributed Systems*. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences.