Priority-Based Offloading Optimization in Cloud-Edge Collaborative Computing

Zhenli He, Yanan Xu, Mingxiong Zhao, Wei Zhou, and Keqin Li, *Fellow, IEEE*

***Abstract*—As an emerging computing paradigm, cloud-edge collaborative computing (CECC) combines computing resources at the back-end and the edge of the network to provide more flexible service delivery, thus striking a good balance between abundant computing resources and high responsiveness. However, mobile devices (MDs) must make strategic offloading decisions in such an environment. Although existing research has made remarkable progress in computation offloading strategies, most works ignore multi-priority settings in complex application scenarios. In this article, we focus on the impact of multi-priority settings and mixed queue disciplines on offloading decisions in CECC. First, we utilize queueing models to characterize all computing nodes in the environment and establish mathematical models to describe the considered scenario. Second, we formulate offloading decisions of the target MD into three multi-variable optimization problems to investigate the cost-performance tradeoff. Third, we propose numerical algorithms based on the Karush-Kuhn-Tucke conditions to address these problems. Finally, we construct numerical examples, a comparative experiment, and a simulation experiment to demonstrate the effectiveness of our methods. Our work provides important insights into the optimization of computation offloading for MDs in complex application scenarios, which can help achieve a better cost-performance tradeoff in CECC.**

***Index Terms*—Cost-performance tradeoff, cloud-edge collaborative computing, offloading optimization, queue disciplines, task priorities.**

I. Introduction

A. Motivation

In RECENT years, the contradiction between service demand and resource supply has gradually intensified with the popularization of mobile devices (MDs, e.g., smartphones, wearable devices, and handheld computers) and the increasing demand for mobile services. Achieving a balance between supply and demand is challenging using only the cloud or edge computing paradigm. In the cloud computing paradigm, the centralized aggregation of mobile network traffic puts tremendous pressure on the network core, causing some services to fail to respond within the time required by users. In the edge computing paradigm, the limited resources of an edge server (a.k.a. edge node, EN) compared to a data center (DC) make it difficult to meet service demands anywhere, anytime [1].

To address this challenge, cloud-edge collaborative computing (CECC) emerges as the times require [2]. In a CECC environment, MDs can offload computation-intensive and power-hungry tasks to either DCs or ENs based on system utilization, task characteristics, etc. Such an approach enables the computing platform to fully leverage the computing resources of the network back-end and edge to provide more flexible service delivery with a tradeoff between abundant computing resources and high responsiveness. It also means that MDs in such environments must make strategic offload decisions to achieve specific optimization objectives, such as reducing energy consumption or task response time.

In this context, computation offloading strategies in CECC environments has recently attracted much attention. There are mainly two optimization objectives for offloading decisions: performance and cost, where the performance is usually measured in terms of latency (e.g., response latency and communication latency), and the cost metric refers to energy consumption and monetary cost [1]. While performance and cost are critical for MDs, these two metrics are often contradictory and, in many cases, need to make a tradeoff between them. There are typically three ways to tradeoff performance and cost: performance optimization with cost constraint, cost optimization with performance constraint, and joint optimization of performance and cost (e.g., minimizing the weighted sum of cost and performance) [3], [4].

Although many researchers have conducted exciting research on these optimization objectives, there is still an overlooked problem: computing nodes in some complex scenarios often require different service rules to accommodate the varying urgency of heterogeneous tasks [5]. For example, in scenarios involving vehicles or unmanned aerial vehicles (UAVs) as ENs for offloading [6], [7], the most urgent tasks are related to flight or vehicle control. These tasks need to take priority over computing tasks offloaded by other MDs to ensure the safety of flight or driving operations. In a CECC environment with a private cloud [8],

Manuscript received 23 February 2023; revised 7 July 2023; accepted 15 July 2023. Date of publication 18 July 2023; date of current version 13 December 2023. This work was supported in part by the Applied Basic Research Foundation of Yunnan Province under Grants 202301AT070194, 202201AT070156, 202201AT070203, and 202301AT070422, in part by the National Natural Science Foundation of China under Grants 62172151 and 62162067, in part by the Major Science and Technology Projects in Yunnan Province under Grants 202202AD080002 and 202202AE09002105. Recommended for acceptance by J. Kolodziej. *(Corresponding author: Mingxiong Zhao.)*

Zhenli He, Mingxiong Zhao, and Wei Zhou are with the Engineering Research Center of Cyberspace, School of Software, Yunnan University, Kunming 650091, China (e-mail: hezl@ynu.edu.cn; mx_zhao@ynu.edu.cn; zwei@ynu.edu.cn).

Yanan Xu is with the College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: bfg_xyn@163.com).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSC.2023.3296601, provided by the authors.

Digital Object Identifier 10.1109/TSC.2023.3296601

the DC gives preferential treatment to tasks offloaded within its institution and only serves tasks offloaded by other MDs when it is not occupied. Similarly, MDs may have non-offloadable urgent tasks, such as system-level tasks related to embedded systems. Additionally, tasks can be prioritized based on their specific characteristics and the offloading scenarios to better fulfill the diverse demands of heterogeneous tasks [9], [10]. It is important to recognize that high-priority tasks cannot be treated as mere background load. The execution of high-priority tasks not only consumes resources but also inevitably interferes with the execution of other tasks due to their interleaved nature.

In summary, in a CECC environment, the offloading strategy must balance performance and cost while considering support for different queue disciplines. Such a strategy will enable MDs to efficiently leverage the CECC resources to meet the growing demand for mobile services.

B. Our Contributions

This paper focuses on a CECC environment that accommodates multiple mixed priority tasks, where each computing node requires the support of different queue disciplines. This is a complex environment, as all computing nodes may have tasks of different priorities at the same time. We investigate and propose an offloading scheme for the MDs to achieve a tradeoff between performance and cost. To achieve this tradeoff, we formulate and solve three optimization problems, including minimizing average response time (ART) under a cost constraint, minimizing average monetary cost (AMC, including MD's energy consumption and service fees in computing and communication) under a performance constraint, and minimizing cost-performance ratio. Our contributions are summarized as follows:

- We formulate the system models of the MD, ENs, and the DC as M/G/1 and M/G/m queueing systems, respectively, and construct a set of mathematical models to characterize the CECC environment.
- We conduct rigorous mathematical analysis and derive the ART of the MD's offloadable tasks and the MD's AMC. Then, we formulate offloading decisions into three different multi-variable optimization problems based on different tradeoff criteria.
- We design a series of algorithms based on the Karush-Kuhn-Tucker (KKT) conditions [11] to solve the above optimization problems and obtain optimal offloading decisions.
- We construct several numerical examples, a comparative experiment, and a simulation experiment to show the effectiveness of the proposed solution.

Our work can provide important insights into the optimization of computation offloading for MDs in complex application scenarios, which can help achieve better cost-performance tradeoffs in CECC. The remainder of the paper is arranged as follows. Section II reviews the existing work related to our research. Section III presents system models for the CECC environment. Section IV derives the performance and cost metrics and then formulates the optimization problems. Sections V–VII describe our numerical methods for solving these three optimization problems. Sections VIII and IX provide numerical examples and comparative experiments to show the effectiveness of our methods. Section X summarizes this paper and identifies directions for future research.

II. Related Work

This section reviews existing work related to our research from the perspective of how to make a tradeoff between performance and cost. (Due to space limitations, a more comprehensive analysis of the current study can be found in Section 2 of the supplementary file, available online.)

Performance Optimization With Cost Constraint: The most common way to tradeoff two conflicting objectives is to constrain one of them in order to optimize the other. Since one of the original intentions of mobile edge computing (MEC) is to reduce latency, performance optimization occupies a large part of the existing research. Wang et al. [12] studied distributed offloading in wireless-powered MEC and addressed the problem of minimizing the average task completion latency with energy constraints based on deep reinforcement learning (DRL). In [13], the authors have addressed the joint optimization of offloading decision, resource and power allocation in a multi-user CECC environment. The main objective is to minimize the overall latency experienced by all MDs, encompassing both transmission and execution durations, while simultaneously ensuring compliance with energy consumption and execution latency constraints.

Cost Optimization With Performance Constraint: Energy consumption and execution cost are critical metrics for some MDs, such as UAVs or internet-of-thing (IoT) devices. As a result, many studies use cost as the primary optimization objective. Hu et al. [14] investigated offloading optimization and resource allocation in a MEC-enabled IoT network with multiple ENs and MDs equipped with energy harvesting (EH) components. To balance energy efficiency and service latency, the authors designed an online offloading scheme based on Lyapunov optimization and semi-definite programming. Ma et al. [15] researched the computation capacity configuration of the edge and tenancy strategy adjustment in a CECC environment with multiple mobile requests, an EN, and a DC. The authors modeled the EN and the DC as M/M/1 queueing systems and designed algorithms to obtain the optimal resource provisioning and cloud tenancy schemes to minimize the system cost with a given latency constraint.

Joint Optimization of Performance and Cost: There are several methods to jointly optimize performance and cost, such as reducing the weighted sum of cost and performance and reducing the ratio of cost-performance (i.e., the product of cost-time). Yao et al. [16] investigated blockchain-enabled offloading decision in CECC via reinforcement learning, including task offloading, resource management, and smart contracts, aiming at minimizing the weighted sum of latency and cost. In addition to the aforementioned literature regarding collaborative resource allocation and offloading optimization, several solutions have emerged to maximize the utility of MDs and cloud resources. These solutions involve the utilization of pre-signed resource trading contracts, which are based on comprehensive assessments of potential risks, analysis of historical statistics (e.g., dynamic variations in resource requirements and unstable network conditions), and multi-party negotiations [17], [18].

It is noteworthy that the queueing models employed in previous research significantly differ from those utilized in our study. Specifically, existing studies commonly adopt M/M/1 or M/M/m queueing models and often assume non-priority systems, such as the first-come-first-served (FCFS) discipline. To highlight the distinctive characteristics of this paper compared to the aforementioned studies, we outline the following unique features.

- We consider task types and mixed queue disciplines as crucial factors influencing offloading decisions in CECC, while acknowledging that the queue disciplines employed by different nodes may vary. The available queue disciplines include non-preemptive nonpriority queue discipline, as well as non-preemptive and preemptive priority queue disciplines.
- We utilize an M/G/1 priority queueing system to characterize MDs, while employing M/G/m priority queueing systems to characterize ENs and the DC. This choice of queueing models is more sophisticated and challenging, but it offers the advantage of accommodating task-related parameters, such as execution requirements and task execution times, that can follow arbitrary probability distributions. This enhanced flexibility in modeling allows for better applicability in real-world scenarios.

To the best of our knowledge, the modeling and analysis approach used in our study has not been explored in existing research investigating offload optimization in CECC.

III. Preliminaries

This section presents the necessary information regarding assumptions, notations, definitions, and models. (The summary of the symbols and their definitions can be found in Section 1 of the supplementary file, available online.)

A. The Cloud-Edge Collaborative Computing Environment

First, we introduce the CECC environment considered in this paper.

Assume that there are n multiserver ENs (denoted as EN_1, ..., EN_n) and a multiserver DC in the CECC environment to provide offloading services for computationally constrained MDs (represented by Fig. 1). These computing nodes (including MDs, ENs, and the DC) are typically heterogeneous in terms of computing power, load conditions (e.g., execution requirements and the number of preloaded tasks), and queue disciplines (i.e., service policies for different tasks). In such a scenario with both wired and wireless communication, the target MD must decide whether to perform the offloadable tasks locally or offload to ENs/DC for remote execution to tradeoff cost and performance when all computing nodes have different priority settings.

For offloading decisions, the MD needs to send offloadable tasks that are difficult to handle to ENs/DC via an appropriate strategy with the tradeoff between high performance and low cost: 1) local execution of computing-intensive tasks may result

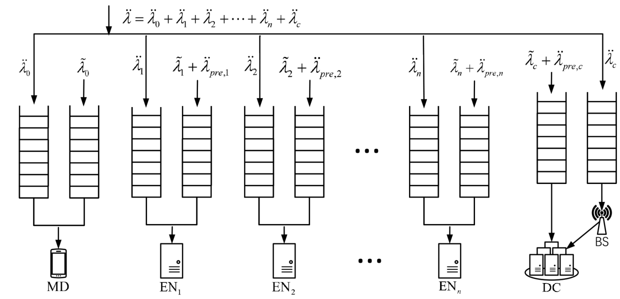

Fig. 1. A CECC environment with an MD, multiple ENs, and a DC.

in high energy consumption/latency issues; 2) remote processing on the DC can ensure low processing time, but there is a potential problem of high network delay; 3) remote processing on ENs is expected to ensure low network delay benefiting from the deployment location, but service demands cannot be fully guaranteed due to limited resources compared with the DC. Thus, in such a cloud-edge-end offloading scenario, finding the optimal offloading solution becomes more challenging.

There are two types of tasks in the environment, i.e., specific tasks and generic tasks, where all nodes have both types of tasks but with different definitions.

- The specific tasks on the MD refer to dedicated tasks that must be executed locally (i.e., non-offloadable tasks). In contrast, other tasks on the MD are generic tasks that can be executed locally or offloaded to some selected ENs or the DC (i.e., offloadable tasks).
- The specific tasks on ENs refer to critical tasks of the ENs and may have a higher priority. The generic tasks on ENs include computing tasks offloaded from the target MDs and other MDs.
- The specific tasks on the DC are preloaded tasks within its institution, which may have a higher priority than offloaded tasks from external institutions. Similarly, generic tasks on the DC refer to the computing tasks offloaded by all MDs.

In addition, computing nodes in CECC can apply any of the following three queue disciplines according to their requirements or characteristics:

- *Non-preemptive nonpriority queue discipline* (DS_1): All tasks in a queueing system are treated equally on an FCFS basis.
- *Non-preemptive priority queue discipline* (DS_2): Specific tasks have higher priority and are always scheduled before generic tasks without preemption.
- *Preemptive priority queue discipline* (DS_3): Specific tasks with high priority will be executed before generic tasks, where generic tasks in the service will be interrupted due to the arrival of specific tasks.

(Section 3 of the supplementary file provides detailed illustrations of these disciplines, available online.)

B. The MD Model

The target MD is modeled as an M/G/1 queueing system that can apply any of the three queue disciplines [19].

Assume that computing tasks generated by the MD conform to a Poisson stream with an arrival rate $\lambda = \tilde{\lambda}_0 + \ddot{\lambda}$ (measured in the number of tasks arriving per second), where $\tilde{\lambda}_0$ denotes arrival rate of specific tasks and $\ddot{\lambda}$ denotes arrival rate of generic tasks (i.e., offloadable tasks).

The Poisson stream of generic tasks can be further divided into $n+2$ substreams, that is, $\ddot{\lambda} = \ddot{\lambda}_0 + \ddot{\lambda}_1 + \cdots + \ddot{\lambda}_n + \ddot{\lambda}_c$, where the substream with arrival rate $\ddot{\lambda}_0$ is executed locally in the MD, the ith substream of generic tasks with arrival rate $\ddot{\lambda}_i$ is offloaded to EN$_i$ (where $1 \le i \le n$), and the substream with arrival rate $\ddot{\lambda}_c$ is offloaded to the DC. Let $\lambda_0 = \tilde{\lambda}_0 + \ddot{\lambda}_0$ denote the total arrival rate of tasks executed locally in the MD. Then, we have $\lambda = \lambda_0 + \sum_{i=1}^{n} \ddot{\lambda}_i + \ddot{\lambda}_c$. Note that the vector $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$ actually represents an offloading strategy of the MD in the CECC environment.

Let s_0 denote the execution speed of the MD (measured in billion instructions per second, BIPS). The execution requirements (measured in billion instructions, BI) of specific tasks generated by the MD are independently and identically distributed (i.i.d.) random variables (r.v.s) $\tilde{r}_0$ with mean $\overline{\tilde{r}_0}$ and second moment $\overline{\tilde{r}_0^2}$. The execution requirements of generic tasks generated by the MD are also i.i.d. r.v.s $\ddot{r}_0$ with mean $\overline{\ddot{r}_0}$ and second moment $\overline{\ddot{r}_0^2}$. The input data sizes (measured in million bits, Mb) of generic tasks, $\ddot{d}$, are i.i.d. r.v.s with mean $\overline{\ddot{d}}$ and second moment $\overline{\ddot{d}^2}$.

C. The EN Model

Each EN is modeled as an M/G/m queueing system that can apply any of the three queue disciplines [19].

Assume that in addition to accepting the task stream with arrival rate $\ddot{\lambda}_i$, there is also a task Poisson stream with arrival rate $\tilde{\lambda}_i + \ddot{\lambda}_{i,pre}$ preloaded to EN$_i$ and processed by EN$_i$. $\tilde{\lambda}_i$ and $\ddot{\lambda}_{i,pre}$ denote the arrival rates of preloaded specific tasks and generic tasks that other MDs have offloaded, respectively. Therefore, the total task arrival rate to be performed on EN$_i$ is calculated by $\lambda_i = \tilde{\lambda}_i + \ddot{\lambda}_{i,pre} + \ddot{\lambda}_i$, for all $1 \le i \le n$.

Let m_i denote the server size of EN$_i$ (i.e., EN$_i$ has m_i identical servers) and s_i denote the execution speed. The average channel capacity (i.e., average data transmission rate) for the MD to communicate with EN$_i$ is c_i (measured in million bits per second, Mbps). The execution requirements of specific tasks preloaded on EN$_i$, $\tilde{r}_i$, are i.i.d. r.v.s with mean $\overline{\tilde{r}_i}$ and second moment $\overline{\tilde{r}_i^2}$. The execution requirements of generic tasks preloaded on EN$_i$, $\ddot{r}_i$, are also i.i.d. r.v.s with mean $\overline{\ddot{r}_i}$ and second moment $\overline{\ddot{r}_i^2}$.

D. The DC Model

Similarly, we consider the DC as an M/G/m queueing system that applies any of the three queue disciplines [19].

Assume that there is a Poisson stream of tasks that are already preloaded to the DC with arrival rate $\tilde{\lambda}_c + \ddot{\lambda}_{c,pre}$, in which $\tilde{\lambda}_c$ and $\ddot{\lambda}_{c,pre}$ denote the arrival rates of preloaded specific tasks and generic tasks offloaded by other MDs, respectively. These streams are unrelated to the target MD. Therefore, the overall task arrival rate to be performed on the DC is $\lambda_c = \tilde{\lambda}_c + \ddot{\lambda}_{c,pre} + \ddot{\lambda}_c$.

Let m_c and s_c represent the server size and execution speed of the DC, respectively. Theoretically, we have $m_c \ge m_i$, for all $1 \le i \le n$. When the MD decides to offload tasks to the DC, these tasks will first be offloaded to a selected base station (BS) that will forward these tasks to the DC via the Metropolitan Area Network (MAN). Let c_b be the average channel capacity for the MD to communicate with the BS and t_p (measured in seconds) be the average propagation latency from the BS to the DC since the DC is located in the center of the core network and is geographically far away from MDs. The execution requirements of specific tasks preloaded on the DC, $\tilde{r}_c$, are i.i.d. r.v.s with mean $\overline{\tilde{r}_c}$ and second moment $\overline{\tilde{r}_c^2}$. The execution requirements of generic tasks preloaded on the DC, $\ddot{r}_c$, are also i.i.d. r.v.s with mean $\overline{\ddot{r}_c}$ and second moment $\overline{\ddot{r}_c^2}$.

To sum up, the heterogeneity among computing nodes is reflected in several key characteristics, including queue disciplines, execution speed (s_0, s_i, s_c), channel capacity (c_i, c_b), execution requirements $(\overline{\tilde{r}_0}, \overline{\tilde{r}_i}, \overline{\tilde{r}_c}, \overline{\ddot{r}_0}, \overline{\ddot{r}_i}, \overline{\ddot{r}_c})$, and preloaded tasks $(\tilde{\lambda}_0, \tilde{\lambda}_i, \ddot{\lambda}_{i,pre}, \tilde{\lambda}_c, \ddot{\lambda}_{c,pre})$, where $1 \le i \le n$.

E. Power Consumption Models

This section describes the MD's power consumption models, including computation power consumption for using computing resources and communication power consumption for transmitting data. Notice that the MD is assumed to not be equipped with the EH device and its power consumption for computation is derived based on the CMOS circuit power model [20] and related to the operating frequency, which is consistent with many studies on energy-efficient offloading optimization (see Section II).

1) Power Consumption Model for Computation: The computation power consumption (measured in Watts) of an MD mainly consists of dynamic power consumption and static power consumption [21], [22], where dynamic power consumption can be formulated as $P^{dy} = \xi s^{\alpha}$, where ξ and α are technology-dependent constants, and s denotes the processor execution speed [23], [24], [25]. Let $P_0^{dy} = \xi_0 s_0^{\alpha_0}$ and P_0^* be the MD's dynamic power consumption and the static power consumption, respectively. Then, the MD's average power consumption (APC) for computation can be obtained by $P_0^{cmp} = \rho_0 P_0^{dy} + P_0^* = \rho_0 \xi_0 s_0^{\alpha_0} + P_0^*$, where ρ_0 denotes the MD's server utilization, which is derived in Section IV-A.

2) Power Consumption Model for Communication: During the communication process, the MD still needs to consume some power. Based on Shannon theorem [26], the channel capacity c can be formulated as $c = B\log_2(1 + gP^{trs}/(BN))$, where B denotes the bandwidth of the channel (measured in MHz), g represents the channel gain (measured in dBm), P^{trs} denotes the average transmission power over the bandwidth, and N represents the noise power spectrum density (measured in dBm/Hz). Thus, we express the average channel capacity c_i for the MD to communicate with EN$_i$ by

$$c_i = B_i \log_2\left(1 + g_i P_i^{trs}/(B_i N_i)\right),$$

then we have $P_i^{trs} = B_i N_i (2^{c_i/B_i} - 1)/g_i$, where $1 \le i \le n$. The average communication latency is $\overline{\ddot{d}}/c_i$ for offloading one generic task from the MD to EN_i. Thus, the average energy consumption (measured in Joules) for communication of one generic task from the MD to EN_i is $P_i^{trs}(\overline{\ddot{d}}/c_i)$.

Similarly, we formulate the average channel capacity c_b for the MD offloading tasks to the DC via the BS as

$$c_b = B_b \log_2 \left(1 + g_b P_b^{trs}/(B_b N_b)\right),$$

and we have $P_b^{trs} = B_b N_b (2^{c_b/B_b} - 1)/g_b$. Since the average communication latency for offloading one generic task to the DC is $\overline{\ddot{d}}/c_b$, the average energy consumption for transmitting one generic task to the DC via the BS is $P_b^{trs}(\overline{\ddot{d}}/c_b)$.

IV. Problem Definitions

In this section, we first derive the performance and cost metrics and then formulate the offloading decisions of the target MD into three different multi-variable optimization problems to investigate the cost-performance tradeoff.

A. Average Response Time

This paper uses the ART of generic tasks as the performance metric, which includes processing latency, queueing latency, and transmission latency.

First, we derive the ART for generic tasks executed locally on the MD, denoted by $\overline{T_0}$. According to Section III-B, the processing latency of specific tasks is i.i.d. r.v.s with mean $\overline{\tilde{t}_0} = \overline{\tilde{r}_0}/s_0$ and second moment $\overline{\tilde{t}_0^2} = \overline{\tilde{r}_0^2}/s_0^2$, and the processing latency of generic tasks is i.i.d. r.v.s with mean $\overline{\ddot{t}_0} = \overline{\ddot{r}_0}/s_0$ and second moment $\overline{\ddot{t}_0^2} = \overline{\ddot{r}_0^2}/s_0^2$. Then, the average processing latency (APL) of tasks on the MD is $\overline{t_0} = (\tilde{\lambda}_0 \overline{\tilde{r}_0} + \ddot{\lambda}_0 \overline{\ddot{r}_0})/(\lambda_0 s_0)$ with second moment $\overline{t_0^2} = (\tilde{\lambda}_0 \overline{\tilde{r}_0^2} + \ddot{\lambda}_0 \overline{\ddot{r}_0^2})/(\lambda_0 s_0^2)$. The MD's server utilization is $\rho_0 = \lambda_0 \overline{t_0} = (\tilde{\lambda}_0 \overline{\tilde{r}_0} + \ddot{\lambda}_0 \overline{\ddot{r}_0})/s_0$.

As mentioned in Section III-A, nodes in CECC can apply any of the three queue disciplines (i.e., DS_1, DS_2, DS_3). For DS_1, the average queueing latency (AQL) and the ART of generic tasks on the MD are calculated by [19, p. 700]

$$\begin{cases} \overline{W_0} = \frac{\lambda_0 \overline{t_0^2}}{2(1-\rho_0)} = \frac{\tilde{\lambda}_0 \overline{\tilde{r}_0^2} + \ddot{\lambda}_0 \overline{\ddot{r}_0^2}}{2 s_0 (s_0 - \tilde{\lambda}_0 \overline{\tilde{r}_0} - \ddot{\lambda}_0 \overline{\ddot{r}_0})}, \\ \overline{T_0} = \overline{\ddot{t}_0} + \overline{W_0}. \end{cases}$$

For DS_2, the AQL and the ART of generic tasks on the MD are [19, p. 702]

$$\begin{cases} \overline{W_0} = \frac{\tilde{\lambda}_0 \overline{\tilde{r}_0^2} + \ddot{\lambda}_0 \overline{\ddot{r}_0^2}}{2 (s_0 - \tilde{\lambda}_0 \overline{\tilde{r}_0})(s_0 - \tilde{\lambda}_0 \overline{\tilde{r}_0} - \ddot{\lambda}_0 \overline{\ddot{r}_0})}, \\ \overline{T_0} = \overline{\ddot{t}_0} + \overline{W_0}. \end{cases}$$

For DS_3, the ART of generic tasks on the MD is [19, p. 704]

$$\overline{T_0} = \frac{2\overline{\ddot{r}_0} s_0 + \tilde{\lambda}_0 (\overline{\tilde{r}_0^2} - 2\overline{\tilde{r}_0}\overline{\ddot{r}_0}) + \ddot{\lambda}_0 (\overline{\ddot{r}_0^2} - 2\overline{\ddot{r}_0}^2)}{2(s_0 - \tilde{\lambda}_0 \overline{\tilde{r}_0})(s_0 - \tilde{\lambda}_0 \overline{\tilde{r}_0} - \ddot{\lambda}_0 \overline{\ddot{r}_0})}. \tag{1}$$

Since ENs and the DC are both regarded as M/G/m queueing models, we uniformly analyze the ART of generic tasks processed on an M/G/m priority queueing system S_j, where $j \in \{1, \ldots, n, c\}$. Specifically, if $j = i$, the index j denotes EN_i, where $1 \le i \le n$; otherwise, $j = c$ denotes the DC. Let $\overline{W_j}$ and $\overline{T_j}$ be the AQL and the ART of generic tasks processed on it, respectively. As discussed in Section III, the processing latency of tasks is i.i.d. r.v.s. On system S_j, let $\overline{\tilde{t}_j}$ be specific tasks' APL with second moment $\overline{\tilde{t}_j^2}$, $\overline{\ddot{t}_j}$ be generic tasks' APL with second moment $\overline{\ddot{t}_j^2}$, and $\overline{t_j}$ be the APL of all tasks with second moment $\overline{t_j^2}$. The server utilization is ρ_j.

For DS_1, the AQL of generic tasks on S_j is [27]

$$\overline{W_j} = \frac{\overline{t_j} \cdot p_{j,m_j}(1 + CV_j^2)}{2 m_j (1 - \rho_j)}, \tag{2}$$

where

$$\begin{cases} CV_j = \sqrt{\overline{t_j^2}/\overline{t_j}^2 - 1}, \\ p_{j,m_j} = \frac{(m_j \rho_j)^{m_j}}{m_j!(1-\rho_j)} \cdot p_{j,0}, \\ p_{j,0} = \left(\sum_{k=0}^{m_j - 1} \frac{(m_j \rho_j)^k}{k!} + \frac{(m_j \rho_j)^{m_j}}{m_j!(1-\rho_j)} \right)^{-1}. \end{cases}$$

For DS_2, the AQL of generic tasks on S_j is given by [28]

$$\overline{W_j} = \frac{\overline{t_j} \cdot p_{j,m_j}(1 + CV_j^2)}{2 m_j (1 - \tilde{\rho}_j)(1 - \rho_j)}, \tag{3}$$

where $\tilde{\rho}_j = \tilde{\lambda}_j \overline{\tilde{r}_j}/(s_j m_j)$. For DS_3, the ART of generic tasks on S_j is [29]

$$\overline{T_j} = (\lambda_j R_j - \tilde{\lambda}_j \tilde{R}_j)/(\ddot{\lambda}_{j,pre} + \ddot{\lambda}_j), \tag{4}$$

where

$$\begin{cases} R_j = \frac{1}{\mu_j} + \frac{p_{j,m_j}}{\rho_j} \left(\frac{\tilde{\lambda}_j \mathring{R}_{j,1} + (\ddot{\lambda}_{j,pre} + \ddot{\lambda}_j) \mathring{R}_{j,2}}{\lambda_j} - \frac{1}{m_j \mu_j} \right), \\ \tilde{R}_j = \overline{\tilde{t}_j} + \frac{\overline{\tilde{t}_j^2}}{2\overline{\tilde{t}_j}} \cdot p_{j,0}^* \cdot \frac{(m_j \tilde{\rho}_j)^{m_j}}{m_j!} \cdot \frac{1}{m_j (1 - \tilde{\rho}_j)^2}, \\ p_{j,0}^* = \left(\sum_{k=0}^{m_j - 1} \frac{(m_j \tilde{\rho}_j)^k}{k!} + \frac{(m_j \tilde{\rho}_j)^{m_j}}{m_j!(1-\tilde{\rho}_j)} \right)^{-1}, \\ \mathring{R}_{j,1} = \overline{\tilde{t}_j} + \tilde{\lambda}_j \overline{\tilde{t}_j^2}/(2(1 - \tilde{\rho}_j)), \\ \mu_j = \lambda_j / \left(\tilde{\lambda}_j \overline{\tilde{t}_j} + (\ddot{\lambda}_{j,pre} + \ddot{\lambda}_j) \overline{\ddot{t}_j} \right). \end{cases} \tag{5}$$

and

$$\mathring{R}_{j,2} = \begin{cases} \frac{1}{1-\tilde{\rho}_i} \left(\frac{\overline{\ddot{r}_0}}{s_i} + \frac{\overline{\ddot{d}}}{c_i} + \frac{\tilde{\lambda}_i \overline{\tilde{t}_i^2} + (\ddot{\lambda}_{i,pre} + \ddot{\lambda}_i) \overline{\ddot{t}_i^2}}{2(1-\rho_i)} \right), & j = i; \\ \frac{1}{1-\tilde{\rho}_c} \left(\frac{\overline{\ddot{r}_0}}{s_c} + \frac{\overline{\ddot{d}}}{c_b} + \frac{\tilde{\lambda}_c \overline{\tilde{t}_c^2} + (\ddot{\lambda}_{c,pre} + \ddot{\lambda}_c) \overline{\ddot{t}_c^2}}{2(1-\rho_c)} \right), & j = c. \end{cases}$$

According to Section III-C, the APL of preloaded specific tasks on EN_i is $\overline{\tilde{t}_i} = \overline{\tilde{r}_i}/s_i$ with second moment $\overline{\tilde{t}_i^2} = \overline{\tilde{r}_i^2}/s_i^2$. The APL of preloaded generic tasks on EN_i is $\overline{\ddot{r}_i}/s_i$ with second moment $\overline{\ddot{r}_i^2}/s_i^2$. The APL of the MD's generic tasks on EN_i is $\overline{\ddot{r}_0}/s_i + \overline{\ddot{d}}/c_i$ with second moment $\overline{\ddot{r}_0^2}/s_i^2 + 2\overline{\ddot{r}_0}\overline{\ddot{d}}/(s_i c_i) + \overline{\ddot{d}^2}/c_i^2$. Then, the APL of all generic tasks on EN_i is

$$\overline{\ddot{t}_i} = \frac{\ddot{\lambda}_i}{\ddot{\lambda}_{i,pre} + \ddot{\lambda}_i} \left(\frac{\overline{\ddot{r}_0}}{s_i} + \frac{\overline{\ddot{d}}}{c_i} \right) + \frac{\ddot{\lambda}_{i,pre}}{\ddot{\lambda}_{i,pre} + \ddot{\lambda}_i} \cdot \frac{\overline{\ddot{r}_i}}{s_i},$$

with second moment

$$\overline{\ddot{t}_i^2} = \frac{\ddot{\lambda}_i}{\ddot{\lambda}_{i,pre} + \ddot{\lambda}_i} \left(\frac{\overline{\ddot{r}_0^2}}{{s_i}^2} + \frac{\overline{\ddot{d}^2}}{{c_i}^2} + 2\frac{\overline{\ddot{r}_0}\overline{\ddot{d}}}{s_i c_i} \right) + \frac{\ddot{\lambda}_{i,pre}}{\ddot{\lambda}_{i,pre} + \ddot{\lambda}_i} \cdot \frac{\overline{\ddot{r}_i^2}}{{s_i}^2},$$

where $\ddot{\lambda}_i/(\ddot{\lambda}_{i,pre} + \ddot{\lambda}_i)$ and $\ddot{\lambda}_{i,pre}/(\ddot{\lambda}_{i,pre} + \ddot{\lambda}_i)$ are the percentages of generic tasks offloaded from the MD and preloaded generic tasks on EN$_i$, respectively. Thus, the APL of tasks on EN$_i$ is

$$\overline{t_i} = \frac{\tilde{\lambda}_i}{\lambda_i} \cdot \frac{\overline{\tilde{r}_i}}{s_i} + \frac{\ddot{\lambda}_i}{\lambda_i}\left(\frac{\overline{\ddot{r}_0}}{s_i} + \frac{\overline{\ddot{d}}}{c_i}\right) + \frac{\ddot{\lambda}_{i,pre}}{\lambda_i} \cdot \frac{\overline{\ddot{r}_i}}{s_i},$$

with second moment

$$\overline{t_i^2} = \frac{\tilde{\lambda}_i}{\lambda_i} \cdot \frac{\overline{\tilde{r}_i^2}}{{s_i}^2} + \frac{\ddot{\lambda}_i}{\lambda_i}\left(\frac{\overline{\ddot{r}_0^2}}{s_i^2} + \frac{\overline{\ddot{d}^2}}{{c_i}^2} + \frac{2\overline{\ddot{r}_0}\overline{\ddot{d}}}{s_i c_i}\right) + \frac{\ddot{\lambda}_{i,pre}}{\lambda_i} \cdot \frac{\overline{\ddot{r}_i^2}}{{s_i}^2},$$

where $\tilde{\lambda}_i/\lambda_i$ and $(\ddot{\lambda}_i + \ddot{\lambda}_i)/\lambda_i$ denote the percentages of specific tasks and generic tasks on EN$_i$, respectively. The server utilization of EN$_i$ is given by

$$\rho_i = \frac{\lambda_i\overline{t_i}}{m_i} = \frac{\tilde{\lambda}_i\overline{\tilde{r}_i}}{s_i m_i} + \frac{\ddot{\lambda}_i}{m_i}\left(\frac{\overline{\ddot{r}_0}}{s_i} + \frac{\overline{\ddot{d}}}{c_i}\right) + \frac{\ddot{\lambda}_{i,pre}\overline{\ddot{r}_i}}{s_i m_i}. \tag{6}$$

Based on (2), the ART of generic tasks offloaded from the MD on EN$_i$ for DS_1 is

$$\overline{T_i} = \frac{\overline{\ddot{r}_0}}{s_i} + \frac{\overline{\ddot{d}}}{c_i} + \frac{\overline{t_i} \cdot p_{i,m_i}(1 + CV_i^2)}{2m_i(1 - \rho_i)}. \tag{7}$$

Based on (3), the ART of the MD's generic tasks on EN$_i$ for DS_2 is

$$\overline{T_i} = \frac{\overline{\ddot{r}_0}}{s_i} + \frac{\overline{\ddot{d}}}{c_i} + \frac{\overline{t_i} \cdot p_{i,m_i}(1 + CV_i^2)}{2m_i(1 - \tilde{\rho}_i)(1 - \rho_i)}. \tag{8}$$

Based on (4), the ART of the MD's generic tasks on EN$_i$ for DS_3 is

$$\overline{T_i} = (\lambda_i R_i - \tilde{\lambda}_i\tilde{R}_i)/(\ddot{\lambda}_{i,pre} + \ddot{\lambda}_i). \tag{9}$$

According to Section III-D, the APL of specific tasks on the DC is $\overline{\tilde{t}_c} = \overline{\tilde{r}_c}/s_c$ with second moment $\overline{\tilde{t}_c^2} = \overline{\tilde{r}_c^2}/s_c^2$, the APL of preloaded generic tasks on the DC is $\overline{\ddot{r}_c}/s_c$ with second moment $\overline{\ddot{r}_c^2}/s_c^2$, and the APL of the MD's generic tasks is $\overline{\ddot{r}_0}/s_c + \overline{\ddot{d}}/c_b + t_p$ with second moment $\overline{\ddot{r}_0^2}/s_c^2 + \overline{\ddot{d}^2}/c_b^2 + t_p^2 + 2\overline{\ddot{r}_0}\overline{\ddot{d}}/(s_c c_b) + 2\overline{\ddot{r}_0}t_p/s_c + 2\overline{\ddot{d}}t_p/c_b$. Then, the APL of generic tasks is

$$\overline{\ddot{t}_c} = \frac{\ddot{\lambda}_c}{\ddot{\lambda}_{c,pre} + \ddot{\lambda}_c}\left(\frac{\overline{\ddot{r}_0}}{s_c} + \frac{\overline{\ddot{d}}}{c_b} + t_p\right) + \frac{\ddot{\lambda}_{c,pre}}{\ddot{\lambda}_{c,pre} + \ddot{\lambda}_c} \cdot \frac{\overline{\ddot{r}_c}}{s_c},$$

with second moment

$$\overline{\ddot{t}_c^2} = \frac{\ddot{\lambda}_c}{\ddot{\lambda}_{c,pre} + \ddot{\lambda}_c}\left(\frac{\overline{\ddot{r}_0^2}}{{s_c}^2} + \frac{\overline{\ddot{d}^2}}{{c_b}^2} + {t_p}^2 + 2\frac{\overline{\ddot{r}_0}\overline{\ddot{d}}}{s_c c_b} + 2\frac{\overline{\ddot{r}_0}t_p}{s_c} + 2\frac{\overline{\ddot{d}}t_p}{c_b}\right) + \frac{\ddot{\lambda}_{c,pre}}{\ddot{\lambda}_{c,pre} + \ddot{\lambda}_c} \cdot \frac{\overline{\ddot{r}_c^2}}{{s_c}^2},$$

where $\ddot{\lambda}_c/(\ddot{\lambda}_{c,pre} + \ddot{\lambda}_c)$ and $\ddot{\lambda}_{c,pre}/(\ddot{\lambda}_{c,pre} + \ddot{\lambda}_c)$ are the percentages of generic tasks offloaded from the MD and preloaded generic tasks on the DC, respectively. Thus, the APL of tasks on the DC is given by

$$\overline{t_c} = \frac{\tilde{\lambda}_c}{\lambda_c} \cdot \frac{\overline{\tilde{r}_c}}{s_c} + \frac{\ddot{\lambda}_c}{\lambda_c}\left(\frac{\overline{\ddot{r}_0}}{s_c} + \frac{\overline{\ddot{d}}}{c_c} + t_p\right) + \frac{\ddot{\lambda}_{c,pre}}{\lambda_c} \cdot \frac{\overline{\ddot{r}_c}}{s_c},$$

with second moment

$$\overline{t_c^2} = \frac{\tilde{\lambda}_c}{\lambda_c} \cdot \frac{\overline{\tilde{r}_c^2}}{{s_c}^2} + \frac{\ddot{\lambda}_c}{\lambda_c}\left(\frac{\overline{\ddot{r}_0^2}}{{s_c}^2} + \frac{\overline{\ddot{d}^2}}{{c_b}^2} + {t_p}^2 + 2\frac{\overline{\ddot{r}_0}\overline{\ddot{d}}}{s_c c_b} + 2\frac{\overline{\ddot{r}_0}t_p}{s_c} + 2\frac{\overline{\ddot{d}}t_p}{c_b}\right) + \frac{\ddot{\lambda}_{c,pre}}{\lambda_c} \cdot \frac{\overline{\ddot{r}_c^2}}{{s_c}^2},$$

where $\tilde{\lambda}_c/\lambda_c$ and $(\ddot{\lambda}_c + \ddot{\lambda}_c)/\lambda_c$ denote the percentages of specific tasks and generic tasks on the DC, respectively. Again, the server utilization of the DC is

$$\rho_c = \frac{\lambda_c\overline{t_c}}{m_c} = \frac{\tilde{\lambda}_c\overline{\tilde{r}_c}}{s_c m_c} + \frac{\ddot{\lambda}_c}{m_c}\left(\frac{\overline{\ddot{r}_0}}{s_c} + \frac{\overline{\ddot{d}}}{c_b} + t_p\right) + \frac{\ddot{\lambda}_{c,pre}\overline{\ddot{r}_c}}{s_c m_c}. \tag{10}$$

Based on (2), the ART of MD's generic tasks on the DC for DS_1 is

$$\overline{T_c} = \frac{\overline{\ddot{r}_0}}{s_c} + \frac{\overline{\ddot{d}}}{c_b} + t_p + \frac{\overline{t_c} \cdot p_{c,m_c}(1 + CV_c^2)}{2m_c(1 - \rho_c)}, \tag{11}$$

Based on (3), the ART of the MD's generic tasks on the DC for DS_2 is

$$\overline{T_c} = \frac{\overline{\ddot{r}_0}}{s_c} + \frac{\overline{\ddot{d}}}{c_b} + t_p + \frac{\overline{t_c} \cdot p_{c,m_c}(1 + CV_c^2)}{2m_c(1 - \tilde{\rho}_c)(1 - \rho_c)}, \tag{12}$$

Based on (4), the ART of the MD's generic tasks on the DC for DS_3 is

$$\overline{T_c} = (\lambda_c R_c - \tilde{\lambda}_c\tilde{R}_c)/(\ddot{\lambda}_{c,pre} + \ddot{\lambda}_c). \tag{13}$$

Therefore, the ART of the MD's offloadable tasks can be determined by

$$\overline{T} = \frac{\ddot{\lambda}_0}{\ddot{\lambda}}\overline{T_0} + \frac{\ddot{\lambda}_1}{\ddot{\lambda}}\overline{T_1} + \frac{\ddot{\lambda}_2}{\ddot{\lambda}}\overline{T_2} + \cdots + \frac{\ddot{\lambda}_n}{\ddot{\lambda}}\overline{T_n} + \frac{\ddot{\lambda}_c}{\ddot{\lambda}}\overline{T_c}. \tag{14}$$

Besides, according to queueing theory, we have $\rho_0 < 1$, $\rho_i < 1$, for all $1 \le i \le n$, and $\rho_c < 1$.

B. Average Monetary Cost

The MD's monetary cost generally refers to following aspects: the cost for local execution on the MD and the cost for remote processing on ENs/DC. The cost for local execution mainly relates to the MD's energy consumption, while the cost for remote processing mainly involves three distinct costs and fees: 1) The channel service fee of the MD for data transmission; 2) The MD's energy consumption for data transmission; 3) The service fee of the MD for tasks processing remotely on ENs or the DC [30], [31].

Local Execution: Since the MD's APC for computation is P_0^{cmp}, the average energy consumption of processing one generic task on the MD is given by $P_0^{cmp}\overline{\ddot{r}_0}/s_0$. Let θ_0 denote the price of energy consumption per Joule (measured in CNY/J). Therefore, the MD's AMC for executing one generic task is given by $C_0^{loc} = \theta_0 \cdot P_0^{cmp}\overline{\ddot{r}_0}/s_0$.

Remote Processing: Offloading one generic task to EN$_i$ incurs the following costs: 1) The channel service fee $(\eta_i\overline{\ddot{d}})$ for the MD transmitting one generic task with mean input data sizes $\overline{\ddot{d}}$ to EN$_i$, where η_i denotes the price of transmitting per million bits of data (measured in CNY/Mb); 2) The cost of energy consumption $(\theta_0 \cdot P_i^{trs}\bar{\bar{d}}/c_i)$ for data transmission, where $P_i^{trs}\bar{\bar{d}}/c_i$ denotes the average energy consumption for transmitting one generic task to EN$_i$; 3) The service fee $(\pi_i\overline{\bar{r}_0})$ to process one generic task on EN$_i$, where π_i represents the price of executing per billion instructions (measured in CNY/BI). Thus, the MD's AMC to offload one generic task to EN$_i$ for remote processing is given by $C_i^{off} = \eta_i\bar{\bar{d}} + \theta_0 \cdot P_i^{trs}\bar{\bar{d}}/c_i + \pi_i\overline{\bar{r}_0}$, where $1 \le i \le n$.

Again, offloading one generic task to the DC also incurs following costs: 1) The channel service fee $(\eta_c\bar{\bar{d}})$ for the MD transmitting one generic task to the DC, where η_c represents the price of transmitting per million bits of data; 2) The cost of energy consumption $(\theta_0 \cdot P_b^{trs}\bar{\bar{d}}/c_b)$ for data transmission; 3) The service fee $(\pi_c\overline{\bar{r}_0})$ to process one generic task remotely on the DC, where π_c represents the price of executing per billion instructions. Thus, the MD's AMC to offload one generic task to the DC for remote processing is given by $C_c^{off} = \eta_c\bar{\bar{d}} + \theta_0 \cdot P_b^{trs}\bar{\bar{d}}/c_b + \pi_c\overline{\bar{r}_0}$. Based on all the above analysis, we can calculate the AMC to perform the generic tasks of MD generation as

$$\overline{C} = \frac{\ddot{\lambda}_0}{\ddot{\lambda}}C_0^{loc} + \sum_{i=1}^{n}\frac{\ddot{\lambda}_i}{\ddot{\lambda}}C_i^{off} + \frac{\ddot{\lambda}_c}{\ddot{\lambda}}C_c^{off}. \tag{15}$$

C. Problem Definitions

This section formally describes three optimization problems to be solved in this paper.

Environment Conditions and Constraints: Given an MD specified by $s_0, \tilde{\lambda}_0, \ddot{\lambda}, \overline{r_0}, \overline{r_0^2}, \overline{\bar{r}_0}, \overline{\bar{r}_0^2}, \bar{\bar{d}}, \overline{\bar{d}^2}, \xi_0, \alpha_0, P_0^*, \theta_0$, and n ENs, where EN$_i$ is specified by $s_i, m_i, \tilde{\lambda}_i, \ddot{\lambda}_{i,pre}, \overline{\tilde{r}_i}, \overline{\tilde{r}_i^2}, \overline{\ddot{r}_i}, \overline{\ddot{r}_i^2}, c_i, B_i, g_i, N_i, \eta_i, \pi_i$, for all $1 \le i \le n$, and a DC specified by $s_c, m_c, \tilde{\lambda}_c, \ddot{\lambda}_{c,pre}, \overline{\tilde{r}_c}, \overline{\tilde{r}_c^2}, \overline{\ddot{r}_c}, \overline{\ddot{r}_c^2}, c_b, B_b, g_b, N_b, t_p, \eta_c, \pi_c$, queue disciplines DS_d for each node, where $d = \{1, 2, 3\}$, and following constraints: $\rho_0 < 1$, $\rho_i < 1$, $\rho_c < 1$, and $\ddot{\lambda}_0 + \sum_{i=1}^{n}\ddot{\lambda}_i + \ddot{\lambda}_c = \ddot{\lambda}$.

Minimizing Average Response Time Under Cost Constraint: Given the cost requirement C^*, environment conditions and constraints, find the optimal offloading decision $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$ that minimizes $\overline{T}$ and satisfies $\overline{C} \le C^*$.

Minimizing Average Monetary Cost Under Performance Constraint: Given the time requirement T^*, environment conditions and constraints, find the optimal offloading decision $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$ that minimizes $\overline{C}$ and satisfies $\overline{T} \le T^*$.

Minimizing the Cost-Performance Ratio: Given environment conditions and constraints, find the optimal offloading decision $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$ that minimizes $R = \overline{T} \cdot \overline{C}$.

V. Minimizing Average Response Time Under Cost Constraint

In this section, we propose a solution for minimizing the ART of MD's generic tasks under cost constraint.

A. Analysis

The problem described in Section IV-C, which aims to minimize the ART of generic tasks while satisfying the given inequality constraint, can be considered as a differentiable multi-variable optimization problem. In this context, the KKT conditions are commonly acknowledged as a standard method for analyzing the optimal solution's characteristics.

First, let's rewrite the cost constraint $\overline{C} \le C^*$ as

$$\begin{aligned} &G(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c) \\ &= \ddot{\lambda}_0 C_0^{loc} + \sum_{i=1}^{n}\ddot{\lambda}_i C_i^{off} + \ddot{\lambda}_c C_c^{off} - \ddot{\lambda}C^* \le 0. \end{aligned} \tag{16}$$

Then we can construct the Lagrange function as

$$\begin{aligned} L = &\overline{T}(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c) + \beta G(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c) \\ &+ \omega\left(\ddot{\lambda}_0 + \sum_{i=1}^{n}\ddot{\lambda}_i + \ddot{\lambda}_c - \ddot{\lambda}\right), \end{aligned} \tag{17}$$

where β and ω are Lagrange multipliers, and we have

$$\begin{cases} 0 = \frac{1}{\ddot{\lambda}}\left(\overline{T_0} + \frac{\partial \overline{T_0}}{\partial \ddot{\lambda}_0}\ddot{\lambda}_0\right) + \beta\left(C_0^{loc} + \frac{\partial C_0^{loc}}{\partial \ddot{\lambda}_0}\ddot{\lambda}_0\right) + \omega, \\ 0 = \frac{\partial \overline{T}}{\partial \ddot{\lambda}_i} + \beta\frac{\partial G}{\partial \ddot{\lambda}_i} + \omega = \frac{1}{\ddot{\lambda}}\left(\overline{T_i} + \frac{\partial \overline{T_i}}{\partial \ddot{\lambda}_i}\ddot{\lambda}_i\right) + \beta C_i^{off} + \omega, \\ 0 = \frac{\partial \overline{T}}{\partial \ddot{\lambda}_c} + \beta\frac{\partial G}{\partial \ddot{\lambda}_c} + \omega = \frac{1}{\ddot{\lambda}}\left(\overline{T_c} + \frac{\partial \overline{T_c}}{\partial \ddot{\lambda}_c}\ddot{\lambda}_c\right) + \beta C_c^{off} + \omega. \end{cases}$$

For simplicity, we set

$$\begin{cases} L'(\ddot{\lambda}_0, \omega, \beta) = \overline{T_0} + \frac{\partial \overline{T_0}}{\partial \ddot{\lambda}_0}\ddot{\lambda}_0 + \ddot{\lambda}\omega \\ \qquad\qquad + \ddot{\lambda}\beta\left(C_0^{loc} + \frac{\partial C_0^{loc}}{\partial \ddot{\lambda}_0}\ddot{\lambda}_0\right), & (18) \\ L'(\ddot{\lambda}_i, \omega, \beta) = \overline{T_i} + \frac{\partial \overline{T_i}}{\partial \ddot{\lambda}_i}\ddot{\lambda}_i + \ddot{\lambda}\beta C_i^{off} + \ddot{\lambda}\omega, 1 \le i \le n, & (19) \\ L'(\ddot{\lambda}_c, \omega, \beta) = \overline{T_c} + \frac{\partial \overline{T_c}}{\partial \ddot{\lambda}_c}\ddot{\lambda}_c + \ddot{\lambda}\beta C_c^{off} + \ddot{\lambda}\omega. & (20) \end{cases}$$

(The detailed derivation process of the first-order partial derivatives is given in Section 12 of the supplementary file, available online.) Basing the KKT conditions, we get

$$\begin{cases} L'(\ddot{\lambda}_0, \omega, \beta) = 0, & (21) \\ L'(\ddot{\lambda}_i, \omega, \beta) = 0, 1 \le i \le n, & (22) \\ L'(\ddot{\lambda}_c, \omega, \beta) = 0, & (23) \\ \omega < 0, & (24) \\ \beta G(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c) = 0, & (25) \\ \beta \ge 0, & (26) \\ G(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c) \le 0, & (27) \\ \ddot{\lambda}_0 + \ddot{\lambda}_1 + \ddot{\lambda}_2 + \cdots + \ddot{\lambda}_n + \ddot{\lambda}_c = \ddot{\lambda}. & (28) \end{cases}$$

From (25)∼(27), we get the following relationship:

$$\begin{cases} \beta = 0, G(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c) < 0; \\ \beta > 0, G(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c) = 0. \end{cases}$$

If $G < 0$ (i.e., $\overline{C} < C^*$ and $\beta = 0$), the values of $\ddot{\lambda}_0$, $\ddot{\lambda}_i$, and $\ddot{\lambda}_c$ only depend on ω, where $1 \le i \le n$; otherwise, β and ω jointly determine $\ddot{\lambda}_0$, $\ddot{\lambda}_i$, and $\ddot{\lambda}_c$.

Besides, there are other constraints mentioned in Section IV-C, i.e., $\rho_0 < 1$, $\rho_i < 1$, and $\rho_c < 1$. Thus, we set the initial range of $\ddot{\lambda}_0$ as $[0, \ddot{\lambda}_0^*)$, the initial range of $\ddot{\lambda}_i$ as $[0, \ddot{\lambda}_i^*)$, and the initial range of $\ddot{\lambda}_c$ as $[0, \ddot{\lambda}_c^*)$, where

$$\begin{cases} \ddot{\lambda}_0^* = (s_0 - \tilde{\lambda}_0 \overline{\tilde{r}_0}) / \overline{\ddot{r}_0}, \\ \ddot{\lambda}_i^* = \frac{m_i - (\tilde{\lambda}_i \overline{\tilde{r}_i} + \ddot{\lambda}_{i,pre} \overline{\ddot{r}_i})/s_i}{\overline{\ddot{r}_0}/s_i + \overline{\ddot{d}}/c_i}, \\ \ddot{\lambda}_c^* = \frac{m_c - (\tilde{\lambda}_c \overline{\tilde{r}_c} + \ddot{\lambda}_{c,pre} \overline{\ddot{r}_c})/s_c}{\overline{\ddot{r}_0}/s_c + \overline{\ddot{d}}/c_b + t_p}. \end{cases}$$

B. Our Solutions

It is difficult to solve these complex nonlinear equations directly, i.e., to obtain a closed-form solution from (21)$\sim$(23). By deriving and analyzing the above equations, we develop a two-stage method consisting of several numerical algorithms to solve the problem.

- *Stage I:* Assume that $G < 0$ (i.e., $\overline{C} < C^*$ and $\beta = 0$), and then adjust the value of ω to obtain an offloading strategy $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$ making (21)$\sim$(23) and the constraint (28) satisfied.
- *Stage II:* We check whether the offloading strategy obtained by *Stage I* makes $G < 0$ hold. If that condition is met, the problem is solved; otherwise, in addition to determining ω, we need to further adjust the value of β so that all the constraints are satisfied.

A Motivating Example: This example helps the reader better understand our algorithms, which are for illustrative purposes only. We consider a CECC environment composed of an MD, $n = 5$ ENs, and a DC, where parameter settings are shown below: $\tilde{\lambda}_0 = 1.0$, $\ddot{\lambda} = 15.0$, $\overline{\tilde{r}_0} = 0.5$, $\overline{\tilde{r}_0^2} = 0.4$, $\overline{\ddot{r}_0} = 1.5$, $\overline{\ddot{r}_0^2} = 3.0$, $\overline{\ddot{d}} = 1.0$, $\overline{\ddot{d}^2} = 1.5$, $s_0 = 1.3$, $\xi_0 = 1.5$, $\alpha_0 = 3.0$, $P_0^* = 2.0$, $\theta_0 = 2.44 * 10^{-4}$, $\tilde{\lambda}_i = 2.5 + 0.05(i-1)$, $\ddot{\lambda}_{i,pre} = 3.0 + 0.05(i-1)$, $\overline{\tilde{r}_i} = 1.0 + 0.05(i-1)$, $\overline{\tilde{r}_i^2} = 1.35\overline{\tilde{r}_i}^2$, $\overline{\ddot{r}_i} = 1.2 + 0.05(i-1)$, $\overline{\ddot{r}_i^2} = 1.5\overline{\ddot{r}_i}^2$, $m_i = 4$, $s_i = 2.5 + 0.1(i-1)$, $c_i = 10 + 0.5(i-1)$, $B_i = 3.0 + 0.1(i-1)$, $N_i = -174 - 0.1(i-1)$, $\eta_i = (0.5 + 0.05(i-1)) * 10^{-4}$, $\pi_i = (2 + 0.1(i-1)) * 10^{-10}$, for all $1 \le i \le n$, $\tilde{\lambda}_c = 6.0$, $\ddot{\lambda}_{c,pre} = 4.0$, $\overline{\tilde{r}_c} = 1.35$, $\overline{\tilde{r}_c^2} = 1.55\overline{\tilde{r}_c}^2$, $\overline{\ddot{r}_c} = 1.5$, $\overline{\ddot{r}_c^2} = 1.7\overline{\ddot{r}_c}^2$, $s_c = 3.5$, $m_c = 15$, $c_b = 11.0$, $B_b = 2.6$, $N_b = -174.0$, and $t_p = 0.4$, $\eta_c = 0.6 * 10^{-4}$, $\pi_c = 2.0 * 10^{-10}$ CNY/BI. In these examples, channel gains (i.e., $g_1, g_2, \ldots, g_n, g_b$) are assumed to be uniformly distributed in $[-50, -30]$ dBm.

1) Stage I: In the first stage, we develop five numerical algorithms to determine ω and $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$ that satisfy (21)$\sim$(23) and (28) under the condition that $\beta = 0$. Since the proposed algorithms will frequently use the bisection method, we define it in Algorithm 1 to avoid repetition.

Algorithm 1: Bisection.

Input: v, lb, ub, ϵ, and *criterion*.
Output: v.
1: **while** $ub - lb > \epsilon$ **do**
2: $v \leftarrow (lb + ub)/2$;
3: Calculate *criterion*;
4: **if** *criterion* **then**
5: $ub \leftarrow v$;
6: **else**
7: $lb \leftarrow v$;
8: **end if**
9: **end while**
10: **return** $v \leftarrow (lb + ub)/2$.

Algorithm 2: Obtain_$\ddot{\lambda}_0$.

Input: Environment conditions and constraints, ω, and β.
Output: $\ddot{\lambda}_0$.
1: //Check whether the MD is involved in offloading
2: **if** (The MD is not involved in offloading decision) **then**
3: **return** $\ddot{\lambda}_0 \leftarrow 0$.
4: **else**
5: $lb \leftarrow 0$, $ub \leftarrow \ddot{\lambda}_0^*$; //Obtain the initial interval of $\ddot{\lambda}_0$
6: // Obtain $L'(\ddot{\lambda}_0, \omega, \beta)$ based on (18);
7: $\ddot{\lambda}_0 \leftarrow$ *Bisection*$(\ddot{\lambda}_0, lb, ub, \epsilon, L'(\ddot{\lambda}_0, \omega, \beta) > 0)$;
8: **return** $\ddot{\lambda}_0$.
9: **end if**

Algorithm 3: Obtain_$\ddot{\lambda}_i$.

Input: Environment conditions and constraints, ω, and β.
Output: $\ddot{\lambda}_i$.
1: // Check whether EN$_i$ is involved in offloading
2: **if** (EN$_i$ is not involved in offloading decision) **then**
3: **return** $\ddot{\lambda}_i \leftarrow 0$.
4: **else**
5: $lb \leftarrow 0$, $ub \leftarrow \ddot{\lambda}_i^*$; //Obtain the initial interval of $\ddot{\lambda}_i$
6: // Obtain $L'(\ddot{\lambda}_i, \omega, \beta)$ based on (19);
7: $\ddot{\lambda}_i \leftarrow$ *Bisection*$(\ddot{\lambda}_i, lb, ub, \epsilon, L'(\ddot{\lambda}_i, \omega, \beta) > 0)$;
8: **return** $\ddot{\lambda}_i$.
9: **end if**

First, we find that if β and ω are given, $L'(\ddot{\lambda}_0, \omega, \beta)$ (18) can be viewed as an increasing function of $\ddot{\lambda}_0$. Thus, we use Algorithm 2 to search $\ddot{\lambda}_0$ within a certain interval making $L'(\ddot{\lambda}_0, \omega, \beta) = 0$ (lines 4–8). The initial interval of $\ddot{\lambda}_0$ is $[0, \ddot{\lambda}_0^*)$ (line 5). Note that Algorithm 2 will first check whether the MD is involved in the offloading decision, which will be discussed later. If the MD does not participate in the offloading decision, which means the MD does not process generic tasks locally, and we set $\ddot{\lambda}_0 = 0$ and return $\ddot{\lambda}_0$ (lines 2–4). (We set $\epsilon = 10^{-7}$.) (Due to space limitation, we move the changing trend plots of functions involved in Algorithms 2$\sim$6 to Section 7 of the supplementary file, available online, such as the changing trend of $L'(\ddot{\lambda}_0, \omega, \beta)$ with $\ddot{\lambda}_0$.)

Second, we find that once β and ω are given, $L'(\ddot{\lambda}_i, \omega, \beta)$ (19) can be viewed as an increasing function of $\ddot{\lambda}_i$. Accordingly, we use Algorithm 3 to search $\ddot{\lambda}_i$ within a certain interval such that $L'(\ddot{\lambda}_i, \omega, \beta) = 0$ (lines 4–8). The initial interval of $\ddot{\lambda}_i$ is $[0, \ddot{\lambda}_i^*)$ (line 5). Again, Algorithm 3 will first check whether EN$_i$ is involved in the offloading decision, which will be discussed later. (We set $\epsilon = 10^{-7}$.)

Third, we find that once β and ω are given, $L'(\ddot{\lambda}_c, \omega, \beta)$ (20) increases as $\ddot{\lambda}_c$ increases. Accordingly, we use Algorithm 4 to search $\ddot{\lambda}_c$ within a certain interval such that $L'(\ddot{\lambda}_c, \omega, \beta) = 0$

Algorithm 4: Obtain_$\ddot{\lambda}_c$.

Input: Environment conditions, ω, and β.
Output:$\ddot{\lambda}_c$.
1: // Check whether the DC is involved in offloading
2: **if** (The DC is not involved in offloading decision) **then**
3: **return** $\ddot{\lambda}_c \leftarrow 0$.
4: **else**
5: $lb \leftarrow 0$, $ub \leftarrow \ddot{\lambda}_c^*$; //Obtain the initial interval of $\ddot{\lambda}_c$
6: // Obtain $L'(\ddot{\lambda}_c, \omega, \beta)$ based on (20);
7: $\ddot{\lambda}_c \leftarrow$ *Bisection*$(\ddot{\lambda}_c, lb, ub, \epsilon, L'(\ddot{\lambda}_c, \omega, \beta) > 0)$;
8: **return** $\ddot{\lambda}_c$.
9: **end if**

Algorithm 5: Obtain_ω.

Input: Environment conditions and constraints, and β.
Output:$\omega, \ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c$.
1: //Determine the interval $[lb, ub]$ of ω
2: Calculate $\omega_1^*, \omega_2^*, \ldots, \omega_{n+2}^*$;
3: Calculate $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_{n+2}$;
4: Determine l such that $\hat{\lambda}_l < \ddot{\lambda} \leq \hat{\lambda}_{l+1}$;
5: $ub \leftarrow \omega_l^*$;
6: **if** $l < n+2$ **then**
7: //The first l nodes participate in offloading decision
8: $lb \leftarrow \omega_{l+1}^*$;
9: **else**
10: //All nodes participate in offloading decision
11: $lb \leftarrow \max\{\omega_0^{\min}, \max\{\omega_1^{\min}, \omega_2^{\min}, \ldots, \omega_n^{\min}\}, \omega_c^{\min}\}$;
12: **end if**
13: $\omega \leftarrow$ *Bisection*$(\omega, lb, ub, \epsilon, \ddot{\lambda}_0 + \sum_{i=1}^n \ddot{\lambda}_i + \ddot{\lambda}_c < \ddot{\lambda})$;
14: Call Algorithm 2 to obtain $\ddot{\lambda}_0$;
15: **for** $i \leftarrow 1$ to n **do**
16: Call Algorithm 3 to obtain $\ddot{\lambda}_i$;
17: **end for**
18: Call Algorithm 4 to obtain $\ddot{\lambda}_c$;
19: **return** ω and $\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c$.

(lines 4−8). The initial interval of $\ddot{\lambda}_c$ is $[0, \ddot{\lambda}_c^*)$ (line 5). Similarly, Algorithm 4 first checks whether the DC is involved in the offloading decision, which will be discussed later. (We set $\epsilon = 10^{-7}$.)

Consequently, given β and ω, we can obtain an offloading strategy $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$ with the above steps. Given the constraint (28), $\ddot{\lambda}_0 + \sum_{i=1}^n \ddot{\lambda}_i + \ddot{\lambda}_c$ obtained from Algorithms 2∼4 has to make (28) hold, which is a decreasing function of ω. Thus, we use Algorithm 5 to find ω satisfying (28) within a certain interval $[lb, ub]$, where we have to determine the initial interval of ω (lines 1−12) before searching ω. By rewriting (18)∼(20), we have

$$\begin{cases} -\ddot{\lambda}\omega = \overline{T_0} + (\partial \overline{T_0}/\partial \ddot{\lambda}_0) \cdot \ddot{\lambda}_0 + \ddot{\lambda}\beta \left(C_0^{loc} + \ddot{\lambda}_0 \frac{\partial C_0^{loc}}{\partial \ddot{\lambda}_0} \right); & (29) \\ -\ddot{\lambda}\omega = \overline{T_i} + (\partial \overline{T_i}/\partial \ddot{\lambda}_i) \cdot \ddot{\lambda}_i + \ddot{\lambda}\beta C_i^{off}, 1 \leq i \leq n; & (30) \\ -\ddot{\lambda}\omega = \overline{T_c} + (\partial \overline{T_c}/\partial \ddot{\lambda}_c) \cdot \ddot{\lambda}_c + \ddot{\lambda}\beta C_c^{off}. & (31) \end{cases}$$

Algorithm 6: Obtain_β.

Input: Environment conditions and constraints, and C^*.
Output: β.
1: $lb \leftarrow$ a small value, $ub \leftarrow$ a large value;
2: $\beta \leftarrow$ *Bisection*$(\beta, lb, ub, \epsilon, \overline{C} < C^*)$;
3: **return** β.

From (29), if β is fixed, ω decreases as $\ddot{\lambda}_0$ increases. That is, ω gets the maximum value $\omega_0^{\max}$ if $\ddot{\lambda}_0 = 0$ and ω gets the minimum value $\omega_0^{\min}$ if $\ddot{\lambda}_0 \approx \ddot{\lambda}_0^*$, i.e.,

$$\begin{cases} \omega_0^{\max} = -(\overline{T_0} + \ddot{\lambda}\beta C_0^{loc})/\ddot{\lambda}, \\ \omega_0^{\min} = -\left(\overline{T_0} + \frac{\partial \overline{T_0}}{\partial \ddot{\lambda}_0} \cdot \ddot{\lambda}_0^* + \ddot{\lambda}\beta \left(C_0^{loc} + \ddot{\lambda}_0^* \frac{\partial C_0^{loc}}{\partial \ddot{\lambda}_0} \right) \right) / \ddot{\lambda}. \end{cases}$$

From (30), if β is fixed, ω decreases as $\ddot{\lambda}_i$ increases, i.e.,

$$\begin{cases} \omega_i^{\max} = -(\overline{T_i} + \ddot{\lambda}\beta C_i^{off})/\ddot{\lambda}, & \text{if } \ddot{\lambda}_i = 0; \\ \omega_i^{\min} = -\left(\overline{T_i} + \ddot{\lambda}_i^* (\partial \overline{T_i}/\partial \ddot{\lambda}_i) + \ddot{\lambda}\beta C_i^{off} \right) / \ddot{\lambda}, & \text{if } \ddot{\lambda}_i \approx \ddot{\lambda}_i^*. \end{cases}$$

From (31), if β is fixed, ω decreases as $\ddot{\lambda}_c$ increases, i.e.

$$\begin{cases} \omega_c^{\max} = -(\overline{T_c} + \ddot{\lambda}\beta C_c^{off})/\ddot{\lambda}, & \text{if } \ddot{\lambda}_c = 0; \\ \omega_c^{\min} = -(\overline{T_c} + \ddot{\lambda}_c^* (\partial \overline{T_c}/\partial \ddot{\lambda}_c) + \ddot{\lambda}\beta C_c^{off})/\ddot{\lambda}, & \text{if } \ddot{\lambda}_c \approx \ddot{\lambda}_c^*. \end{cases}$$

Due to the heterogeneity between nodes, the $n+2$ nodes have different domains of ω. Since $\ddot{\lambda}_0 + \sum_{i=1}^n \ddot{\lambda}_i + \ddot{\lambda}_c$ is a decreasing function of ω, ω needs to be big enough when $\ddot{\lambda}$ is too small, which means that not all nodes are required to process generic tasks from the MD. As a result, for a given β, ω determines which nodes should be involved in the offloading decision. Accordingly, Algorithm 5 first calculates $\omega_0^{\max}$, $\omega_i^{\max}$, for all $1 \leq i \leq n$, and $\omega_c^{\max}$, and then arranges them in descending order (line 2).

Then, let $\hat{\lambda}_l = \sum_{k=1}^{l-1} \hat{\lambda}'_k$, where $\hat{\lambda}'_k$ is chosen such that $L'(\hat{\lambda}'_k, \omega_l^*, \beta) = 0$, for all $1 \leq l \leq n+2$ and $1 \leq k \leq l-1$ (line 3). Specifically, (1) if index k denotes the MD, we require $L'(\ddot{\lambda}_0, \omega_l^*, \beta) = 0$ and set $\hat{\lambda}'_k = \ddot{\lambda}_0$; (2) if k denotes EN_i, we require $L'(\ddot{\lambda}_i, \omega_l^*, \beta) = 0$ and set $\hat{\lambda}'_k = \ddot{\lambda}_i$, where $1 \leq i \leq n$; (3) if k denotes the DC, we require $L'(\ddot{\lambda}_c, \omega_l^*, \beta) = 0$ and set $\hat{\lambda}'_k = \ddot{\lambda}_c$. Clearly, we have $\hat{\lambda}_1 = 0$ and $\hat{\lambda}_{n+3} = \ddot{\lambda}_0^* + \sum_{i=1}^n \ddot{\lambda}_i^* + \ddot{\lambda}_c$ representing the maximum workload in environment. The node can be involved in the offloading decision only when $\ddot{\lambda} > \hat{\lambda}_l$ and $\omega < \omega_l^*$, which possibly satisfies (21)∼(23). That is, when $\hat{\lambda}_l < \ddot{\lambda} \leq \hat{\lambda}_{l+1}$, only the first l nodes are involved in the offloading decision, where $1 \leq l \leq n+2$. Note that when $\hat{\lambda}_{n+2} < \ddot{\lambda} \leq \hat{\lambda}_{n+3}$, all nodes are required to participate in the offloading decision. In Algorithm 5, we obtain $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_{n+2}$ (line 3). When $\hat{\lambda}_l < \ddot{\lambda} \leq \hat{\lambda}_{l+1}$, only the first l nodes are involved in the offloading decision, where $1 \leq l \leq n+2$ (lines 4−12). (We set $\epsilon = 10^{-10}$.)

2) Stage II: In this stage, we check whether the offloading strategy $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$ obtained by Algorithm 5 with $\beta = 0$ makes (27) hold (i.e., $\overline{C} \leq C^*$). If that condition is met, the problem is solved; otherwise, the adjustment of β is needed to satisfy (27). A crucial observation is that $\overline{C}$ decreases as β decreases. Therefore, we propose Algorithm 6 to find β

Algorithm 7: Minimize Average Response Time.

Input: Environment conditions and C^*.
Output: $\overline{T}$ and $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$.
1: // Stage *I*
2: $\beta \leftarrow 0$;
3: Call Algorithm 5 to obtain $\omega, \ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c$;
4: Calculate $\overline{C}$ by using (15);
5: // Stage *II*
6: **if** $\overline{C} > C^*$ **then**
7: Call Algorithm 6 to obtain β;
8: Call Algorithm 5 to obtain $\omega, \ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c$;
9: **end if**
10: Calculate $\overline{T}$ by using (14);
11: **return** $\overline{T}$ and $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ddot{\lambda}_2, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c)$.

satisfying the cost constraint. Note that in line 2, the criterion ($\overline{C} \le C^*$) in the bisection method is given by two steps: 1) call Algorithm 5 with β to obtain $\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c$; 2) calculate $\overline{C}$ based on (15). (We set $\epsilon = 10^{-7}$.)

Finally, Algorithm 7 illustrates the procedures to get the optimal offloading strategy of the MD that minimizes the ART of generic tasks under the AMC constraint.

VI. Minimizing Average Monetary Cost Under Performance Constraint

In this section, we propose a solution for minimizing the AMC under given performance. (Due to space limitation, this section is moved to Section 4 of the supplementary file, available online.)

VII. Minimizing the Cost-Performance Ratio

In this section, we propose a solution for minimizing the product of cost-time. (Due to space limitation, we move this section to Section 5 of the supplementary file, available online.)

In the supplementary file, available online, we also provide a detailed time-complexity analysis of all algorithms in Section 6.

VIII. Numerical Examples

This section provides five examples for the three optimization problems by calling the proposed algorithms to show the effectiveness of our methods, where the Python code for implementing algorithms is written by us from scratch. All experiments are performed on a computer with intel(R) Xeon(R) 8375 C CPU @ 2.90 GHz and NVIDIA GeForce RTX 3090Ti. It should be noted that parameter settings used in this paper refer to the parameters in Refs. [24], [30], [32].

Specifically, Examples 1 and 2 correspond to minimizing the ART with a cost constraint, Examples 3 and 4 correspond to minimizing the AMC with a time constraint, and Example 5 corresponds to minimizing the cost-performance ratio. For each numerical example, there is an MD, $n = 5$ ENs, and a DC. Additionally, the environment-related parameter settings are the same for these examples: $\overline{\tilde{r}_0} = 0.5, \overline{\tilde{r}_0^2} = 0.4, \overline{\ddot{d}} = 1.0, \overline{\ddot{d}^2} = 1.5$, $\xi_0 = 1.5, \alpha_0 = 3.0, \overline{\ddot{r}_c} = 1.5, \overline{\ddot{r}_c^2} = 1.7\overline{\ddot{r}_c}^2, m_c = 15, B_b = 2.6$, and $\pi_c = 2.0 * 10^{-10}$.

Example 1: The MD is given by $\tilde{\lambda}_0 = 1.0$, $\ddot{\lambda} = 10.0$, $\overline{\ddot{r}_0} = 1.5$, $\overline{\ddot{r}_0^2} = 3.0$, $s_0 = 1.3$, $P_0^* = 2.0$, $\theta_0 = 2.44 * 10^{-4}$. There are $n = 5$ ENs, where EN_i is given by $\tilde{\lambda}_i = 2.5 + 0.05(i-1)$. $\ddot{\lambda}_{i,pre} = 3.0 + 0.05(i-1)$, $\overline{\tilde{r}_i} = 1.0 + 0.05(i-1)$. $\overline{\tilde{r}_i^2} = 1.35\overline{\tilde{r}_i}^2$, $\overline{\ddot{r}_i} = 1.2 + 0.05(i-1)$, $\overline{\ddot{r}_i^2} = 1.5\overline{\ddot{r}_i}^2$, $m_i = 4$, $s_i = 2.5 + 0.1(i-1)$, $c_i = 10.0 + 0.5(i-1)$, $B_i = 3.0 + 0.1(i-1)$, $N_i = -174 - 0.1(i-1)$, $\eta_i = (0.5 + 0.05(i-1)) * 10^{-4}$, $\pi_i = (2.0 + 0.1(i-1)) * 10^{-10}$, for all $3 \le i \le n$. Parameter settings of $\text{EN}_1 \sim \text{EN}_3$ are the same, except that queue disciplines are different. The DC is given by $\tilde{\lambda}_c = 6.0$, $\ddot{\lambda}_{c,pre} = 4.0$, $\overline{\tilde{r}_c} = 1.35$, $\overline{\tilde{r}_c^2} = 1.55\overline{\tilde{r}_c}^2$, $s_c = 3.5$, $c_b = 11.0$, $N_b = -174.0$, $t_p = 0.4$, and $\eta_c = 0.6 * 10^{-4}$. We set $C^* = 0.003$.

Table I presents other experimental data: (1) $\overline{\ddot{r}_0}/s_i$ (average processing latency of MD's generic tasks on each node); (2) $\overline{\ddot{d}}/c_i$ (average communication latency for offloading one generic task from the MD to ENs/BS); (3) g_i (channel gains); (4) P_i^{trs} (average transmission power); (5) C_i^{off} (the MD's AMC for process one generic task remotely on ENs/DC); (6) DS_d (queue discipline applied by each node).

From Table II(a), we obtain the optimal offloading strategy $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_5, \ddot{\lambda}_c)$, where the minimized ART is $\overline{T} = 2.387431$ and the AMC is $\overline{C} = 0.002999$.

Example 2: In this example, the parameter settings are the same as those in Example 1, except that the cost constraint is set to $C^* = 0.004$. From Table II(b), we obtain the optimal offloading strategy $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_5, \ddot{\lambda}_c)$, the minimized ART $\overline{T} = 1.082083$, and the AMC of the MD $\overline{C} = 0.004000$.

Example 3: The MD is given by $\tilde{\lambda}_0 = 1.0$, $\ddot{\lambda} = 15.0$, $\overline{\ddot{r}_0} = 1.0$, $\overline{\ddot{r}_0^2} = 1.35$, $s_0 = 1.2$, $P_0^* = 2.0$, $\theta_0 = 2.44 * 10^{-4}$. There are $n = 5$ ENs, where EN_i is specified with $\tilde{\lambda}_i = 2.6 + 0.05(i-1)$, $\ddot{\lambda}_{i,pre} = 3.2 + 0.05(i-1)$, $\overline{\tilde{r}_i} = 1.0 + 0.05(i-1)$, $\overline{\tilde{r}_i^2} = 1.35\overline{\tilde{r}_i}^{2^2}$, $\overline{\ddot{r}_i} = 1.2 + 0.05(i-1)$, $\overline{\ddot{r}_i^2} = 1.5\overline{\ddot{r}_i}^{2^2}$, $m_i = 3 + i$, $s_i = 2.5 + 0.1(i-1)$, $c_i = 10.0 + 0.5(i-1)$, $B_i = 2.8 + 0.1(i-1)$, $N_i = -174 - 0.1(i-1)$, $\eta_i = (0.5 + 0.05(i-1)) * 10^{-4}$, $\pi_i = (2.0 + 0.1(i-1)) * 10^{-10}$, for all $1 \le i \le n$. The DC is given by $\tilde{\lambda}_c = 6.5$, $\ddot{\lambda}_{c,pre} = 5.0$, $\overline{\tilde{r}_c} = 1.35$, $\overline{\tilde{r}_c^2} = 1.55\overline{\tilde{r}_c}^2$, $s_c = 3.4$, $c_b = 10.0$, $N_b = -173.0$, $t_p = 0.4$, and $\eta_c = 0.6 * 10^{-4}$. We set $T^* = 1.0$. Table III presents other experimental data in the environment. From Table IV(a), we obtain the optimal offloading strategy $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_5, \ddot{\lambda}_c)$, the minimized AMC $\overline{C} = 0.003335$, and the ART $\overline{T} = 0.999990$.

Example 4: In this example, the parameter settings are the same as those in Example 3, except that the total arrival rate of generic tasks generated on the MD is set to $\ddot{\lambda} = 21$. From Table IV(b), we obtain the optimal offloading strategy $(\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_5, \ddot{\lambda}_c)$, the minimized AMC of the MD $\overline{C} = 0.003582$, and the ART of generic tasks $\overline{T} = 1.000001$.

Example 5: The MD is given by $\tilde{\lambda}_0 = 1.1, \ddot{\lambda} = 25.0, \overline{\ddot{r}_0} = 1.0$, $\overline{\ddot{r}_0^2} = 1.35$, $s_0 = 1.2$, $P_0^* = 1.5$, $\theta_0 = 2.34 * 10^{-4}$. There are $n = 5$ ENs, where EN_i is specified with $\tilde{\lambda}_i = 2.6 + 0.05(i-1)$, $\ddot{\lambda}_{i,pre} = 3.2 + 0.05(i-1)$, $\overline{\tilde{r}_i} = 0.7 + 0.05(i-1)$, $\overline{\tilde{r}_i^2} =$

TABLE I
EXPERIMENTAL PARAMETERS OF EXAMPLES 1 AND 2

i	0	1	2	3	4	5	c(DC)
$\overline{\ddot{r}}_0/s_i$	1.153846	0.600000	0.600000	0.600000	0.576923	0.555555	0.428571
$\ddot{d}/c_i$	-	0.100000	0.100000	0.100000	0.095238	0.090909	$(\ddot{d}/c_b = 0.090909)$
g_i	-	-45.374166	-45.374166	-45.374166	-37.432194	-31.189472	$(g_b = -31.174308)$
P_i^{trs}	-	104.452173	104.452173	104.452173	136.427352	175.760900	$(P_b^{trs} = 257.955237)$
C_i^{off}	-	0.002598	0.002598	0.002598	0.003225	0.003958	0.005781
DS_d	DS_1	DS_1	DS_2	DS_3	DS_2	DS_2	DS_2

TABLE II
EXPERIMENTAL RESULTS FOR MINIMIZING ART WITH A COST CONSTRAINT

(a) Example 1

i	0	1	2	3	4	5	c(DC)
ρ_i	0.836364	0.941393	0.932590	0.764989	0.933501	0.934368	0.348883
$\ddot{\lambda}_i$	0.391516	1.893676	1.843373	0.885651	1.841601	1.834006	1.310170
$\overline{T_i}$	4.000686	2.055902	2.235538	5.593603	2.219111	2.207423	0.919497
$\beta = 313.702207, \omega = -1.905754, C_0^{loc} = 0.001198, \overline{T} = 2.387431$							

(b) Example 2

i	0	1	2	3	4	5	c(DC)
ρ_i	0.620552	0.860608	0.842145	0.610000	0.845369	0.848420	0.539364
$\ddot{\lambda}_i$	0.204478	1.432048	1.326542	0.000000	1.317127	1.302206	4.417595
$\overline{T_i}$	1.944028	1.154367	1.205710	2.474269	1.190514	1.178201	0.920968
$\beta = 43.497522, \omega = -0.344300, C_0^{loc} = 0.000998, \overline{T} = 1.082083$							

TABLE III
EXPERIMENTAL PARAMETERS OF EXAMPLES 3 AND 4

i	0	1	2	3	4	5	c(DC)
$\overline{\ddot{r}}_0/s_i$	0.833333	0.400000	0.384615	0.370370	0.357142	0.344827	0.294117
$\ddot{d}/c_i$	-	0.100000	0.095238	0.090909	0.086956	0.083333	$(\ddot{d}/c_b = 0.100000)$
g_i	-	-45.374166	-44.037500	-41.279687	-37.432194	-31.189472	$(g_b = -35.109487)$
P_i^{trs}	-	116.908182	129.564606	148.111740	174.424223	222.848258	$(P_b^{trs} = 171.436584)$
C_i^{off}	-	0.002902	0.003065	0.003345	0.003765	0.004601	0.004243
DS_d	DS_2	DS_1	DS_2	DS_3	DS_2	DS_2	DS_3

TABLE IV
EXPERIMENTAL RESULTS FOR MINIMIZING AMC WITH A PERFORMANCE CONSTRAINT

(a) Example 3

i	0	1	2	3	4	5	c(DC)
ρ_i	0.778025	0.911236	0.891419	0.570105	0.838737	0.350000	0.319117
$\ddot{\lambda}_i$	0.433630	2.137890	3.802004	1.586332	7.040142	0.000000	0.000000
$\overline{T_i}$	3.475744	1.230460	1.027071	1.611601	0.625073	0.428806	0.443754
$\beta = 191.024197, \omega = -0.815539, C_0^{loc} = 0.000816, \overline{C} = 0.003335$							

(b) Example 4

i	0	1	2	3	4	5	c(DC)
ρ_i	0.774811	3.424264	0.901370	0.599231	0.881537	0.436925	0.486936
$\ddot{\lambda}_i$	0.429773	2.190519	3.905689	1.965182	7.714751	1.624163	3.169918
$\overline{T_i}$	3.424264	1.301911	1.099876	1.885293	0.733480	0.430593	0.731192
$\beta = 114.433779, \omega = -0.547178, C_0^{loc} = 0.000815, \overline{C} = 0.003582$							

TABLE V
EXPERIMENT OF MINIMIZING COST-PERFORMANCE RATIO

(a) Experimental Parameters of Example 5.

i	0	1	2	3	4	5	c(DC)
$\overline{\ddot{r}}_0/s_i$	0.833333	0.384615	0.384615	0.370370	0.357142	0.344827	0.294117
$\ddot{d}/c_i$	-	0.105263	0.105263	0.100000	0.095238	0.090909	$(\ddot{d}/c_b = 0.105263)$
g_i	-	-39.106422	-39.106422	-49.454888	-35.707396	-37.928949	$(g_b = -32.795174)$
P_i^{trs}	-	118.470956	118.470956	101.284140	151.034201	152.512289	$(P_b^{trs} = 159.837359)$
C_i^{off}	-	0.002968	0.002973	0.002430	0.003430	0.003314	0.003992
DS_d	DS_1	DS_1	DS_1	DS_2	DS_2	DS_2	DS_3

(b) Experimental Results of Example 5.

i	0	1	2	3	4	5	c(DC)
ρ_i	0.546802	0.785691	0.785549	0.792743	0.779933	0.797772	0.444307
$\ddot{\lambda}_i$	0.106163	2.430895	2.429733	4.127593	5.722008	7.862064	2.321541
$\overline{T_i}$	1.280251	0.651922	0.651736	0.625469	0.557672	0.532528	0.610843
$\beta = 7.404866, \omega = -0.063635, C_0^{loc} = 0.000568, \overline{C} = 0.003179, \overline{T} = 0.587271$							

$1.1\overline{\ddot{r}_i}^2$, $\overline{\ddot{r}_i} = 0.9 + 0.05(i-1)$, $\overline{\ddot{r}_i^2} = 1.3\overline{\ddot{r}_i}^2$, $m_i = 2 + i$, $s_i = 2.5 + 0.1(i-1)$, $c_i = 9.0 + 0.5(i-1)$, $B_i = 2.7 + 0.1(i-1)$, $N_i = -174 - 0.1(i-1)$, $\eta_i = (0.5 + 0.05(i-1)) * 10^{-4}$, $\pi_i = (2.0 + 0.1(i-1)) * 10^{-10}$, for all $2 \leq i \leq n$. EN_1 has the same parameter settings as EN_2, except that $\eta_1 = 0.5 * 10^{-4}$ and $\pi_1 = 2.0 * 10^{-10}$ (i.e., $\eta_1 < \eta_2, \pi_1 < \pi_2$). The DC is given by $\tilde{\lambda}_c = 6.0$, $\ddot{\lambda}_{c,pre} = 5.5$, $\overline{\tilde{r}_c} = 1.35$, $\overline{\tilde{r}_c^2} = 1.55\overline{\tilde{r}_c}^2$, $s_c = 3.4$, $c_b = 9.5$, $N_b = -174.0$, $t_p = 0.4$, and $\eta_c = 0.55 * 10^{-4}$.

Table V shows other experimental data and results of minimizing the cost-performance ratio. From Table V(b), the minimized

TABLE VI
OFFLOADING DECISIONS IN PERFORMANCE COMPARISON

i	Our Solution		LWSF		PSO		DDPG	
	$\ddot{\lambda}_i$	ρ_i	$\ddot{\lambda}_i$	ρ_i	$\ddot{\lambda}_i$	ρ_i	$\ddot{\lambda}_i$	ρ_i
Minimizing ART under AMC constraint								
0	0.391516	0.836364	0.355488	0.794794	0.373419	0.815483	0.474900	0.932577
1	1.893676	0.941393	1.485647	0.869988	1.949364	0.951138	1.807705	0.926348
2	1.843373	0.932590	1.485647	0.869988	1.797412	0.924547	1.840808	0.932141
3	0.885651	0.764989	1.485647	0.869988	0.917162	0.770503	0.938165	0.774178
4	1.841601	0.933501	1.491486	0.874668	1.880249	0.939996	1.812015	0.928530
5	1.834006	0.934368	1.493336	0.879310	1.788464	0.927007	1.792875	0.927720
c(DC)	1.310170	0.348883	2.202745	0.403596	1.293927	0.347887	1.333528	0.350315
Minimizing AMC under ART constraint								
0	0.433630	0.778025	0.000000	0.416666	0.389146	0.740955	0.000000	0.416666
1	2.137890	0.911236	1.898600	0.881325	2.212928	0.920616	0.000000	0.644000
2	3.802004	0.891419	3.288864	0.842173	3.802657	0.891482	0.000000	0.526538
3	1.586332	0.570105	0.000000	0.448148	1.770362	0.584253	0.000000	0.448148
4	7.040142	0.838737	6.387928	0.797359	6.824904	0.825082	0.001032	0.392157
5	0.000000	0.350000	0.000000	0.350000	0.000000	0.350000	10.898455	0.933286
c(DC)	0.000000	0.319117	3.424606	0.500420	0.000000	0.319117	4.100511	0.536203
Minimizing Cost-Performance Ratio								
0	0.106163	0.546802	0.000000	0.458333	0.223898	0.644915	0.000019	0.458349
1	2.430895	0.785691	2.787123	0.829318	3.387602	0.902859	3.038749	0.860135
2	2.429733	0.785549	2.787123	0.829318	2.084834	0.743309	0.257823	0.519556
3	4.127593	0.792743	4.220405	0.801475	4.560407	0.833460	4.751182	0.851407
4	5.722008	0.779933	7.798219	0.936472	5.486460	0.762173	5.785776	0.784741
5	7.862064	0.797772	7.407127	0.769453	7.363114	0.766714	8.686562	0.849096
c(DC)	2.321541	0.444307	0.000000	0.320588	1.893681	0.421506	2.479884	0.452746

cost-performance ratio is $R = 0.001867$, where $\overline{T} = 0.587271$ and $\overline{C} = 0.003179$.

In the supplementary file, available online, we summarize the objective laws from these examples in Section 8, discuss how queue disciplines impact offloading decisions in Section 9, and design simulation experiments to analyze the task response time on different nodes in Section 11, available online.

IX. PERFORMANCE COMPARISON

To further demonstrate the effectiveness of our solution, we compare it with other algorithms, including a greedy-based method, particle swarm optimization (PSO) [33], and the deep deterministic policy gradient (DDPG) algorithm [34].

Lowest-Weighted-Sum-First (LWSF): Here, the target MD will preferentially offload tasks to nodes that have the lower weighted sum of cost and performance, i.e., $g = w_c \cdot \overline{C} + (1 - w_c) \cdot \overline{T}$, where w_c is the weighting factor and its value is set according to the specific optimization problem. For instance, taking too small a value of w_c may lead to an unsatisfied cost constraint in terms of minimizing average response time under a given cost constraint.

Particle Swarm Optimization: PSO is a heuristic algorithm that solves optimization problems by searching for candidate solutions in an iterative way. Let's consider a swarm with $N = 20$ particles moving in an $n + 2$ dimensional search space determined by the bounds of $\ddot{\lambda}_0, \ddot{\lambda}_1, \ldots, \ddot{\lambda}_n, \ddot{\lambda}_c$ (discussed in Section V-A). We set the number of iterations as $k = 100$, the inertia weight as $\omega = 0.8$, the cognitive coefficient as $c_p = 1.5$, and social coefficient as $c_g = 1.5$.

Deep Deterministic Policy Gradient: DDPG is a classical DRL algorithm for learning deterministic policies from continuous action spaces [35]. Employing DDPG, the offloading process can be modeled as Markov decision processes, which mainly consists of three components: 1) *state space* that includes the information of the current CECC environment (e.g., computing resources, execution requirements, and latency/cost constraints); 2) *action space* that is defined as the MD's offloading strategy; 3) *reward function* that refers to the optimization objective and constraint. (Due to space limitations, detailed information regarding the DDPG algorithm used in the comparison experiments, such as the reward function definition, hyperparameter settings, and convergence performance, are provided in Section 10 of the supplementary file, available online.)

TABLE VII
THE ART AND AMC IN PERFORMANCE COMPARISON

	Our Solution	LWSF	PSO	DDPG
Minimizing ART under AMC constraint				
$\overline{T}$	2.387431	2.924557	2.428785	2.581306
$\overline{C}$	0.002999	0.003280	0.002996	0.002999
Minimizing AMC under ART constraint				
$\overline{T}$	0.999990	0.722808	1.013424	0.940901
$\overline{C}$	0.003335	0.003612	0.003334	0.004503
Minimizing Cost-Performance Ratio				
$\overline{T}$	0.587271	0.763248	0.639094	0.642902
$\overline{C}$	0.003179	0.003124	0.0031304	0.003194
R	0.001867	0.002384	0.002000	0.002054

For simplicity, we use the same parameter settings of numerical examples in the previous sections for comparative analysis. That is, we use the parameters in Example 1 for minimizing $\overline{T}$ under constraint C^*, Example 3 for minimizing $\overline{C}$ under constraint T^*, and Example 5 for minimizing R. Tables VI and VII show the corresponding experimental results, including the offloading decisions, the ART of offloadable tasks, and the MD's AMC. To sum up, our methods are effective and can obtain the optimal offloading decision of the target MD in various situations.

X. Conclusion

In this article, we have conducted research on a priority-based offloading optimization approach in CECC. We have specifically focused on three queue disciplines that serve different types of tasks and have thoroughly analyzed their performance and cost metrics. By leveraging the KKT conditions, we have devised a series of algorithms to determine the optimal offloading decisions for MDs, aiming to strike a balance between enhancing performance and reducing costs. Moreover, we have implemented numerical examples to effectively demonstrate the efficacy of our proposed methods. This work represents an initial and valuable contribution to the field of priority-based offload optimization in complex distributed computing environments. It is important to note that the accuracy of our solution relies heavily on the precision of environmental parameters. Hence, it is crucial to adjust and refine these parameters accordingly when changes occur in the offloading environment.

However, there are several areas that require further investigation in future work. First, we have not analyzed resource configuration or multi-user scenarios, where there may be competitive or cooperative relationships among users. This aspect deserves careful examination. Second, our consideration of the MD's power consumption assumes an energy-limited scenario without considering EH or rechargeable capabilities. It would be meaningful to explore scenarios involving green energy and rechargeable devices. Third, our focus has primarily been on the impact of task priority on the offloading strategy in heterogeneous environments. However, real-world scenarios involve multiple factors, such as communication conditions, implementation environment, software architecture, and network interference, which can influence offloading, transmission, and execution in positive or negative ways. For instance, unstable network conditions can result in long-tail latency, making service guarantee challenging. Moreover, it would be interesting to apply the proposed scheme to engineering applications, such as production scheduling and design, as suggested in references [36], [37]. Further research is necessary to address these issues and explore more complex application scenarios.

Acknowledgment

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments are very important to improve this manuscript.

References

[1] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[2] Z. Tong, X. Deng, J. Mei, B. Liu, and K. Li, "Response time and energy consumption co-offloading with slrta algorithm in cloud-edge collaborative computing," *Future Gener. Comput. Syst.*, vol. 129, pp. 64–76, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2100443X

[3] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surv. Tuts.*, vol. 23, no. 4, pp. 2131–2165, Fourth Quarter 2021.

[4] K. Li, "Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2019.2904680.

[5] R. Rodriguez-Zurrunero, R. Utrilla, A. Rozas, and A. Araujo, "Process management in IoT operating systems: Cross-influence between processing and communication tasks in end-devices," *Sensors*, vol. 19, no. 4, 2019, Art. no. 805. [Online]. Available: https://www.mdpi.com/1424-8220/19/4/805

[6] B. Liu, Y. Wan, F. Zhou, Q. Wu, and R. Q. Hu, "Resource allocation and trajectory design for MISO UAV-assisted MEC networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4933–4948, May 2022.

[7] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.

[8] J. Feng, L. T. Yang, X. Nie, and N. J. Gati, "Edge-cloud-aided differentially private tucker decomposition for cyber-physical-social systems," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8387–8396, Jun. 2022.

[9] H. Gao, X. Wang, X. Ma, W. Wei, and S. Mumtaz, "Com-DDPG: A multiagent reinforcement learning-based offloading strategy for mobile edge computing," 2020, *arXiv:2012.05105*.

[10] Z. Sharif, L. Tang Jung, M. Ayaz, M. Yahya, and S. Pitafi, "Priority-based task scheduling and resource allocation in edge computing for health monitoring system," *J. King Saud Univ. - Comput. Inf.*, vol. 35, no. 2, pp. 544–559, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157823000010

[11] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[12] X. Wang, Z. Ning, L. Guo, S. Guo, X. Gao, and G. Wang, "Online learning for distributed computation offloading in wireless powered mobile edge computing networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1841–1855, Aug. 2022.

[13] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 2, pp. 624–634, Jun. 2021.

[14] H. Hu, Q. Wang, R. Q. Hu, and H. Zhu, "Mobility-aware offloading and resource allocation in a MEC-enabled IoT network with energy harvesting," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17541–17556, Dec. 2021.

[15] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, Third Quarter 2021.

[16] S. Yao et al., "Blockchain-empowered collaborative task offloading for cloud-edge-device computing," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3485–3500, Dec. 2022.

[17] M. Liwang and X. Wang, "Overbooking-empowered computing resource provisioning in cloud-aided mobile edge networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2289–2303, Oct. 2022.

[18] M. Liwang and X. Wang, "Overbook in advance, trade in future: Computing resource provisioning in hybrid device-edge-cloud networks," *IEEE Netw.*, to be published, doi: 10.1109/MNET.125.2200187.

[19] A. O. Allen, *Probability, Statistics, and Queueing Theory*. Houston, TX, USA: Gulf Professional Publishing, 1990.

[20] W. Lin, F. Shi, W. Wu, K. Li, G. Wu, and A.-A. Mohammed, "A taxonomy and survey of power models and power modeling for cloud servers," *ACM Comput. Surv.*, vol. 53, no. 5, Sep. 2020, Art. no. 100. [Online]. Available: https://doi.org/10.1145/3406208

[21] C. Jin, X. Bai, C. Yang, W. Mao, and X. Xu, "A review of power consumption models of servers in data centers," *Appl. Energy*, vol. 265, 2020, Art. no. 114806. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261920303184

[22] K. Li, "Profit maximization in a federated cloud by optimal workload management and server speed setting," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 3, pp. 668–680, Third Quarter 2022.

[23] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 7947–7962, Dec. 2021.

[24] Z. He, K. Li, and K. Li, "Cost-efficient server configuration and placement for mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 9, pp. 2198–2212, 2022.

[25] J. Huang, R. Li, Y. Wei, J. An, and W. Chang, "Bi-directional timing-power optimisation on heterogeneous multi-core architectures," *IEEE Trans. Sustain. Comput.*, vol. 6, no. 4, pp. 572–585, Fourth Quarter 2021.
[26] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[27] P. Hokstad, "Approximations for the m/g/m queue," *Oper. Res.*, vol. 26, no. 3, pp. 510–523, 1978.
[28] T. Williams, "Special products and uncertainty in production/inventory systems," *Eur. J. Oper. Res.*, vol. 15, no. 1, pp. 46–54, 1984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037722178490047X
[29] J. P. Buzen and A. B. Bondi, "The response times of priority classes under preemptive resume in m/m/m queues," *Oper. Res.*, vol. 31, no. 3, pp. 456–465, 1983. [Online]. Available: https://doi.org/10.1287/opre.31.3.456
[30] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079–1092, Feb. 2019.
[31] J. Yan, S. Bi, L. Duan, and Y.-J. A. Zhang, "Pricing-driven service caching and task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4495–4512, Jul. 2021.
[32] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 420–423, Jun. 2018.
[33] F. Marini and B. Walczak, "Particle swarm optimization (PSO). A tutorial," *Chemom. Intell. Lab. Syst.*, vol. 149, pp. 153–165, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743915002117
[34] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA: PMLR, 2016, pp. 2829–2838.
[35] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.
[36] Y. Wu, F. He, D. Zhang, and X. Li, "Service-oriented feature-based data exchange for cloud-based design and manufacturing," *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 341–353, Mar./Apr. 2018.
[37] V. Iannino et al., "A hybrid approach for improving the flexibility of production scheduling in flat steel industry," *Integr. Comput.-Aided Eng.*, vol. 29, no. 4, pp. 367–387, 2022.

Zhenli He received the MS degree in software engineering and the PhD degree in systems analysis and integration from Yunnan University (YNU), Kunming, China, in 2012 and 2015, respectively. He was a postdoctoral researcher with Hunan University (HNU), Changsha, China, from 2019 to 2021. He is currently an associate professor with the School of Software, Yunnan University, Kunming, China. His current research interests include edge computing, energy-efficient computing, heterogeneous computing, and machine learning.

Yanan Xu received the BS and MS degrees in software engineering from Yunnan University, China, in 2019 and 2023, respectively. She is currently working toward the PhD degree in information and communication engineering with Shenzhen University, Shenzhen, China. Her current research interests include edge computing and energy-efficient computing.

Mingxiong Zhao received the BS degree in electrical engineering and the PhD degree in information and communication engineering from the South China University of Technology (SCUT), Guangzhou, China, in 2011 and 2016, respectively. Since 2016, he has been with the School of Software, Yunnan University, Kunming, China, where he is currently an associate professor. His current research interests include physical layer security, mobile edge computing, and edge AI techniques.

Wei Zhou received the PhD degree from the University of Chinese Academy of Sciences. He is currently a full professor with the Software School, Yunnan University. His current research interests include the distributed data intensive computing and bio-informatics. He is currently a fellow of the China Communications Society, a member of the Yunnan Communications Institute, and a member of the Bioinformatics Group of the Chinese Computer Society. He won the Wu Daguan Outstanding Teacher Award of Yunnan University, in 2016, and was selected into the Youth Talent Program of Yunnan University, in 2017. Hosted a number of National Natural Science Foundation projects.

Keqin Li (Fellow, IEEE) received the BS degree in computer science from Tsinghua University, in 1985 and the PhD degree in computer science from the University of Houston, in 1990. He is currently a SUNY Distinguished professor with the State University of New York and also a National Distinguished professor with Hunan University, China. He has authored or coauthored more than 910 journal articles, book chapters, and refereed conference papers. He received several best paper awards from international conferences including *PDPTA-1996*, *NAECON-1997*, *IPDPS-2000*, *ISPA-2016*, *NPC-2019*, *ISPA-2019*, *CPSCom-2022*. He holds nearly 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top five most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He was a 2017 recipient of the Albert Nelson Marquis Lifetime Achievement Award for being listed in Marquis Who's Who in Science and Engineering, Who's Who in America, Who's Who in the World, and Who's Who in American Education for more than twenty consecutive years. He received the Distinguished Alumnus Award of the Computer Science Department with the University of Houston, in 2018. He received the IEEE TCCLD Research Impact Award from the IEEE Technical Committee on Cloud Computing, in 2022. He is an AAAS fellow, and an AAIA fellow. He is a member of SUNY Distinguished Academy. He is a member of Academia Europaea (Academician of the Academy of Europe).