Throughput-Aware Cooperative Task Offloading in Dynamic Mobile Edge Computing Systems

Longbao Dai, Fanzi Zeng, *Member, IEEE*, Haoran Kong, Jianghao Cai, Hongbo Jiang, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—With the commercialization of fifth-generation (5G) mobile communication technology and the rapid proliferation of mobile devices (MDs), demand for data computation is surging. This growth increases the reliance of MDs on low latency and high throughput. For this purpose, Mobile Edge Computing (MEC) enhances the user's data processing capability by offloading computation tasks to servers at the network edge. However, achieving high efficiency in task offloading is challenging due to factors such as decision complexity, network dynamics, and user data privacy protection. Additionally, energy causal constraints and the coupling between offloading proportions and resource distribution cannot be ignored. In this paper, we first establish a dynamic task offloading problem to optimize the long-term throughput of the system. Using perturbed Lyapunov optimization, we transform MD delay and energy threshold constraints into the stability control of corresponding virtual queues. Then, we propose the Lyapunov-guided federated deep reinforcement learning (DRL) online task offloading algorithm called LyFOTO, which combines a federated learning (FL) framework and an Actor-Critic (AC) model. Under favorable communication conditions, the LyFOTO algorithm adaptively boosts system throughput; under poorer conditions, it properly delays task offloading, without violating queue backlog constraints. Through mathematical analysis, we discuss the performance of the LyFOTO algorithm. Simulation experiments validate that LyFOTO effectively balances system throughput and device battery energy. Finally, Comparative results show that LyFOTO outperforms other benchmark algorithms in maximizing system throughput while ensuring task backlog and energy threshold constraints.**

***Index Terms*—Deep reinforcement learning, federated learning (FL), Lyapunov optimization, mobile edge computing (MEC), task offloading.**

I. Introduction

With the commercial deployment of fifth-generation (5G) mobile communication technology and the exponential growth of mobile devices (MDs), an era of ubiquitous connectivity is emerging [1], [2]. It is estimated that by 2028, over 32.2 billion MDs will be connected to the internet worldwide, and generate vast amounts of interactive data [3]. This exponential growth has revealed several issues, including high latency, network congestion, and data privacy risks [4], [5]. Mobile Edge Computing (MEC) is regarded as a promising solution that enhances user data processing capabilities by deploying computational resources near base stations (BS) at the network edge [6]. This approach significantly reduces the transmission delay of data, enabling real-time applications to run more efficiently on MDs. In this context, the design of appropriate task offloading strategies helps to optimize resource utilization and boost network performance [7], [8]. However, in multi-BS collaborative offloading scenarios, MEC faces a series of bottlenecks when maintaining dynamic system performance [9]. Specifically, how to optimize the long-term performance of one or more metrics under system constraints is a key problem.

Many researchers have carried out extensive studies on the above problems. The authors in [10], [11], [12], [13] have adopted convex optimization methods to optimize task offloading decisions. Since the offloading strategy is typically generated iteratively, these methods often involve high computational complexity. This makes their deployment in large-scale networks practically infeasible. To improve real-time offloading efficiency, some researchers [14], [15], [16], [17] have introduced Lyapunov optimization methods. In real-world scenarios, optimization models are often formulated as continuous stochastic problems, which restricts the applicability of Lyapunov optimization techniques designed primarily for discrete convex optimization. In recent years, Deep reinforcement learning (DRL) has gained widespread application for its adaptability to dynamic and uncertain conditions [18], [19], [20], [21]. It optimizes task offloading decisions by continuously learning from interactions with the environment. Despite their inspiring results, existing works struggle to maintain high system throughput in dynamic environments due to frequent network fluctuations and device heterogeneity. Without ensuring high throughput, the system may fail to meet the growing computational demands of users, leading to degraded overall performance.

There is a critical imperative to optimize system throughput in multi-BS collaborative offloading scenarios. The problems caused by network fluctuation and device heterogeneity in dynamic systems still need to be solved [22]. Specifically, we face the following challenges. First, how can resource distribution among BS be balanced with limited global information? The

Received 5 December 2024; revised 21 March 2025; accepted 15 July 2025. Date of publication 24 July 2025; date of current version 5 November 2025. This work was supported in part by the Joint Funds of the National Natural Science Foundation of China under Grant U24A20246 and Grant U24A20247, and in part by the Program of National Natural Science Foundation of China under Grant 62232007. Recommended for acceptance by C. Lin. *(Corresponding author: Fanzi Zeng.)*

Longbao Dai, Fanzi Zeng, Haoran Kong, Jianghao Cai, and Hongbo Jiang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China (e-mail: longbdai00@hnu.edu.cn; zengfanzi@hnu.edu.cn; konghaoran@hnu.edu.cn; caijianghao@hnu.edu.cn; hongbojiang@hnu.edu.cn).

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

Digital Object Identifier 10.1109/TMC.2025.3592450

MDs served and the task demands across different BS vary significantly, leading to resource competition and scheduling conflicts. During multi-BS collaborative load balancing, the lack of global information sharing makes it difficult to maintain balanced resource allocation, thereby increasing the complexity of throughput optimization. Second, how can queue delays be stabilized and minimum battery levels be maintained to maximize throughput? Frequent task offloading and computation increase the MD's energy consumption and make task queue delays uncontrollable. Ensuring that the battery level of MDs remains above the minimum threshold while stabilizing queue delays to maximize throughput remains a critical challenge. Third, how can resources be allocated among MDs under dynamic conditions? Task offloading requires careful consideration of dynamic changes in MD locations, server workloads, and network conditions, which increases the complexity of resource allocation schemes. Inappropriate decisions not only reduce throughput but also violate system constraints [23].

To address the aforementioned challenges, this paper proposes a novel task offloading approach in multi-BS collaborative offloading scenarios, aiming to jointly optimize task queue delays and energy level constraints while enhancing system throughput. For the first challenge, we construct a distributed training scenario that leverages the federated learning (FL) framework to achieve collaborative optimization among multi-BS while preserving data privacy. For the second challenge, we designed two virtual queues to transform the task queue delay and battery energy constraints into the stability constraints of the virtual queues. For the third challenge, we employ Lyapunov optimization to derive an upper bound for the *drift-plus-penalty* function, decomposing the long-term optimization problem into per time slot sub-problems. To handle the complex decision-making process in dynamic environments, we utilize DRL methods to continuously adjust the offloading strategy. Based on this framework, we propose a fast optimal resource distribution algorithm to optimize system throughput in real-time while satisfying the associated constraints. The following are the main contributions of this article.

- *Optimization target based on perturbation Lyapunov technique:* We construct a multi-BS collaborative MEC scenario model with heterogeneous computing capabilities. Using perturbed Lyapunov techniques, the long-term system throughput optimization problem is converted into a non-convex problem relying solely on the current system state.
- *Task offloading decision employing FL and the AC model:* We utilize FL to share training parameters among base stations and aggregate and distribute these parameters through a global server. Based on this framework, the Actor-Critic (AC) model in DRL is introduced to determine task offloading decisions for each time slot.
- *Simulation verification from benchmark algorithms:* We conducted a comparative analysis with three benchmark algorithms to demonstrate the superiority of the proposed algorithm. It is shown that the algorithm not only enhances system throughput but also effectively stabilizes device task queues and battery energy levels near the threshold.

This method is applicable in the field of smart cities, offering a solution specifically tailored to meet the demands of efficient computing and real-time decision-making. In a smart city environment, Internet of Things (IoT) devices continuously generate a large volume of latency-sensitive computational tasks, such as traffic flow monitoring, public safety management, and environmental data collection. The real-time processing capability of these tasks directly impacts the efficiency and safety of urban operations. The LyFOTO algorithm enhances data processing timeliness through dynamic task offloading and energy efficiency optimization, while effectively reducing device energy consumption and network load. This ensures system efficiency and sustainable operation, providing robust technological support for the stable development of smart cities.

The rest of this paper is organized as follows. Section II reviews the related work in MEC. Section III describes the system model utilized in this study, and provides a rigorous definition of the offloading optimization problem. In Section IV, we provide a detailed explanation of how to leverage the perturbation Lyapunov optimization method to transform the optimization problem into a non-convex problem reliant solely on current system information and present an upper bound for the drift-plus-penalty function. Section V integrates FL frame with the Actor-Critic model to address the non-convex problem and presents an algorithm called LyFOTO. Section VI we discuss the asymptotic optimality of the proposed algorithm by mathematical analysis. Section VII conducts a series of simulation experiments to validate the performance of the proposed algorithm. In Section VIII, we summarize this paper.

II. Related Work

In the MEC system, task offloading scheme include partial task offloading scheme and binary task offloading scheme [24]. The former allows task data to be executed in parallel on both MDs and MEC servers, while the latter requires the entire task data to be executed as a whole either on MD or MEC server. This paper adopts the binary task offloading scheme, which is widely used in MEC scenarios.

Before the widespread application of machine learning (ML), convex optimization methods dominated resource allocation and management problems in MEC systems. Ei et al. [10] proposed a multiple uncrewed aerial vehicles (UAV) assisted MEC scenario, where they modeled the task offloading problem as a non-convex problem and used the Block Successive Upper Bound Minimization (BSUM) method to find an asymptotically optimal solution to the objective function. Li et al. [12] proposed a statistical computation and transmission model to quantify the correlation between task offloading and Quality of Service (QoS), and used convex optimization methods to solve for the optimal task offloading strategy. Dai et al. [13] modeled the UAV-assisted vehicle task offloading problem as a Markov chain and employed convex optimization methods to iteratively solve for the optimal task offloading strategy. However, relying solely on convex optimization methods for iterative optimization typically results in high computational complexity. Consequently, while the aforementioned algorithms can quickly find optimal offloading solutions in small-scale scenarios, they struggle to adapt to environments with rapidly increasing numbers of devices.

To reduce the computational complexity of offloading algorithms, some researchers have adopted stochastic optimization strategies to improve task offloading efficiency, among which Lyapunov theory is a commonly used approach [14], [15], [16], [17], [25]. Li et al. [14] considered a mobile cloud computing (MCC) scenario. They proposed a queue backlog model and leveraged the Lyapunov optimization framework to balance device load backlog with system utility. In an edge cloud scenario, Liet al. [16] established an online learning-assisted cooperative offloading mechanism, using Lyapunov optimization to explore the spatiotemporal optimality of system cost. To meet low-latency requirements, they designed a delay-aware online learning method to predict task completion times. Abbas et al. [25] considered a user-centered real-time task offloading framework, utilizing Lyapunov drift-plus-penalty techniques to design a dynamic task offloading scheme oriented toward support vector machines (SVM). However, the algorithm in this study applies only to SVM-based tasks and cannot be extended to environments with heterogeneous computing requirements. Although the aforementioned studies introduced Lyapunov optimization techniques to effectively reduce algorithm complexity, this approach is limited to discrete convex optimization problems. In real scenarios, optimization problems are often continuous and non-convex, which restricts the applicability of Lyapunov optimization. Additionally, Lyapunov optimization requires frequent updates to system states and constraints, which may introduce extra computational overhead, making it challenging to operate effectively in scenarios with strict real-time requirements.

In recent years, deep reinforcement learning (DRL) methods have gradually emerged as a highly promising solution for MEC task offloading problems [26]. Cheng et al. [18] employed a DRL approach to design a task offloading algorithm, aiming to minimize server energy consumption in a space-air-ground integrated scenario. Additionally, combining deep reinforcement learning (DRL) methods with the Lyapunov optimization framework is considered an effective and innovative strategy, capable of enhancing task processing efficiency while meeting multiple system requirements. Tang et al. [19] modeled the stochastic task offloading problem in MEC as a continuous non-convex problem. They used an Actor-Critic framework combined with Lyapunov optimization techniques to solve for the optimal power cost of both devices and satellites in a multi-ground-device environment. Dai et al. [20] considered a stochastic task environment within digital twin networks and used an asynchronous learning algorithm to determine the optimal resource allocation scheme. Zhang et al. [21] proposed a mobility management framework and joint optimization of service migration decisions between base stations. However, references [18], [19], [20], [21] only focus on the offload decision of a single agent and fail to make full use of global information in large-scale networks. In addition, the above studies adopt a centralized approach where data is processed on servers, posing potential risks of data leakage and privacy exposure during data transmission. Hence, there is

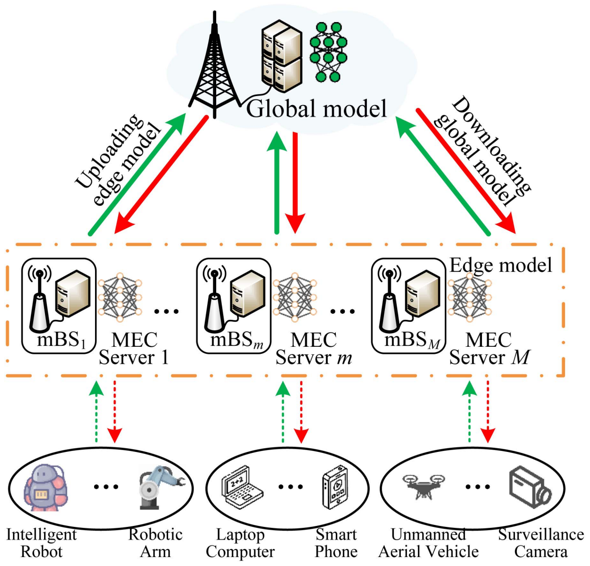

Fig. 1. The system architecture.

an urgent need to establish an efficient multi-agent collaboration mechanism.

This paper proposes an online task-aware offloading algorithm based on a federated DRL framework for heterogeneous MEC scenarios comprising multiple MDs and BS. The algorithm reduces privacy leakage risks by employing distributed training of local agents and a centralized DRL aggregation model. Additionally, a perturbed Lyapunov optimization method is applied to ensure that task backlogs and battery energy levels of devices remain near perturbation thresholds while maximizing system task throughput.

III. System Model and Problem Formulation

This section discusses a multi-network federated offloading environment within an MEC-enabled IoT scenario, as illustrated in Fig. 1. M BS are connected to M MEC servers, serving multiple MDs with limited computing capacity. The sets of MEC servers and MDs are denoted by $\mathcal{M} = \{1, 2, \ldots, m, \ldots, M\}$ and $\mathcal{N} = \{1, 2, \ldots, n, \ldots, N\}$, respectively. Each MD communicates with nearby BS via wireless network. Considering the dynamic nature of the IoT environment, the system operates in discrete time slots $\mathcal{S} = 0, 1, \ldots, s, \ldots, S$ [27], [28]. Assuming each time slot has an equal duration τ, the system can be considered static within each slot [29]. In each time slot, local computing or remote offloading are two approaches for MDs to process tasks. Let a binary variable x_{ij}^s indicate the offloading decision, $x_{ij}^s = 1$ for remote offloading, $x_{ij}^s = 0$ for local computing. Key notations used throughout this paper are summarized in Table I.

A. Task and Queuing Model

At the start of each time frame s, the MD j in edge network (EN) i generates computational tasks a_{ij}^s (in Mbits). The arrival a_{ij}^s is assumed to follow a general independent and identically distributed (i.i.d.) distribution with bounded second order

TABLE I
DEFINITIONS OF MATHEMATICAL NOTATIONS

Notation	Definition
a_{ij}^s	The amount of arrived tasks of MD j in EN i on slot s
$b_{ij}(s)$	The residual battery energy of MD j in EN i on slot s
d_{ij}^s	The amount of tasks processed in the end of slot s
e_{ij}^s	The power consumption of MD j in EN i on slot s
f_{ij}^s	The computation speed of MD j in EN i on slot s
h_{ij}^s	The channel power gain of MD j in EN i on slot s
$t_{ij}(s)$	The initial queue length of MD j in EN i on slot s
r_{ij}^s	The computation rate of MD j in EN i on slot s
x_{ij}^s	The offloading decision of MD j in EN i on slot s
$B_{ij}(s)$	The virtual energy queue of MD j in EN i on slot s
H_{ij}^s	The harvested energy of MD j in EN i on slot s
N_0	The noise power spectral density
P_{ij}^s	The transmit power of MD j in EN i on slot s
$T_{ij}(s)$	The virtual task queue of MD j in EN i on slot s
W	The system bandwidth
ϕ	The number of cycles required to compute 1 bit
ξ	The effective capacitance co-efficient of CPU
τ_{ij}^s	The offloading time of MD j in EN i on slot s
ε_{ij}	The task queue perturbation parameter of MD j in EN i
ϵ_{ij}	The energy level perturbation parameter of MD j in EN i

moment, i.e., $\mathbb{E}[(a_{ij}^s)^2] = \eta_{ij} < \infty$. We treat the value of η_{ij} as known, which can be derived from historical information. As noted in [30], the tasks generated by devices are bit-wise independent, allowing the tasks to be decomposed and processed bit by bit. After the tasks arrive, they are not processed immediately but stored in a task queue buffer $t_{ij}(s)$ that satisfies first-come-first-served (FCFS) basis. To avoid resource under utilization during low load and ensure system stability and efficiency under high load, each device doesn't process all of the incoming tasks at each time slot s. For each MD, d_{ij}^s denotes the amount of tasks processed in the end of current time slot. Hence, the task buffer backlog in next time slot is

$$t_{ij}(s+1) = t_{ij}(s) - d_{ij}^s + a_{ij}^s. \tag{1}$$

Notice that the amount of tasks processed in the end of current time slot don't exceed the queue backlog, i.e., $d_{ij}^s \le t_{ij}(s)$.

Since tasks are queued before processing, the MD's queue backlog directly influences task waiting times [31]. Thus, it is essential to impose strict limits on the task queue length of each MD to ensure that task processing is completed within a manageable time frame, i.e.,

$$\lim_{S\to\infty} \frac{1}{S}\sum_{s=0}^{S-1} \mathbb{E}\left\{t_{ij}(s)\right\} < \varepsilon_{ij},\ \forall i \in \mathcal{M}, \forall j \in \mathcal{N}, \tag{2}$$

where ε_{ij} is the queue perturbation parameter.

B. Computation and Energy Models

1) Local Computing: Recall that $x_{ij} = 0$ indicates the MD process tasks locally. For the j-th MD in the i-th EN, we use f_{ij}^s (in MHz) to denote the local computation frequency, with an upper bound of $f_{ij}^{\max}$. ϕ to represent the number of cycles required to compute 1 bit. Thus, in time slot s, the tasks processed locally and the energy consumption generated are

$$d_{ij}^{1,s} = (1 - x_{ij}) \times \frac{f_{ij}^s \tau}{\phi}, \tag{3}$$

$$e_{ij}^{1,s} = (1 - x_{ij}) \times \xi {f_{ij}^s}^3 \tau, \tag{4}$$

where ξ is the effective capacitance co-efficient of CPU.

2) Remote Offloading: Recall that $x_{ij} = 1$ indicates the MDs generated tasks is offloaded for edge execution. Assume each device employs Time Division Multiple Access (TDMA) to send tasks to the MEC server. For the j-th MD in the i-th EN, we use $\tau_{ij}^s\tau$ to denote its offloading time with a constraint of $\tau_{ij}^s \in [0, 1]$, P_{ij}^s to represent the transmit power with an upper bound of $P_{ij}^{\max}$, and h_{ij}^s to indicate the channel gain between the MD and the MEC server. In the block fading model, h_{ij}^s remains fixed within a time frame but changes independently from one frame to the next [32]. Thus, during time frame s, the tasks processed at the MEC server is

$$d_{ij}^{o,s} = x_{ij} \times \tau_{ij}^s \tau W \log_2\left(1 + \frac{h_{ij}^s e_{ij}^{o,s}}{N_0 \tau_{ij}^s \tau}\right), \tag{5}$$

where W is the system bandwidth, and $e_{ij}^{o,s} = P_{ij}^s \tau_{ij}^s \tau$ is the transmission energy consumption. To simplify the discussion, the following derivations assume $\tau = 1$, without loss of generality.

In this study, we denote $d_{ij}^s = (1 - x_{ij}) \times d_{ij}^{o,s} + x_{ij} \times d_{ij}^{o,s}$ and $e_{ij}^s = (1 - x_{ij}) \times e_{ij}^{1,s} + x_{ij} \times e_{ij}^{o,s}$ as the tasks computed and energy consumed in time slot s. Additionally, we define *computation rate* r_{ij}^s and *power consumption* e_{ij}^s in time slot s as

$$r_{ij}^s = (1 - x_{ij})\frac{f_{ij}^s}{\phi} + x_{ij}\tau_{ij}^s W \log_2\left(1 + \frac{e_{ij}^{o,s} h_{ij}^s}{\tau_{ij}^s N_0}\right), \tag{6}$$

$$e_{ij}^s = (1 - x_{ij})\xi {f_{ij}^s}^3 + x_{ij} e_{ij}^{o,s}. \tag{7}$$

We denote by $r_{ij}^{o,s}$ the remote offloading rate, and then we have $r_{ij}^s = (1 - x_{ij}) f_{ij}^s/\phi + x_{ij}\tau_{ij}^s r_{ij}^{o,s}$.

C. Battery and Energy Harvesting Model

Similar to the arrival tasks, the energy arrival process for MDs is i.i.d. over different time slots. We denote by P^s the average energy harvesting power in this scenario, which can be determined via offline measurements [33]. For the j-th MD in the i-th EN, we use H_{ij}^s (in mJ) to denote the harvested energy, with an upper bound of $H_{ij}^{\max}$. In this study, we presume that the energy used by MDs for offloading tasks only comes from the battery and the harvested energy H_{ij}^s in the current time slot can only be used in the next time slot, i.e., harvest-use-storage strategy [34]. Let $b_{ij}(s)$ denote the residual battery energy in time slot s. Hence, it is evident that $b_{ij}(s+1)$ is

$$b_{ij}(s+1) = b_{ij}(s) - e^s_{ij} + H^s_{ij}. \tag{8}$$

Notice that the energy consumed in the end of current time slot don't exceed the residual battery energy, i.e., $e^s_{ij} \leq b_{ij}(s)$. To extend the battery life, its remaining energy should be kept above the minimum level, i.e.,

$$\lim_{S\to\infty} \frac{1}{S}\sum_{s=0}^{S-1} \mathbb{E}\left\{b_{ij}(s)\right\} > \epsilon_{ij},\ \forall i \in \mathcal{M}, \forall j \in \mathcal{N}, \tag{9}$$

where ϵ_{ij} is the energy level perturbation parameter.

D. Computation Rate Maximization Problem

In this study, our objective is to develop an online task offloading algorithm (called LyFOTO) to optimize the long-term average throughput of system, while ensuring tasks queue and battery energy constraints. Here, system throughput generally refers to the weighted computation rate of all MDs. Specifically, we find online decision-making by optimizing task offloading decision and resource allocation scheme in each time frame. This is done without assuming knowledge of future occurrences of random system information. Denote $\boldsymbol{x}^s = \{\boldsymbol{x}^s_i\}^N_{i=1} = \{[x^s_{ij}]^M_{j=1}\}^N_{i=1}$, $\boldsymbol{f}^t = \{\boldsymbol{f}^s_i\}^N_{i=1} = \{[f^s_{ij}]^M_{j=1}\}^N_{i=1}$, $\boldsymbol{\tau}^s = \{\boldsymbol{\tau}^s_i\}^N_{i=1} = \{[\tau^s_{ij}]^M_{j=1}\}^N_{i=1}$, and $\boldsymbol{e}^s = \{\boldsymbol{e}^s_{oi}\}^N_{i=1} = \{[e^{o,s}_{ij}]^M_{j=1}\}^N_{i=1}$, the problem can be expressed as a multi-stage stochastic optimization problem:

$$\mathcal{P}_1: \quad \underset{\boldsymbol{x}^s,\boldsymbol{f}^s,\boldsymbol{\tau}^s,\boldsymbol{e}^s}{\text{maximize}} \lim_{S\to\infty}\frac{1}{S}\sum_{s=0}^{S-1}\mathbb{E}\left\{c_{ij}r^s_{ij}\right\} \tag{10}$$

$$\text{subject to}\ \ \mathrm{C1}: x^s_{ij} \in \{0,1\}; \tag{10a}$$

$$\mathrm{C2}: 0 \leq d^s_{ij} \leq t_{ij}(s); \tag{10b}$$

$$\mathrm{C3}: 0 \leq e^s_{ij} \leq b_{ij}(s); \tag{10c}$$

$$\mathrm{C4}: 0 \leq f^s_{ij} \leq f^{\max}_{ij}; \tag{10d}$$

$$\mathrm{C5}: 0 \leq e^{o,s}_{ij} \leq P^{\max}_{ij}\tau^s_{ij}; \tag{10e}$$

$$\mathrm{C6}: \sum_{j=1}^{M}\tau^s_{ij} \leq 1, \tau^s_{ij} \geq 0; \tag{10f}$$

$$\mathrm{C7}: \lim_{S\to\infty}\frac{1}{S}\sum_{s=0}^{S-1}\mathbb{E}\left\{t_{ij}(s)\right\} < \varepsilon_{ij}; \tag{10g}$$

$$\mathrm{C8}: \lim_{S\to\infty}\frac{1}{S}\sum_{s=0}^{S-1}\mathbb{E}\left\{b_{ij}(s)\right\} > \epsilon_{ij}; \tag{10h}$$

C1−C8 satisfy $i \in \mathcal{N}, j \in \mathcal{M}$ and $s \in \mathcal{S}$.

Here, c_{ij} is the weight of the j-th MD in the i-th EN. C1 denotes the offloading decision. C2 and C3 denote the tasks computed and energy consumed constraint. C4 and C5 correspond to the boundary constraints. C6 denotes the offloading time constraint. C7 and C8 are the long-term stability constraints of the queue and battery.

When offloading schemes are taken within each specific time slot without insight into upcoming random changes, it poses challenges to satisfy long-term constraints. This is particularly true in environments where channels and data arrivals are inherently unpredictable. In addition, the rapidly changing nature of channel conditions necessitates immediate decision-making within very brief time slots. In subsequent sections, we introduce an innovative LyFOTO framework designed to address $\mathcal{P}_1$, ensuring both high levels of robustness and efficiency.

IV. Lyapunov-Based Problem Transformation

In this section, we utilize the perturbation Lyapunov optimization to eliminate queue backlog and battery energy constraints (C7 and C8). The Lyapunov optimization theory introduces the concept of queues and regards pending tasks as queue backlogs. Once the task offloading decision is determined, tasks are queued and processed by the system following FCFS principle. For this purpose, the virtual task and energy queues are introduced, which are defined as

$$T_{ij}(s) = t_{ij}(s) - \varepsilon_{ij}, \tag{11}$$

$$B_{ij}(s) = \epsilon_{ij} - b_{ij}(s), \tag{12}$$

and

$$T_{ij}(s+1) = T_{ij}(s) - d^s_{ij} + a^s_{ij}, \tag{13}$$

$$B_{ij}(s+1) = B_{ij}(s) - H^s_{ij} + e^s_{ij}, \tag{14}$$

for $i \in \mathcal{N}, j \in \mathcal{M}$, and $s \in \mathcal{S}$. Intuitively, when $T \to 0$ and $B \to 0$ of each time slot, the constraints (C7 and C8) are satisfied.

To jointly control the virtual queues, we define $\boldsymbol{Q}^s_i = \{\boldsymbol{T}^s_i, \boldsymbol{B}^s_i\}$ as the Joint queue backlog for each EN, where $\boldsymbol{T}^s_i = \{T_{ij}(s)\}^M_{j=1}$ and $\boldsymbol{B}^s_i = \{B_{ij}(s)\}^M_{j=1}$. Then, we apply the Lyapunov function $L(Q^s_i)$ and Lyapunov drift $\Delta L(Q^s_i)$ as

$$L(\boldsymbol{Q}^s_i) = 0.5\sum_{j=1}^{M}\mathrm{w}_0 T^2_{ij}(s) + 0.5\textstyle\sum_{j=1}^{M}\mathrm{w}_1 B^2_{ij}(s), \tag{15}$$

$$\Delta L(\boldsymbol{Q}^s_i) = \mathbb{E}\{L\left(\boldsymbol{Q}^{s+1}_i\right) - L(\boldsymbol{Q}^s_i))|\boldsymbol{Q}^s_i\}. \tag{16}$$

To optimize the long-term average computation rate while keeping the queue $\boldsymbol{Q}^s_i$ stable, we employ the *drift-plus-penalty* function Δ_V minimization method. The detailed transformation process is illustrated in Fig. 2.

In particular, our goal is to minimize an upper limit of Δ_V for each time slot s

$$\Delta_V(L(\boldsymbol{Q}^s_i)) = \Delta L(\boldsymbol{Q}^s_i) - V\mathbb{E}\left\{c_{ij}r^s_{ij}|\boldsymbol{Q}^s_i\right\} \tag{17}$$

where $V > 0$ is the penalty weight to balance between the queue length and the value of optimization problem. To derive an upper limit of Δ_V in each EN, we have

$$T^2_{ij}(s+1) = T^2_{ij}(s) + 2T_{ij}(s)\left(a^s_{ij} - d^s_{ij}\right) + \left(a^s_{ij} - d^s_{ij}\right)^2,$$

$$B^2_{ij}(s+1) = B^2_{ij}(s) + 2B_{ij}(s)\left(e^s_{ij} - H^s_{ij}\right) + \left(e^s_{ij} - H^s_{ij}\right)^2.$$

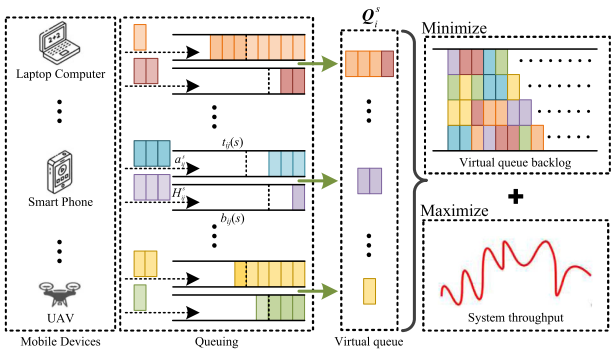

Fig. 2. Transformation process of virtual queue $\boldsymbol{Q}_i^s$.

Accordingly, we can obtain

$$\frac{\mathrm{w}_0}{2}\sum_{j=1}^{M} T_{ij}^2(s+1) - \frac{\mathrm{w}_0}{2}\sum_{j=1}^{M} T_{ij}^2(s) = \frac{\mathrm{w}_0}{2}\sum_{j=1}^{M}\left(a_{ij}^s - d_{ij}^s\right)^2 + \mathrm{w}_0\sum_{j=1}^{M} T_{ij}(s)\left(a_{ij}^s - d_{ij}^s\right), \quad (18)$$

$$\frac{\mathrm{w}_1}{2}\sum_{j=1}^{M} B_{ij}^2(s+1) - \frac{\mathrm{w}_1}{2}\sum_{j=1}^{M} B_{ij}^2(s) = \frac{\mathrm{w}_1}{2}\sum_{j=1}^{M}\left(e_{ij}^s - H_{ij}^s\right)^2 + \mathrm{w}_1\sum_{j=1}^{M} B_{ij}(s)\left(e_{ij}^s - H_{ij}^s\right). \quad (19)$$

Applying the conditional expectation to both sides of (18) and (19), we can obtain

$$C1 + \mathrm{w}_0\sum_{j=1}^{M} T_{ij}(s)\mathbb{E}\left[\left(a_{ij}^s - d_{ij}^s\right)|\boldsymbol{Q}_i^s\right], \quad (20)$$

and

$$C2 + \mathrm{w}_1\sum_{j=1}^{M} B_{ij}(s)\mathbb{E}\left[\left(e_{ij}^s - H_{ij}^s\right)|\boldsymbol{Q}_i^s\right]. \quad (21)$$

Here, $C1$ and $C2$ are the fixed values derived as follows

$$\frac{\mathrm{w}_0}{2}\sum_{j=1}^{M}\mathbb{E}\left[(a_{ij}^s - d_{ij}^s)^2\right] \le \frac{\mathrm{w}_0}{2}\sum_{j=1}^{M}\mathbb{E}\left[(a_{ij}^s)^2 + (d_{ij}^s)^2\right] \le \frac{\mathrm{w}_0}{2}\sum_{j=1}^{M}\left(\eta_{ij} + \left[\tau\max\{f_{ij}^s\tau/\phi, r_{ij}^{\max}\}\right]^2\right) \triangleq C1,$$

$$\frac{\mathrm{w}_1}{2}\sum_{j=1}^{M}\mathbb{E}\left[(e_{ij}^s - H_{ij}^s)^2\right] \le \frac{\mathrm{w}_1}{2}\sum_{j=1}^{M}\mathbb{E}\left[(e_{ij}^s)^2 + (H_{ij}^s)^2\right] \le \frac{\mathrm{w}_1}{2}\sum_{j=1}^{M}\left(\left[\tau\max\left\{\xi(f_{ij}^{\max})^3, P_{ij}^{\max}\right\}\right]^2 + \left(H_{ij}^{\max}\right)^2\right) \triangleq C2,$$

where $r_{ij}^{\max} \triangleq \mathbb{E}\left[W\log_2\left(1 + \frac{P_{ij}^{\max}h_{ij}^s}{N_0}\right)\right]$. Adding the inequalities presented in (20) and (21) together, we have

$$\Delta L(\boldsymbol{Q}_i^s) \le C + \mathrm{w}_0\sum_{j=1}^{M} T_{ij}(s)\mathbb{E}\left[\left(a_{ij}^s - d_{ij}^s\right)|\boldsymbol{Q}_i^s\right] + \mathrm{w}_1\sum_{j=1}^{M} B_{ij}(s)\mathbb{E}\left[\left(e_{ij}^s - H_{ij}^s\right)|\boldsymbol{Q}_i^s\right], \quad (22)$$

where $C = C1 + C2$. Hence, $\Delta_V(L(\boldsymbol{Q}_i^s))$ is expression in

$$C + \mathrm{w}_0\sum_{j=1}^{M} T_{ij}(s)\mathbb{E}\left[(a_{ij}^s - d_{ij}^s)|\boldsymbol{Q}_i^s\right] + \mathrm{w}_1\sum_{j=1}^{M} B_{ij}(s)\mathbb{E}\left[(e_{ij}^s - H_{ij}^s)|\boldsymbol{Q}_i^s\right] - V\mathbb{E}\left\{c_{ij}r_{ij}^s|\boldsymbol{Q}_i^s\right\}. \quad (23)$$

During time slot s, we employ the *opportunistic expectation minimization* technique as proposed in [35]. Specifically, we monitor the queue backlogs Q_i^s and determine the joint offloading and resource allocation control strategy based on these observations, aiming to minimize the upper bound specified in (23). After excluding the constant terms from the observation made at the start of the s-th time slot, the algorithm determines the optimal actions by maximizing the following expression,

$$\sum_{i=1}^{N}\sum_{j=1}^{M}(\mathrm{w}_0 T_{ij}(s) + V c_{ij})r_{ij}^s - \sum_{i=1}^{N}\sum_{j=1}^{M}\mathrm{w}_1 B_{ij}e_{ij}^s.$$

Intuitively, the above equation aims to enhance the computation rate for tasks with significant backlogs beyond the disturbance queue parameter or for devices with high weight. Meanwhile, it penalizes devices whose battery energy levels fall below the disturbance energy threshold. Let $\boldsymbol{r}^s = \{\boldsymbol{r}_{oi}^s\}_{i=1}^N = \{[r_{ij}^{o,s}]_{j=1}^M\}_{i=1}^N$. Utilizing the Lyapunov *drift plus penalty* method, problem $\mathcal{P}_1$ is reformulated into a deterministic sub-problem $\mathcal{P}_2$ for each time slot.

$$\mathcal{P}_2: \underset{\boldsymbol{x}^s,\boldsymbol{f}^s,\boldsymbol{\tau}^s,\boldsymbol{e}^s,\boldsymbol{r}^s}{\text{maximize}} \sum_{i=1}^{N}\sum_{j=1}^{M}(\mathrm{w}_0 T_{ij}(s) + V c_{ij})r_{ij}^s - \sum_{i=1}^{N}\sum_{j=1}^{M}\mathrm{w}_1 B_{ij}e_{ij}^s$$
$$\text{subject to } \ \mathrm{C1} - \mathrm{C6}. \quad (24)$$

Next, the main challenge that remains is solving the problem $\mathcal{P}_2$ within each time slot. To tackle this, the next section introduces a FL-based algorithm designed to efficiently solve it.

V. The LyFOTO Algorithm Details

In this section, we design a novel Lyapunov-guided federated DRL online task offloading algorithm to address $\mathcal{P}_2$.

Algorithm 1: The Proposed LyFOTO Algorithm for Solving $\mathcal{P}_2$.

```
Input: System scenario parameters;
Output: Offloading actions $\{\boldsymbol{x}^s, \boldsymbol{y}^s\}_{s=1}^S$;
1  Randomly initialize the weights $\{\boldsymbol{\theta}_i\}_{i=1}^N$ and $\{\boldsymbol{\theta}'_i\}_{i=1}^N$ of the
   current action network, and empty experience replay buffer;
2  Empty initial task and battery queue $t_{ij}(1) = b_{ij}(1) = 0$ for
   $i = 1, \dots, N$ and $j = 1, \dots, M$;
3  for $s = 1, 2, \dots, S$ do
4      for $i$ = 1 to $N$ do
5          Initialize virtual queues $T_{ij}(t)$ and $B_{ij}(t)$ according to
           (11) and (12) for each device $j$;
6          Obtain actions $\boldsymbol{x}_i^s$ according to the current state space $\mathcal{S}_i$;
7          Quantize offloading actions $\hat{x}_i^s$ using the NOP method [37];
8          Calculate $F\left(\boldsymbol{x}_i^s, \boldsymbol{\zeta}_i^s\right)$ by determining the optimal resource
           distribution $\boldsymbol{y}_i^s$ in P2 for each $\hat{x}_i^s$;
9          Choose the optimal strategy $\left(\boldsymbol{x}_i^s\right)^*$ according to $\mathcal{P}_3$ and
           execute the joint action $\left[\left(\boldsymbol{x}_i^s\right)^*, \boldsymbol{y}_i^s\right]$;
10         Store the joint action $\left[\left(\boldsymbol{x}_i^s\right)^*, \boldsymbol{y}_i^s\right]$ in experience replay
           buffer;
11         if mod $(s, \gamma_s) = 0$ then
12             Randomly select a batch of sample $\{(\boldsymbol{x}_i^t, \boldsymbol{\zeta}_i^t), t \in \mathcal{Q}\}$
               from the experience replay buffer;
13             Train the DNN with $\{(\boldsymbol{x}_i^t, \boldsymbol{\zeta}_i^t), t \in \mathcal{Q}\}$ and update $\boldsymbol{\theta}_i$
               and $\boldsymbol{\theta}'_i$ using the Eq. (25) and Eq. (26);
14         end
15         Iteration $\{t_{ij}(s+1), b_{ij}(s+1)\}_{j=1}^M$ based on tasks
           arrival observation $a_{ij}^s$, energy harvesting observation
           $H_{ij}^s$, and $\left[\left(\boldsymbol{x}_i^s\right)^*, \boldsymbol{y}_i^s\right]$ using Eqs. (1) and (8);
16     end
17     if mod $(s, \Gamma_s) = 0$ then
18         Send the local model parameters from MEC servers to the
           central cloud server for global aggregation according to
           Eq. (27);
19         Transmit the global model parameters from the central
           cloud server to the MEC servers;
20     end
21     $s \leftarrow s + 1$.
22 end
```

A. Markov Decision Process Model

In the i-th EN, the system information includes the queue states $\boldsymbol{Q}_i^s = \{T_{ij}(s), B_{ij}(s)\}_{j=1}^M$ and the channel gains $\{h_{ij}^s\}_i^s$. Let $\boldsymbol{\zeta}_i^s \triangleq \{h_{ij}^s, T_{ij}(s), B_{ij}(s)\}_{j=1}^M$ and determine the control strategies $\{\boldsymbol{x}_i^s, \boldsymbol{y}_i^s\}$ based on $\boldsymbol{\zeta}_i^s$, where $\boldsymbol{y}_i^s = \{\boldsymbol{f}_i^s, \boldsymbol{\tau}_i^s, \boldsymbol{e}_{\text{o}i}^s, \boldsymbol{r}_{\text{o}i}^s\}$. The task offloading and resource allocation problem can be modeled as Markov Decision Process (MDP) [36]. The main components of the i-th EN are outlined below.

- *State Space:* In time slot s, the sate space $\mathcal{S}_i$ is represented as the system information, i.e., $\mathcal{S}_i = \zeta_i^s$.
- *Action Space:* In time slot s, the action space $\mathcal{A}_i$ is represented as the offloading strategies $\boldsymbol{x}_i^s$. After the offloading strategies $\boldsymbol{x}_i^s$ is obtained, we apply the noisy order-preserving (NOP) quantization method to generate $g_i^s \le 2M$ groups of candidate strategies [37].
- *Immediate Reward:* Since Problem $\mathcal{P}_2$ is a maximization problem, the reward function $R_i = (\mathrm{w}_0 T_{ij}(s) + V c_{ij}) r_{ij}^s - \mathrm{w}_1 B_{ij} e_{ij}^s$.

B. FL-Based Training Framework

Fig. 3 illustrates the framework of the proposed LyFOTO algorithm. A full global iteration consists of the following steps. First, the server distributes the global model to each MEC server for edge training. Next, the MEC servers handle model training using the local data within their respective cells and subsequently upload the updated models to the global server for global aggregation, which leads to the formation of the updated global model.

1) Edge Training: Each MEC server is responsible for edge training within its respective cell and functions as an agent, which depicted in Fig. 3. This agent consists of two key modules: the actor module, which includes two networks θ and θ', and the critic module, which utilizes a function. Notice that the use of two actor networks is intended to address the overfitting issue. The actor module selects offloading actions based on the current state, applying the NOP quantization method to generate up to $g_i^s \le 2M$ groups of candidate actions [37]. The critic module evaluates these actions and identifies the optimal one, which is described as follows:

$$\mathcal{P}_3: \quad \left(\boldsymbol{x}_i^s\right)^* = \arg \underset{\boldsymbol{x}_i^s \in \{0,1\}^M}{\text{maximize}} \ F\left(\boldsymbol{x}_i^s, \boldsymbol{\zeta}_i^s\right),$$

where $F(\boldsymbol{x}_i^s, \boldsymbol{\zeta}_i^s)$ is the optimal value of $\mathcal{P}_2$ by optimizing $\boldsymbol{y}_i^s$ given the actions and the current state.

The LyFOTO algorithm utilizes $\left(\boldsymbol{x}_i^s, \boldsymbol{\zeta}_i^s\right)$ as a labeled I/O sample for the update of θ'. Specifically, we set up an experience replay buffer to store recent q data samples. In practice, training of θ' begins once more than $q/2$ samples have been collected in the initially empty experience replay buffer. θ' is then updated periodically for every γ_s time slot to prevent over-fitting problem of the model. Furthermore, soft target updating (25) is employed to ensure gradual adjustments to the network's weight parameters, thereby improving the stability of the learning process.

$$\theta = \alpha\theta + (1 - \alpha)\theta'. \tag{25}$$

Every δ_s time slot, LyFOTO randomly selects a batch of data samples $\{(\boldsymbol{x}_i^t, \boldsymbol{\zeta}_i^t), t \in \mathcal{Q}\}$ and uses the Adam algorithm [38] to minimize its binary cross entropy (BCE) loss function $\mathcal{L}_i(s, \theta')$ on the data samples to update θ'.

$$\begin{aligned} \mathcal{L}_i(s, \theta') = &-\frac{1}{|\mathcal{Q}^s|} \cdot \sum_{t \in \mathcal{Q}^s} \left[(\boldsymbol{x}^t)^\intercal \log \Pi_{\theta'}(\boldsymbol{\zeta}_i^t)\right. \\ &\left. + (1 - \boldsymbol{x}^t)^\intercal \log\left(1 - \Pi_{\theta'}(\boldsymbol{\zeta}_i^t)\right)\right]. \end{aligned} \tag{26}$$

When the training completes, we update the parameter of the actor network θ according to (25) at the beginning of the next slot.

2) Global Aggregation: Each MEC server trains the edge model utilizing its local data and then uploads the revised model parameters to the global server, where they are aggregated to form the global model. Repeat this process every Γ_s time slot. Specifically, the global model is obtained through (27).

$$\boldsymbol{\theta} = \frac{1}{\sum_{i=1}^N d_i} \sum_{i=1}^N d_i \cdot \boldsymbol{\theta}_i, \tag{27}$$

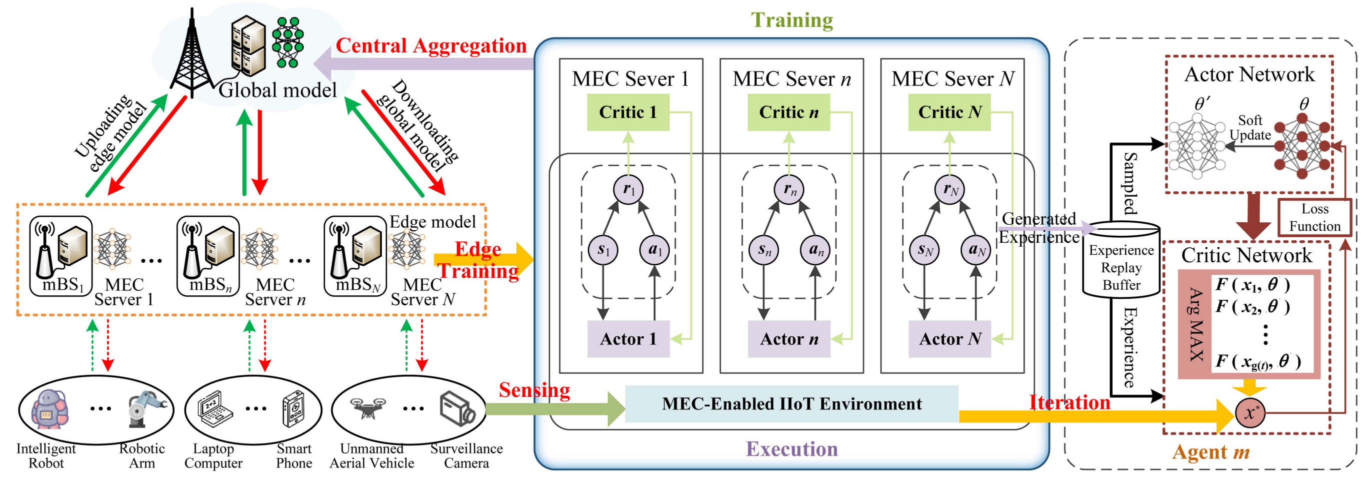

Fig. 3. The framework of the proposed LyFOTO algorithm.

where $\boldsymbol{\theta}$ represents the global model parameters aggregated by the global server, and d_i refers to the number of training samples from MEC server i. Algorithm 1 outlines the operational process of the proposed LyFOTO algorithm.

C. Fast Near-Optimal Resource Distribution Algorithm

With the value of $\boldsymbol{x}_i^s$ in $\mathcal{P}_2$, let $\mathcal{M}_1^s$ represent the index set of MDs where $x_{ij}^s = 1$, and $\mathcal{M}_0^s$ represent the index set of MDs where $x_{ij}^s = 0$. For simplicity, the superscript s and the subscript i are omitted. Thus, $F(\boldsymbol{x}_i^s, \boldsymbol{\zeta}_i^s)$ is expressed as follows

$$\mathcal{P}_4: \underset{\boldsymbol{f}^s,\boldsymbol{\tau}^s,\boldsymbol{e}^s,\boldsymbol{r}^s}{\text{maximize}} \sum_{j\in\mathcal{M}_0^s} \left\{(\mathrm{w}_0 T_j(s)+Vc_i)\frac{f_j}{\phi} - \mathrm{w}_1 B_j(s)\xi f_j^3\right\} + \sum_{j\in\mathcal{M}_1^s}\left\{(\mathrm{w}_0 T_j(s)+Vc_i)r_j^{\mathrm{o}} - \mathrm{w}_1 B_j(s) e_j^{\mathrm{o}}\right\}$$
$$\text{subject to } \ \mathrm{C2}-\mathrm{C6}.$$

Be aware that $\mathcal{P}_4$ can be optimized independently for MDs in both $\mathcal{M}_0^s$ and $\mathcal{M}_1^s$. Specifically, each MD $j \in \mathcal{M}_0^s$ addresses its parallel sub-problem

$$\mathcal{P}_5: \underset{f_j}{\text{maximize}} \ (\mathrm{w}_0 T_j(s)+Vc_i)\frac{f_j}{\phi} - \mathrm{w}_1 B_j(s)\xi f_j^3$$
$$\text{subject to } \ 0 \le f_j \le \min\left\{\phi t_j(s), \sqrt[3]{b_i(s)/\xi}, f_j^{\max}\right\};$$

Its optimal strategy is given by (28) shown at the bottom of this page, where $\mathrm{w}_2 = \mathrm{w}_0 T_j(s) + Vc_j$, and

$$\overline{f} = \begin{cases} \sqrt{\frac{\mathrm{w}_0 T_j(s)+Vc_i}{\mathrm{w}_1 \xi B_j(s)}}, \\ \quad \text{if } \min\left\{\phi t_j(s), \sqrt[3]{b_i(s)/\xi}, f_j^{\max}\right\} < \sqrt{\frac{\mathrm{w}_0 T_j(s)+Vc_i}{\mathrm{w}_1 \xi B_j(s)}}; \\ 0, \\ \quad \text{otherwise}; \end{cases}$$

Clearly, the computation rate of the MD j depends on the state of virtual queues T and B.

Next, it is necessary to address the following problem for the WD $j \in \mathcal{M}_1^s$,

$$\underset{\boldsymbol{\tau}_{\mathrm{o}}^s,\boldsymbol{e}_{\mathrm{o}}^s,\boldsymbol{r}_{\mathrm{o}}^s}{\text{maximize}} \sum_{j\in\mathcal{M}_1^s}\left\{(\mathrm{w}_0 T_j(s)+Vc_i)r_j^{\mathrm{o}} - \mathrm{w}_1 B_j(s)e_j^{\mathrm{o}}\right\} \quad (29)$$

$$\text{subject to } \sum_{j\in\mathcal{M}_1^s} \tau_j \le 1, \tau_j \ge 0; \quad (29a)$$

$$r_j^{\mathrm{o}} \le t_j(s), e_j^{\mathrm{o}} \le b_j(s), e_j^{\mathrm{o}} \le P_j^{\max}\tau_j^s, \forall j \in \mathcal{M}_1^s; \quad (29b)$$

$$r_j^{\mathrm{o}}, e_j^{\mathrm{o}} \ge 0, \forall j \in \mathcal{M}_1^s; \quad (29c)$$

where $\boldsymbol{\tau} = \{\tau_j, \forall j \in \mathcal{M}_1^s\}$, $\boldsymbol{e}^{\mathrm{o}} = \{e_j^{\mathrm{o}}, \forall j \in \mathcal{M}_1^s\}$, and $\boldsymbol{r}^{\mathrm{o}} = \{r_j^{\mathrm{o}}, \forall j \in \mathcal{M}_1^s\}$.

To solve this problem, we introduce a partial Lagrangian function, which is

$$L(\boldsymbol{\tau}, \boldsymbol{e}^{\mathrm{o}}, \boldsymbol{r}^{\mathrm{o}}; \mu) = \sum_{j\in\mathcal{M}_1^s}\left\{(\mathrm{w}_0 T_j(s)+Vc_i)\, r_j^{\mathrm{o}} - \mathrm{w}_1 B_j(s)e_j^{\mathrm{o}}\right\} + \mu\left(1 - \sum_{j\in\mathcal{M}_1^s}\tau_j\right),$$

$$f_j^* = \begin{cases} \min\left\{\sqrt{\frac{\mathrm{w}_0 T_j(s)+Vc_i}{3\mathrm{w}_1\phi\xi B_j(s)}}, \phi t_j(s), \sqrt[3]{b_j(s)/\xi}, f_j^{\max}\right\}, & \{T_j(s)\ge 0 \text{ and } B_j(s)\ge 0\} \text{ or } \{T_j(s)<0, B_j(s)\ge 0 \text{ and } \mathrm{w}_2 \ge 0\}; \\ \min\left\{\phi t_j(s), \sqrt[3]{b_j(s)/\xi}, f_j^{\max}\right\}, & \{T_j(s)\ge 0 \text{ and } B_j(s)< 0\} \text{ or } \{T_j(s)<0, B_j(s)< 0 \text{ and } \mathrm{w}_2 \ge 0\}; \\ 0, & T_j(s)<0, B_j(s)\ge 0 \text{ and } \mathrm{w}_2 < 0; \\ \overline{f}_j, & T_j(s)<0, B_j(s)< 0 \text{ and } \mathrm{w}_2 < 0; \end{cases} \quad (28)$$

where μ indicates the dual variable, and the dual problem is

$$\underset{\mu\geq 0}{\text{minimize}}\ \mathcal{D}(\mu)$$

Here, the dual function is

$$\begin{aligned}\mathcal{D}(\mu) = \ & \underset{\boldsymbol{\tau},\boldsymbol{e}^{\text{o}},\boldsymbol{r}^{\text{o}}}{\text{maximize}}\ L(\boldsymbol{\tau},\boldsymbol{e}^{\text{o}},\boldsymbol{r}^{\text{o}};\mu)\\ \text{subject to}\ \ & r_j^{\text{o}} \leq t_j(s), \forall j \in \mathcal{M}_1^s;\\ & e_j^{\text{o}} \leq b_j(s), e_j^{\text{o}} \leq P_j^{\max}\tau_j^s, \forall j\in\mathcal{M}_1^s;\\ & \tau_j, r_j^{\text{o}}, e_j^{\text{o}} \geq 0, \forall j \in \mathcal{M}_1^s.\end{aligned}$$

Observe that $\mathcal{D}(\mu)$ can be broken down into independent subproblems. Specifically, each MD $j \in \mathcal{M}_1^s$ solves

$$\begin{aligned}\mathcal{P}_6:\ & \underset{\tau_j, e_j^{\text{o}}, r_j^{\text{o}}}{\text{maximize}}\ \mathrm{w}_2 r_j^{\text{o}} - \mathrm{w}_1 B_j(s) e_j^{\text{o}} - \mu\tau_j\\ & \text{subject to } 0 \leq r_j^{\text{o}} \leq t_j(s), \tau_j \geq 0;\\ & \qquad 0 \leq e_j^{\text{o}} \leq \min\{b_j(s), P_j^{\max}\tau_j^s\};\end{aligned}$$

Recall that

$$\frac{r_j^{\text{o}}}{\tau_j} = W\log_2\left(1+\frac{e_j^{\text{o}}h_j}{\tau_j N_0}\right) = W\log_2\left(1+\frac{P_j h_j}{N_0}\right), \quad (30)$$

with an upper bound of

$$W\log_2\left(1+\min\left\{b_j(s), P_j^{\max}\right\}\times\frac{h_j}{N_0}\right) \triangleq r_j^{\max}.$$

Here, $r_j^{\max}$ represents the maximum offloading rate. Notice that $b_j(s)/\tau_j$ is simplified to $b_j(s)$ in the min function, because in most cases, $b_j(s)/\tau_j > P_j^{\max}$, making the ratio irrelevant in the minimization.

According to (30), e_j^{o} as a function of r_j^{o} and τ_j, i.e.,

$$e_j^{\text{o}}\left(\frac{r_j^{\text{o}}}{\tau_j}\right) \triangleq \frac{N_0\tau_j}{h_j}\left(\exp\left(\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}\right)-1\right), \quad (31)$$

Thus, $\mathcal{P}_6$ is equivalent to $\mathcal{P}_{6\text{a}}$

$$\mathcal{P}_{6\text{a}}: \underset{r_j^{\text{o}}}{\text{maximize}}\left\{G_i(r_j^{\text{o}}) | 0 \leq r_j^{\text{o}} \leq t_j(s)\right\}$$

where

$$\begin{aligned}\mathcal{P}_{6\text{b}}: G_i(r_j^{\text{o}}) \triangleq \max_{\tau_j}\ & \mathrm{w}_2 r_j^{\text{o}} - \mu\tau_j - \mathrm{w}_1 B_j(s) e_j^{\text{o}}\left(\frac{r_j^{\text{o}}}{\tau_j}\right)\\ & \text{subject to } \tau_j \geq r_j^{\text{o}}/r_j^{\max}.\end{aligned} \quad (32)$$

By taking the second derivative of (32), we can determine that $\mathcal{P}_{6\text{b}}$ is a convex optimization problem. Its optimal strategy is given by Proposition 1.

Proof: Given r_j^{o}, we denote by $\chi(\tau_j) = \mathrm{w}_2 r_j^{\text{o}} - \mu\tau_j - \mathrm{w}_1 B_j(s) e_j^{\text{o}}\left(\frac{r_j^{\text{o}}}{\tau_j}\right)$. Within the feasible set $\tau_j \geq r_j^{\text{o}}/r_j^{\max}$, we can see $\chi(\tau_j)$ is a concave function. The minimum value is obtained either at the boundary point $r_j^{\text{o}}/r_j^{\max}$ or at the point x_0 that satisfies $\chi'(x_0) = 0$.

By taking the derivative of $\chi(x_0)$, we have

$$\begin{aligned}\chi'(\tau_j) &= -\mu\\ &\quad - \frac{\mathrm{w}_1 B_j(s) N_0}{h_j}\left(e^{\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}} - 1 - e^{\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}}\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}\right)\\ &= -\frac{\mathrm{w}_1 B_j(s) N_0}{h_j}\\ &\quad \times\left[e^{\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}}\left(1-\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}\right) - 1 + \frac{\mu h_j}{\mathrm{w}_1 B_j(s) N_0}\right]\\ &= -\frac{\mathrm{w}_1 B_j(s) N_0 e}{h_j}\\ &\quad \times\left[\frac{\mu h_j e^{-1}}{\mathrm{w}_1 B_j(s) N_0} - e^{-1} - e^{\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}-1}\left(\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}-1\right)\right].\end{aligned} \quad (33)$$

To find x_0, we set $\chi'(\tau_j) = 0$, i.e.,

$$e^{\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}-1}\left(\frac{\ln 2}{W}\cdot\frac{r_j^{\text{o}}}{\tau_j}-1\right) = e^{-1}\left(\frac{\mu h_j}{\mathrm{w}_1 B_j(s) N_0}-1\right) \quad (34)$$

Due to $e^{-1}\left(\frac{\mu h_j}{\mathrm{w}_1 B_j(s) N_0}-1\right) \geq -1$, we can get the value of x_0, which is,

$$x_0 = \frac{\ln 2\cdot r_j^{\text{o}}}{W\cdot\left[\mathcal{W}\left(e^{-1}\left[\frac{\mu h_j}{\mathrm{w}_1 B_j(s) N_0}-1\right]\right)+1\right]}. \quad (35)$$

Notice that $\mathcal{W}(\cdot)$ is the Lambert-W function.

If $x_0 < r_j^{\text{o}}/r_j^{\max}$ or $\chi'(\tau_j) = 0$ is unattainable within the feasible set, we have $\tau_j^* = r_j^{\text{o}}/r_j^{\max}$ according to the concave function definition. Given $\chi'(\tau_j) = 0$, $x_0 < r_j^{\text{o}}/r_j^{\max}$ is equal to $\chi'(\tau_j) < 0$. By replacing τ_j with $r_j^{\text{o}}/r_j^{\max}$ in (34), we have $x_0 < r_j^{\text{o}}/r_j^{\max}$ when

$$\begin{aligned}&\mu + \mathrm{w}_1 B_j(s)\min\left\{b_j(s), P_j^{\max}\right\}\\ &\quad\times\left[1-\ln(1+u_j)\left(1+\frac{1}{u_j}\right)\right] > 0\\ &\Rightarrow \ln(1+u_j) \leq \left(1+\frac{\mu}{\mathrm{w}_1 B_j(s)\min\left\{b_j(s), P_j^{\max}\right\}}\right)\\ &\qquad\times\left(1-\frac{1}{1+u_j}\right)\\ &\Rightarrow \ln\left(\frac{1}{1+u_j}\right) \geq -l_j + \frac{l_j}{1+u_j}\\ &\Rightarrow \exp\left(-\frac{l_j}{1+u_j}\right)\left(-\frac{l_j}{1+u_j}\right) \leq -l_j\exp(-l_j),\end{aligned} \quad (36)$$

where

$$u_j \triangleq \frac{h_j\min\left\{b_j(s), P_j^{\max}\right\}}{N_0},$$

$$l_j \triangleq 1 + \frac{\mu}{\mathrm{w}_1 B_j(s) \min\left\{b_j(s), P_j^{\max}\right\}}.$$

Since $-e^{-1} \leq -l_j \exp(-l_j) \leq 0$, the inequality above is equal to

$$\left(-\frac{l_j}{1+u_j}\right) \leq \mathcal{W}\left(-l_j \exp\left(-l_j\right)\right) \in [-1, 0]. \quad (37)$$

The equivalence holds since $\mathcal{W}(x)$ is a monotonically increasing function of x when $x \geq -e^{-1}$. Therefore, we can obtain the optimal solution $\tau_j^* = r_j^{\mathrm{o}}/r_j^{\max}$ from (37) when $h_i \leq \frac{N0}{\min\{b_j(s), P_j^{\max}\}}\left(\frac{l_j}{-\mathcal{W}(-l_j \exp(-l_j))} - 1\right)$. Otherwise, we can obtain that $\tau_j^* = x_0 \geq r_j^{\mathrm{o}}/r_j^{\max}$ and $\chi'(\tau_j) = 0$. ■

Proposition 1: The optimal solution of $\mathcal{P}_{6\mathrm{b}}$ is (38) shown at the bottom of this page, where $l_j = 1 + \frac{\mu}{\mathrm{w}_1 B_j(s) \min\{b_j(s), P_j^{\max}\}}$.

Hence, $\mathcal{P}_{6\mathrm{b}}$ can be rewritten as

$$\mathcal{P}_{6\mathrm{c}}: \underset{r_j^{\mathrm{o}}}{\text{maximize}} \left\{\mathrm{w}_2 - \mathrm{w}_1 B_j(s) \frac{e_j^{\mathrm{o}}\left[\mathcal{J}_i(\mu)\right]}{\mathcal{J}_i(\mu)} - \frac{\mu}{\mathcal{J}_i(\mu)}\right\} r_j^{\mathrm{o}}$$
$$\text{subject to } 0 \leq r_j^{\mathrm{o}} \leq t_j(s),$$

where the optimal policy is

$$r_j^* = \begin{cases} t_j(s), & \text{if } \mathrm{w}_2 - \mathrm{w}_1 B_j(s) \frac{e_j^{\mathrm{o}}[\mathcal{J}_i(\mu)]}{\mathcal{J}_i(\mu)} - \frac{\mu}{\mathcal{J}_i(\mu)} \geq 0; \\ 0, & \text{otherwise}; \end{cases} \quad (39)$$

After calculating r_j^*, we can obtain $\tau_j^* = r_j^*/\mathcal{J}_i(\mu)$, and then compute the value of μ according to $1 - \sum_{j \in \mathcal{M}_1^s} \tau_j^*$. To compute the optimal dual variable μ^*, we use the bi-section search method to solve the approximation until the specified accuracy is satisfied.

Next,the primal problem (29) is simplified as a linear programming, which is

$$\underset{r_j^{\mathrm{o}}}{\text{maximize}} \sum_{j \in \mathcal{M}_1^s} \left\{\mathrm{w}_2 - \mathrm{w}_1 B_j(s) \frac{e_j^{\mathrm{o}}\left[\mathcal{J}_i(\mu^*)\right]}{\mathcal{J}_i(\mu^*)}\right\} r_j^{\mathrm{o}}$$
$$\text{subject to } \sum_{j \in \mathcal{M}_1^s} \frac{r_j^{\mathrm{o}}}{\mathcal{J}_i(\mu^*)} \leq 1; r_j^{\mathrm{o}} \leq t_j(s), \forall j \in \mathcal{M}_1^s \quad (40)$$

We find its optimal solution r_j^* easily using convex optimization tools. Hence, the solution of τ_j and e_j^{o} in (29) is

$$\tau_j^* = r_j^*/\mathcal{J}_i(\mu^*), e_j^{\mathrm{o}*} = \tau_j^* \frac{e_j^{\mathrm{o}}\left[\mathcal{J}_i(\mu^*)\right]}{\mathcal{J}_i(\mu^*)}, \forall j \in \mathcal{M}_1^s. \quad (41)$$

The pseudo-code of the algorithm to solve $\mathcal{P}_4$ is outlined in Algorithm 2.

Algorithm 2: Fast Optimal Resource Distribution Algorithm for Solving $\mathcal{P}_4$.

```
   Input: $\boldsymbol{x}_i^s, \boldsymbol{\zeta}_i^s = \{h_{ij}^s, T_{ij}(s), B_{ij}(s)\}_{j=1}^M$;
1  initialize $\mu_0 \leftarrow$ adequately small value $o$, $Low \leftarrow 0$, $Up \leftarrow$
     adequately large value $O$;
2  Turn $\boldsymbol{x}_i^s$ into $\{\mathcal{M}_0^s, \mathcal{M}_1^s\}$ in $\mathcal{P}_4$;
3  for each $j \in \mathcal{M}_0^s$ do
4      Compute $f_j^*$ according to Eq. (28);
5  end
6  repeat
7      $\mu \leftarrow \frac{Low+Up}{2}$;
8      for each $j \in \mathcal{M}_1^s$ do
9          Compute $\mathcal{J}_i(\mu)$ and $r_j^*$ according to Eqs. (38) and (39);
10         $\tau_j^* \leftarrow r_j^*/\mathcal{J}_i(\mu^*)$;
11     end
12     if $1 - \sum_{j \in \mathcal{M}_1^s} \tau_j^* < 0$ then
13         $Low \leftarrow \mu$;
14     else
15         $Up \leftarrow \mu$;
16     end
17 until $|Up - Low| \leq \mu_0$;
18 $\mu^* \leftarrow \mu$;
19 obtain $\boldsymbol{\tau}_{\mathrm{o}}^s$ and $\boldsymbol{e}_{\mathrm{o}}^s$ according to Eq. (41);
20 return an optimal offloading strategy of $\mathcal{P}_4$.
```

VI. Performance Analysis for LyFOTO

In this section, we first analyze the computational complexity of the proposed algorithm, and then analyze the convergence performance of the proposed algorithm in solving $\mathcal{P}_1$.

A. Computational Complexity

The execution of the LyFOTO algorithm consists of two main components: edge training and federated aggregation. Federated aggregation is performed after a fixed number of time slots, whereas edge training is carried out within each individual time slot. The edge training process includes two stages: the generation of offloading strategies and the updating of these strategies. While strategy updates occur periodically after several time slots, offloading operations are generated within each time slot. Therefore, this paper primarily focuses on analyzing the complexity of offloading strategy generation within a single time slot. A careful examination reveals that the resource allocation step in line 8 of Alg 1 is the primary source of complexity.

The primary complexity of Alg 2 arises from two parts: the binary search applied to μ and the solution of the linear program (LP) in (40). For the first part, the complexity of the binary search is $O\left(M \log_2\left(\frac{\Delta}{\mu_0}\right)\right)$ where μ_0 represents a small positive precision parameter. In addition, the solution accuracy mainly depends on the precision of the dual variable μ. Higher precision values of μ generally provide more accurate solutions, as they capture the impact of the constraints on the optimization process more precisely. However, increasing the precision of μ significantly raises the computational burden, especially during the iterative process, where more computational resources and

$$\tau_j^* = r_j^{\mathrm{o}}/\mathcal{J}_i(\mu) = \begin{cases} \frac{r_j^{\mathrm{o}}}{r_j^{\max}}, & \text{if } h_j \leq \frac{N0}{\min\left\{b_j(s), P_j^{\max}\right\}}\left(\frac{l_j}{-\mathcal{W}(-l_j \exp(-l_j))} - 1\right); \\ \frac{r_j^{\mathrm{o}}}{\left[\mathcal{W}\left(e^{-1}\left[\frac{\mu h_j}{\mathrm{w}_1 B_j(s) N_0} - 1\right]\right) + 1\right] \cdot W/\ln 2}, & \text{otherwise}; \end{cases} \quad (38)$$

time are required to update and adjust the value of μ. The strategy of gradually increasing the precision of μ is commonly adopted to strike a balance between computational efficiency and solution accuracy. For the second part, the LP in (40) is solved using the interior-point method [39], with a complexity of $O(M^3\bar{I})$, where $\bar{I}$ denotes the input length of the binary offloading strategy in problem (40). Compared to directly solving the $\mathcal{P}_4$ with $4 \times M$ variables using the interior-point method, Alg 2 only solves an LP with at most M variables, resulting in a significantly lower computational complexity, especially when M is large. Since the proposed algorithm executes Alg 2 g_i^s times within each time slot s, the overall complexity of the algorithm is $O\left(\left[M\log_2\left(\frac{\Delta}{\mu_0}\right) + M^3\bar{I}\right] g_i^s\right)$.

In contrast to conventional methods, which typically use interior-point methods to directly solve general convex optimization problems involving $4M$ variables, Alg 2 only requires solving a linear programming problem involving M variables. As the value of M increases, representing scenarios with a large number of MD connections, the proposed algorithm significantly reduces computational overhead, thereby improving computational efficiency and system performance. Additionally, to further enhance the scalability of the system, LyFOTO utilizes FL, allowing multiple BS to collaborate in optimizing task offloading decisions without sharing actual data. This decentralized approach reduces computational overhead, enhances privacy, and ensures the system can scale effectively with the increasing number of BS.

In Section VII, simulation results demonstrate that the proposed algorithm achieves significantly short computation times, making it suitable for real-time implementation in dynamic multi-BS collaborative offloading scenarios.

B. Convergence Performance

Before analyzing the asymptotic convergence performance of the proposed algorithm for solving $\mathcal{P}_1$, we first introduce some preliminary work on Lyapunov optimization. In the collaborative offloading scenario, the stochastic events in the optimization problem are represented by an i.i.d. process ω_i^s, which includes task arrivals, energy harvesting, and fading channels, i.e., $\omega_i^s = \{a_{ij}^s, H_{ij}^s, h_{ij}^s\}_{j=1}^M$. We denote by $\mathcal{R}^{\text{opt}}$ the optimal value of $\mathcal{P}_1$ obtained all available strategies. There is a set of control decisions, independent of queue backlog $\boldsymbol{Q}_i^s$, referred to as the *ω-only strategy*, which satisfies the following lemma.

Lemma 1. Given any $\beta > 0$, there is an ω-only strategy Ω that makes control decision $\Xi^{\Omega,s}$, which satisfy,

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{R}^s\left(\Xi^{\Omega,s}\right)\right] \geq \mathcal{R}^{\text{opt}} - \beta,\\
&\mathbb{E}\left[a_{ij}^s\right] \leq \mathbb{E}\left[d_{ij}^s\left(\Xi^{\Omega,s}\right)\right] + \beta,\\
&\mathbb{E}\left[H_{ij}^s\right] \geq \mathbb{E}\left[e_{ij}^s\left(\Xi^{\Omega,s}\right)\right] - \beta, \qquad (42)
\end{aligned}$$

where $\mathcal{R}^s \triangleq \sum_{j=1}^M c_{ij} r_{ij}^s$.

Proof: The proof is provided Theorem 4.5 of [35], which is not described in detail here. ■

Next, Theorem 1 proves the performance of the proposed algorithm.

Theorem 1: Given any queue backlog $\boldsymbol{Q}_i^s$ in time slot s, the upper bound value (23) of *drift plus penalty* is not larger than the constant B. When running the LyFOTO algorithm in each time slot s, the time average computation rate satisfies

$$\lim_{S\to\infty}\frac{1}{S}\sum_{s=0}^{S-1}\mathbb{E}\left\{\mathcal{R}^s\right\} \geq \mathcal{R}^{\text{opt}} - (B+C)/V \qquad (43)$$

Proof: Since $\omega_i(t)$ is an i.i.d. process, We can apply Lemma 1 to (23), i.e.,

$$\begin{aligned}
\Delta_V(L(\boldsymbol{Q}_i^s)) &= C + \mathrm{w}_0\sum_{j=1}^M T_{ij}(s)\mathbb{E}\left[\left(a_{ij}^s - d_{ij}^s\right)|\boldsymbol{Q}_i^s\right]\\
&\quad + \mathrm{w}_1\sum_{j=1}^M B_{ij}(s)\mathbb{E}\left[\left(e_{ij}^s - H_{ij}^s\right)|\boldsymbol{Q}_i^s\right] - V\mathbb{E}\left\{c_{ij}r_{ij}^s|\boldsymbol{Q}_i^s\right\}\\
&\leq B + C + \sum_{j=1}^M\left(\mathrm{w}_0 T_{ij}(s)\mathbb{E}\left[\left(a_{ij}^s - d_{ij}^s\left(\Xi^{\Omega,s}\right)\right)|\boldsymbol{Q}_i^s\right]\right.\\
&\quad \left.+\mathrm{w}_1 B_{ij}(s)\mathbb{E}\left[\left(e_{ij}^s\left(\Xi^{\Omega,s}\right) - H_{ij}^s\right)|\boldsymbol{Q}_i^s\right] - V\mathbb{E}\left\{\mathcal{R}^s\left(\Xi^{\Omega,s}\right)\right\}\right)\\
&\leq B + C + \sum_{j=1}^M\left(\mathrm{w}_0 T_{ij}(s)\mathbb{E}\left[\left(a_{ij}^s - d_{ij}^s\left(\Xi^{\Omega,s}\right)\right)\right]\right.\\
&\quad \left.+\mathrm{w}_1 B_{ij}(s)\mathbb{E}\left[\left(e_{ij}^s\left(\Xi^{\Omega,s}\right) - H_{ij}^s\right)\right] - V\mathbb{E}\left\{\mathcal{R}^s\left(\Xi^{\Omega,s}\right)\right\}\right)\\
&\leq +\beta\left[\sum_{j=1}^M\left(\mathrm{w}_0 T_{ij}(s) + \mathrm{w}_1 B_{ij}(s)\right)\right]\\
&\quad - V\left(\mathcal{R}^{\text{opt}} - \beta\right) \qquad (44)
\end{aligned}$$

Let $\beta \to 0$, we have,

$$\Delta_V(L(\boldsymbol{Q}_i^s)) \leq B + C - V\mathcal{R}^{\text{opt}}. \qquad (45)$$

Summing (45) from $s = 0$ to $S-1$ and then dividing both sides by SV, we obtain,

$$\begin{aligned}
&\frac{1}{SV}\left(\mathbb{E}\left[L(\boldsymbol{Q}_i^S)\right] - \mathbb{E}\left[L(\boldsymbol{Q}_i^0)\right] - V\sum_{s=0}^{S-1}\mathbb{E}\left\{\mathcal{R}^s\right\}\right)\\
&\quad \leq (B+C)/V - \mathcal{R}^{\text{opt}} \qquad (46)
\end{aligned}$$

Since $L(\boldsymbol{Q}_i^S) \geq 0$ and $L(\boldsymbol{Q}_i^0) = 0$, by letting $S \to \infty$ of (46), we have

$$\lim_{S\to\infty}\frac{1}{S}\sum_{s=0}^{S-1}\mathbb{E}\left\{\mathcal{R}^s\right\} \geq \mathcal{R}^{\text{opt}} - (B+C)/V$$

This completes the proof. ■

VII. Results Analysis for LyFOTO

This section conducts a series of experiments to simulate the performance of the LyFOTO algorithm in practical MEC scenarios, while varying key parameters to evaluate system efficiency. The task arrival rate reflects the computation load, usually measured by the amount of data on the task. This corresponds to the amount of data collected for smart city

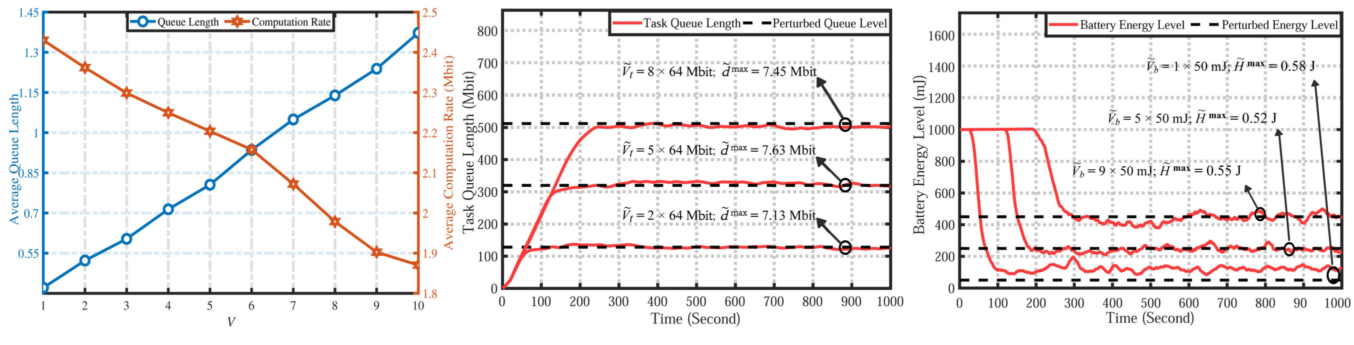

Fig. 4. Impact on system performance with different values of V: (a) Impart of V on the computation rate and queue length; (b) Impart of V on task queue backlog; (c) Impart of V on battery energy level.

scenes. The energy arrival rate reflects the energy harvested by the device. The variation in distance between the device and the base station reflects the user's mobility characteristics, which are typically measured using communication distance. Furthermore, we analyze the system's robustness by adjusting different disturbance parameters.

A. Parameter Settings

In LyFOTO, the actor network is implemented as a five-layer Fully Connected Neural Network (FCNN) with neuron sizes of $[5M, 256, 128, 64, M]$. The *tanh* activation function is applied in the output layer, while the *ReLU* activation function is used across all other layers.

For the scenario parameter setup, $c_{ij} = 1$ if $i + j$ is an even number and $c_{ij} = 1.5$ otherwise. The task arrival rate for all MDs follows an exponential distribution, i.e., $a_{ij}^s \sim \mathrm{E}(\lambda)$ Mbit. The channel gain h_{ij}^s is modeled as an i.i.d. Rician distribution with a line-of-sight link gain equal to $\bar{h}_{ij}^s$ [40], which given by

$$\bar{h}_{ij}^s = 0.9 \times \left(\frac{3 \times 10^8}{4\pi \times 915\text{MHz} \times d_{ij}^s} \right)^3,$$

where d_{ij}^s denotes the distance between BS i and MD j and satisfies Unif [120, 240] m by default. Time slot duration is set to $\tau = 1$ second. The disturbance parameters are defined as

$$\varepsilon_{ij} = \widetilde{V}_t + (\widetilde{d}^{\max})^{-1} = V \times 64\text{ Mbit} + (\widetilde{d}^{\max})^{-1};$$
$$\epsilon_{ij} = \widetilde{V}_b + (\widetilde{H}^{\max})^{-1} = V \times 50\text{ mJ} + (\widetilde{H}^{\max})^{-1}.$$

The following experiments use the default parameter values outlined in Table II, most of which are obtained from [27] and [30].

TABLE II
EXPERIMENTAL PARAMETERS

Parameters	Value	Parameters	Value
N	3	M	5
N_0	-174 dBm/Hz	W	2 MHz
λ	2.5 Mbit/s	S	1000
$f_{ij}^{\max}$	300 MHz	ϕ	100
$P_{ij}^{\max}$	0.1 Watt	ξ	10^{-26}
$H_{ij}^{\max}$	0.6 J	V	5
q	1024	γ_s	12
Q^s	128	Γ_s	64
α	0.4	δ_s	8
w_0	0.3	w_1	0.7

B. Performance Evaluation and Comparative Schemes

1) Computation Rate & Delay Trade-Off: We select several different values of V and compute the average queue backlog and computation rate for each V. The results are illustrated in Fig. 4(a) for comparison. Notice that the average queue length is obtained by normalizing the task queue backlog and battery energy level. Under the condition of a task arrival rate of $\lambda = 2.5$ Mbits, as V increases from 1 to 10 in the step of 1, the average queue length rises from 0.42 to 1.37, while the average computation rate decreases from 2.43 Mbit/s to 1.87 Mbit/s. In Fig. 4(a), the queue backlog increases linearly with V, while the computation rate decreases significantly. This occurs because the trade-off parameter not only balances between drift and penalty but also influences the queue disturbance parameter. Additionally, Fig. 4(a) demonstrates a $[O(1/V), O(V)]$ trade-off between the system's computation rate and queue backlog.

Fig. 4(b) and (c) validate the LyFOTO algorithm's stable control over task queue backlog and battery energy level under varying values of V. Fig. 4(b) illustrates the task queue backlog for $V = 2, 5,$ and 8. It can be observed that the task queue backlog initially increases continuously until stabilizing near the disturbance queue level. This stabilization occurs because, as the backlog approaches the disturbance energy level, the upper bound of the Lyapunov drift plus penalty function in each time slot nears its minimum. Fig. 4(c) presents the battery energy levels for $V = 1, 5,$ and 9. Notice that the battery energy level remains at its initial state initially, then drops sharply and stabilizes around the disturbed energy level. This phenomenon results from the total arrival of tasks being stored in the task queue at the outset, with task processing commencing only when

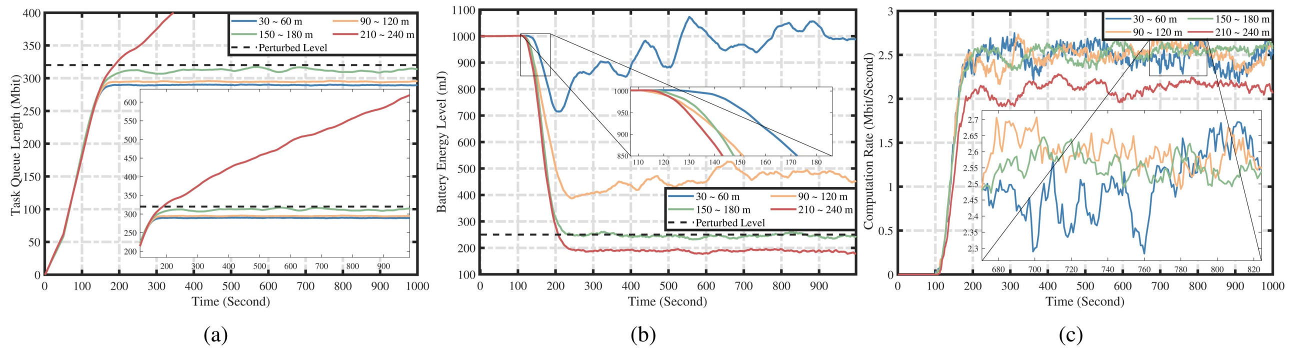

Fig. 5. Impact on system performance with different values of d_{ij}^{s}. (a) Impart of d_{ij}^{s} on task queue backlog. (b) Impart of d_{ij}^{s} on battery energy level. (c) Impart of d_{ij}^{s} on the computation rate.

the queue approaches the disturbance queue level. As the energy level nears this threshold, the upper bound of the Lyapunov drift plus penalty function in each time slot reaches its minimum, prompting the proposed algorithm to prioritize local processing to deplete battery energy until it approaches the disturbance energy level. In Fig. 4, the device's computation rate decreases and the convergence speed of the queue is slower with V increased. Therefore, it is essential to select an appropriate V value based on different scenarios. In the subsequent experiments, we will set V to 5.

2) Impact of Distances d_{ij}^{s}: We evaluate the task offloading performance of MDs within the effective coverage area of a BS, analyzing the average queue backlog and computation rate at different distances d_{ij}^{s}. Fig. 5(a) demonstrates that the task queue backlog quickly converges to the disturbance level. This convergence occurs with a consistent speed when the distance between the device and the BS is below 210 meters. However, when the distance increases to 210-240 meters, the task queue begins to diverge. This is because the effective communication range of the BS is typically around 200 meters. As the distance increases, the quality of the communication channel deteriorates. This degradation hinders efficient task offloading, leading to divergence in the task queue backlog. Fig. 5(b) shows the battery energy levels of MDs at different distances. It can be observed that the battery energy remains at a relatively high level when the device is $30-60$ meters away from the BS. This is because short-distance communication improves channel conditions, enabling the device to offload more tasks within the same time slot. At this point, the number of tasks designated for local offloading in the queue decreases, reducing offloading energy consumption and increasing the battery's remaining energy level. As the distance increases, the communication conditions deteriorate, requiring the device to consume more energy to handle the same number of tasks. At a distance of $150-180$ meters, both the battery energy and task backlog stabilize near the disturbance level. This indicates that the device is nearing the boundary of the effective coverage area of BS. Outside this range, the proposed algorithm continues to strive for stable computation rates and battery energy levels. However, the device's computation rate drops below the task arrival rate. This results in the task queue backlog becoming uncontrollable.

3) Impact of Real-World Task Arrival a_{ij}^{s}: To improve the performance insights of the proposed algorithm in real-world scenarios, we incorporated task instances from the publicly available "cluster-trace-v2017" dataset [1], which were originally generated in a cluster environment. These task instances were then adapted to mobile devices, simulating the task arrival rates for each time slot according to practical conditions. The experimental results are shown in Fig. 6(a)–(c), which illustrate the impact of task arrival rates based on real data on the device's average queue backlog and computation rate under different time slots. To validate the robustness of the proposed algorithm, we multiplied the task arrival rate by a coefficient ι, where $\iota = 0.8$, 1.0, 1.2, 1.5, and 1.8, respectively. As shown in Fig. 6(a), the task queue backlog increases initially and then stabilizes near the disturbance threshold. It is noteworthy that due to significant fluctuations in the task arrival rate in the dataset, when $\iota = 1.8$, the task queue backlog exceeds the preset threshold constraint. Fig. 6(b) illustrates the device's battery energy levels at five different task arrival rates. From Fig. 6(b), it can be observed that the device's battery energy remains above the threshold across all task arrival rates. Fig. 6(c) presents the computation rates of the device at various task arrival rates. The figure indicates that when $\iota = 1.0$, the average computation rate is approximately 1.5 Mbit/s. For comparison, when $\iota = 0.8$, 1.2, 1.5, and 1.8, the average computation rates are approximately 1.2, 1.8, 2.25, and 2.7 Mbit/s, respectively. The experimental results are in general agreement with the trends observed in the computation rates, confirming the robustness and stability of the proposed algorithm.

C. Comparison of Different Offloading Schemes

We conduct a comparison of the LyFOTO algorithm with two additional algorithms to assess the algorithm's practical effectiveness.

- *All Local Execution Algorithm* (ALEA)
  In time slot s, the tasks are all executed locally, and the local main frequency is solved using (28).
- *All Offloading Execution Algorithm* (AOEA)

[1]https://github.com/alibaba/clusterdata/tree/v2018/cluster-trace-v2017

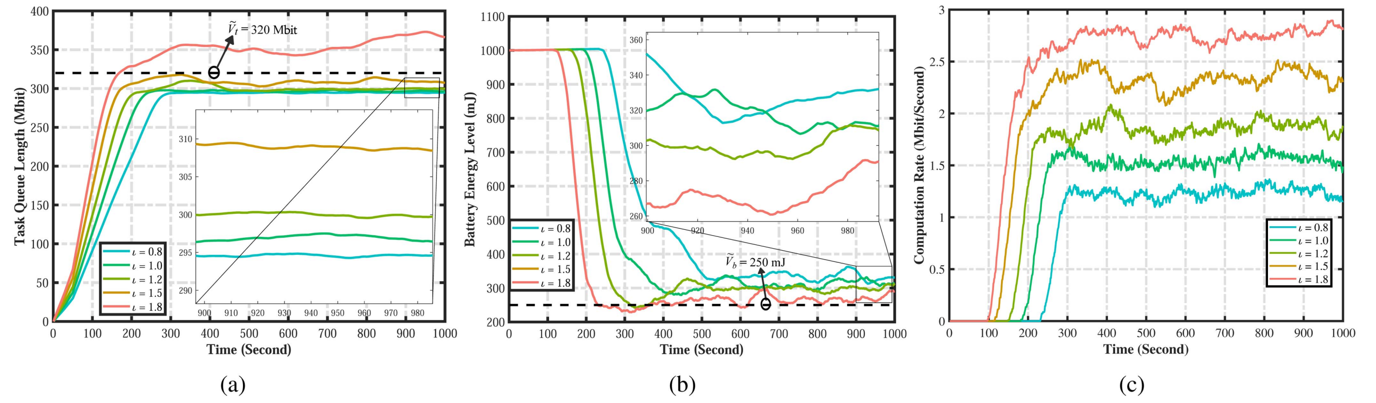

Fig. 6. Impact on system performance with different values of ι. (a) Impart of ι on task queue backlog. (b) Impart of ι on battery energy level. (c) Impart of ι on the computation rate.

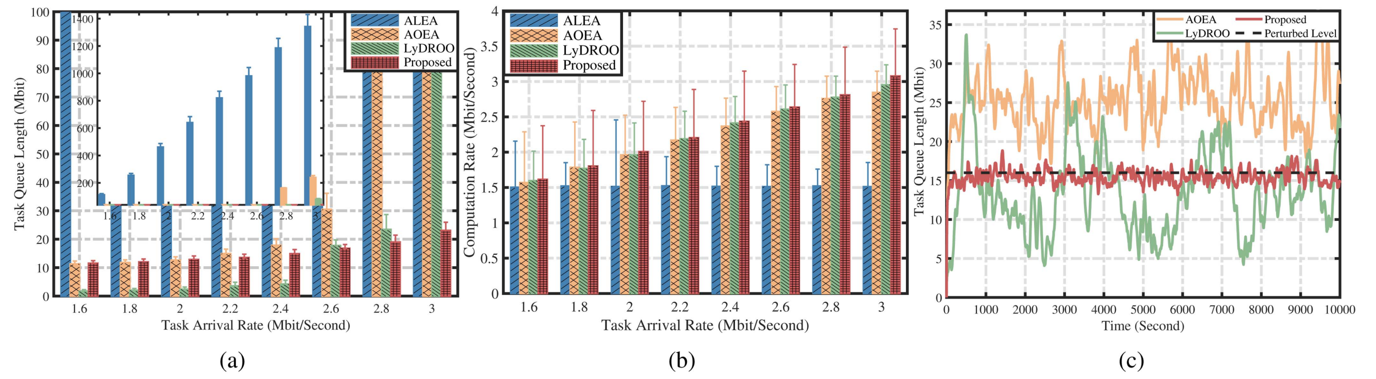

Fig. 7. Impact on system performance with different offloading algorithms (a) Impart of different offloading schemes on task queue backlog; (b) Impart of different offloading algorithms on computation rate; (c) Queue evolution across 10000 time slots with different algorithms.

In time slot s, all tasks are offloaded to the MEC server, and the task offloading strategy is solved using Alg (2).

- *Lyapunov-guided DRL for Online Computation Offloading* (LyDROO) [40]
  In time slot s, the arriving tasks follow a binary offloading method. By leveraging Lyapunov optimization techniques, the task offloading strategy and resource allocation scheme are determined in real-time, aiming to maximize the offloading rate while ensuring task queue stability. This algorithm is similar to the proposed one, but our algorithm emphasizes disturbance threshold constraints and employs a federated offloading approach.

We selected multiple sets of task arrival rates and calculated the average task queue backlog and computation rate for four algorithms under each task arrival rate. The results are presented in Fig. 7(a) and (b) for comparative analysis. It can be observed that the ALEA algorithm can't control the task queue backlog. This is because ALEA relies solely on the local processing mode. Compared to remote offloading, local processing consumes more energy and can only handle a limited number of tasks per time slot. Moreover, the energy harvested by the device in each time slot is inadequate to sustain the local CPU at full capacity. This limitation leads to an inability to process tasks efficiently, causing the task queue backlog to become uncontrollable. As shown in Fig. 7(b), the task computation rate of the ALEA algorithm reaches a maximum of 1.5 Mbit/s.

For the AOEA algorithm, when the system operates under a low task arrival rate (e.g., 1.6 or 1.8 Mbit/s), its average task queue backlog remains at the same level as that of the proposed algorithm. However, as the task arrival rate increases, the AOEA algorithm's task queue backlog rapidly increases and significantly exceeds that of the proposed algorithm. This is because the remote offloading mode reaches its maximum computation capacity when the task arrival rate hits 2.4 Mbit/s. Once the task arrival rate surpasses this threshold, the AOEA algorithm struggles to handle the arrived tasks promptly, leading to an unmanageable queue backlog and demonstrating weak robustness. Similarly, the LyDROO algorithm exhibits near-zero task queue backlog at low task arrival rates, but the backlog increases significantly as the task arrival rate grows. This is because the LyDROO algorithm does not account for the disturbance level of the task queue and only controls its stability. In contrast, the proposed algorithm shows better performance in controlling the task queue backlog. When the task arrival rate is within the system's processing capacity, the proposed algorithm can effectively control the task queue backlog near the disturbance level. When the task arrival rate increases and approaches 3.0 Mbit/s, the proposed algorithm gradually surpasses the disturbance level constraint. This is because of the constraints of the device's channel conditions and energy harvesting, which impose an upper limit on the system's computation capacity. As the number of arrived tasks nears this limit, the system becomes unable to offload the queued tasks promptly. Consequently, the task queue backlog ultimately exceeds the disturbance level. Unlike the comparative algorithms, the proposed algorithm can effectively control the slow rise of queue backlogs under such conditions, rather than losing control immediately.

To visually demonstrate the differences in queue stability control between the proposed algorithm and the comparative algorithms, we plot the variation trend of the task queue state over 10000 time slots, as shown in Fig. 7(c). Notice that the task queue backlog controlled by the ALEA algorithm has diverged and is not presented. From Fig. 7(c), we can see that the task queue variation trends of the AOEA and LyDROO algorithms exhibit considerable fluctuations. While both algorithms stabilize within a specific range, there are significant differences in task queue backlog between consecutive time slots. According to Little's Law [16], the average queuing delay is proportional to the queue backlog. When the queue backlog experiences large fluctuations, the task response time becomes unstable. In contrast, the proposed algorithm effectively stabilizes the task queue backlog near the disturbance level over a longer period, ensuring that the device maintains stable response times. In practical applications, a stable response time for the device reflects the reliability of the system, facilitating rational resource allocation and enhancing overall system efficiency.

VIII. Conclusion

In this paper, the task offloading problem in resource-constrained MEC systems is studied. It aims to optimize long-term system throughput by constructing a dynamic task offloading model, where task backlog and energy threshold constraints are transformed into virtual queue constraints using the perturbed Lyapunov optimization method. By minimizing the upper bound of drift-plus-penalty, the long-term optimization problem dependent on future time slots is simplified into a non-convex problem relying only on the current time slot. Based on this, we propose the LyFOTO algorithm to derive the optimal offloading strategy. When communication conditions are favorable, the LyFOTO algorithm adapts to enhance system throughput, while under poor conditions, it strategically delays task offloading to maintain queue backlog constraints. Comparative experimental results demonstrate that LyFOTO outperforms other benchmark algorithms by ensuring device task backlog and energy threshold constraints, achieving maximum system throughput.

Although the proposed method optimizes average performance metrics, such as queue length and throughput, it does not enforce strict hard constraints. As a result, instantaneous parameters, such as queue length or energy levels, may temporarily exceed their thresholds as long as the averages remain controlled. While this flexibility is generally suitable for dynamic environments, it may be inadequate in systems that require strict guarantees, particularly in safety-critical applications with hard energy limitations or those that cannot tolerate task loss due to buffer size constraints. Furthermore, since local models are aggregated through averaging, adversarial actors can exploit this mechanism by manipulating their local models, thereby adversely affecting the global model. This could lead to degraded overall model performance or biased outcomes, compromising the integrity and security of the system.

To address the aforementioned problem, future research will introduce more robust aggregation algorithms, such as weighted averaging, robust aggregation methods, or differential privacy techniques, to mitigate the impact of malicious participants on the global model. Additionally, model validation mechanisms and anomaly detection algorithms in federated learning will be employed to identify and eliminate faulty data, enhancing the system's security and reliability. Moreover, future work will explore hybrid optimization frameworks that combine average and hard constraints. Promising directions include the integration of predictive algorithms or adaptive threshold mechanisms to minimize the likelihood of threshold violations. Furthermore, we aim to extend the optimization objectives within MEC systems and explore federated learning optimization in large-scale intelligent transportation scenarios, specifically improving task queuing model accuracy by incorporating user mobility features.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] X. Ye and L. Fu, "Joint MCS adaptation and RB allocation in cellular networks based on deep reinforcement learning with stable matching," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 549–565, Jan. 2024.

[2] Q. Zhang et al., "Eye of Sauron: Long-range hidden spy camera detection and positioning with inbuilt memory EM radiation," in *Proc. 33rd USENIX Secur. Symp.*, 2024, pp. 109–126.

[3] S. Sinha, "State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion globally," 2024. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices

[4] Z. Xiao et al., "Multi-objective parallel task offloading and content caching in D2D-aided MEC networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6599–6615, Nov. 2023.

[5] Q. Zhang et al., "CamShield: Tracing electromagnetics to steer ultrasound against illegal cameras," *IEEE Internet Things J.*, vol. 11, no. 20, pp. 33296–33311, Oct. 2024.

[6] Z. Xiao, J. Shu, H. Jiang, G. Min, H. Chen, and Z. Han, "Perception task offloading with collaborative computation for autonomous driving," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 457–473, Feb. 2023.

[7] H. Zhao, S. Deng, C. Zhang, W. Du, Q. He, and J. Yin, "A mobility-aware cross-edge computation offloading framework for partitionable applications," in *Proc. 2019 IEEE Int. Conf. Web Serv.*, 2019, pp. 193–200.

[8] H. Jiang, J. Cai, Z. Xiao, K. Yang, H. Chen, and J. Liu, "Vehicle-assisted service caching for task offloading in vehicular edge computing," *IEEE Trans. Mobile Comput.*, vol. 24, no. 7, pp. 6688–6700, Jul. 2025.

[9] H. Tang, H. Wu, Y. Zhao, and R. Li, "Joint computation offloading and resource allocation under task-overflowed situations in mobile-edge computing," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 1539–1553, Jun. 2022.

[10] N. N. Ei, M. Alsenwi, Y. K. Tun, Z. Han, and C. S. Hong, "Energy-efficient resource allocation in multi-UAV-assisted two-stage edge computing for beyond 5G networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16421–16432, Sep. 2022.

[11] X. Dai et al., "Offloading dependent tasks in edge computing with unknown system-side information," *IEEE Trans. Serv. Comput.*, vol. 16, no. 6, pp. 4345–4359, Nov./Dec. 2023.

[12] Q. Li, S. Wang, A. Zhou, X. Ma, F. Yang, and A. X. Liu, “QoS driven task offloading with statistical guarantee in mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 278–290, Jan. 2022.

[13] X. Dai, Z. Xiao, H. Jiang, and J. C. Lui, “UAV-assisted task offloading in vehicular edge computing networks,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 2520–2534, Apr. 2024.

[14] Y. Li, S. Xia, M. Zheng, B. Cao, and Q. Liu, “Lyapunov optimization-based trade-off policy for mobile cloud offloading in heterogeneous wireless networks,” *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 491–505, First Quarter, 2022.

[15] X. Zhang et al., “Energy-efficient computation peer offloading in satellite edge computing networks,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3077–3091, Apr. 2024.

[16] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, “Learning-aided computation offloading for trusted collaborative mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2833–2849, Dec. 2020.

[17] S. Liu, P. Cheng, Z. Chen, W. Xiang, B. Vucetic, and Y. Li, “Contextual user-centric task offloading for mobile edge computing in ultra-dense network,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5092–5108, Sep. 2023.

[18] N. Cheng et al., “Space/Aerial-assisted computing offloading for IoT applications: A learning-based approach,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[19] Q. Tang et al., “Stochastic computation offloading for LEO satellite edge computing networks: A learning-based approach,” *IEEE Internet Things J.*, vol. 11, no. 4, pp. 5638–5652, Feb. 2024.

[20] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, “Deep reinforcement learning for stochastic computation offloading in digital twin networks,” *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4968–4977, Jul. 2021.

[21] H. Zhang, R. Wang, W. Sun, and H. Zhao, “Mobility management for blockchain-based ultra-dense edge computing: A deep reinforcement learning approach,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7346–7359, Nov. 2021.

[22] Q. An, Y. Zhou, Z. Wang, H. Shan, Y. Shi, and M. Bennis, “Online optimization for over-the-air federated learning with energy harvesting,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 7, pp. 7291–7306, Jul. 2024.

[23] C. Wang and H. Wu, “Energy optimization for federated learning on consumer mobile devices with asynchronous SGD and application co-execution,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 11, pp. 10235–10250, Nov. 2024.

[24] H. Jiang, Z. Xiao, Z. Li, J. Xu, F. Zeng, and D. Wang, “An energy-efficient framework for Internet of Things underlaying heterogeneous small cell networks,” *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 31–43, Jan. 2022.

[25] N. Abbas, W. Fawaz, S. Sharafeddine, A. Mourad, and C. Abou-Rjeily, “SVM-based task admission control and computation offloading using Lyapunov optimization in heterogeneous MEC network,” *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 3121–3135, Sep., 2022.

[26] J. Du, C. Jiang, J. Wang, Y. Ren, and M. Debbah, “Machine learning for 6G wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service,” *IEEE Veh. Technol. Mag.*, vol. 15, no. 4, pp. 122–134, Dec. 2020.

[27] J. Mei, L. Dai, Z. Tong, X. Deng, and K. Li, “Throughput-aware dynamic task offloading under resource constant for MEC with energy harvesting devices,” *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 3460–3473, Sep. 2023.

[28] H. Jiang, X. Dai, Z. Xiao, and A. Iyengar, “Joint task offloading and resource allocation for energy-constrained mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4000–4015, Jul. 2023.

[29] S. Xia, Z. Yao, G. Wu, and Y. Li, “Distributed offloading for cooperative intelligent transportation under heterogeneous networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16701–16714, Sep. 2022.

[30] Y. Ye, L. Shi, X. Chu, R. Q. Hu, and G. Lu, “Resource allocation in backscatter-assisted wireless powered MEC networks with limited MEC computation capacity,” *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 10678–10694, Dec. 2022.

[31] H. Wu, J. Chen, T. N. Nguyen, and H. Tang, “Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems,” *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 2117–2128, Feb. 2023.

[32] Z. Xiao, J. Shu, H. Jiang, G. Min, J. Liang, and A. Iyengar, “Toward collaborative occlusion-free perception in connected autonomous vehicles,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4918–4929, May 2024.

[33] S. Xia, Z. Yao, Y. Li, and S. Mao, “Online distributed offloading and computing resource management with energy harvesting for heterogeneous MEC-enabled IoT,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6743–6757, Oct. 2021.

[34] M.-L. Ku, W. Li, Y. Chen, and K. R. Liu, “Advances in energy harvesting communications: Past, present, and future challenges,” *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1384–1412, Second Quarter, 2016.

[35] M. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.

[36] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. Berlin, Germany: Springer, 2012.

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

[38] S. Marsland, *Machine Learning: An Algorithmic Perspective*. London, U.K./Boca Raton, FL, USA: Chapman and Hall/CRC, 2011.

[39] P. R. Murthy, *Operations Research (Linear Programming)*. Münster, Germany: Bohem Press, 2005.

[40] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, “Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021.

[41] Q. Zhang et al., “Enhancing perception for intelligent vehicles via electromagnetic leakage,” *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 7, pp. 7029–7043, Jul. 2024.

[42] Q. Zhang et al., “E-Argus: Drones detection by side-channel signatures via electromagnetic radiation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 11, pp. 18978–18991, Nov. 2024.

Longbao Dai received the BS degree in computer science and technology from the Hunan University of Science and Engineering, Yongzhou, China, in 2021, and the MS degree in computer science and technology from Hunan Normal University, Changsha, China, in 2024. He is currently working toward the PhD degree in computer science and technology with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests focus on distributed parallel computing and resource allocation in mobile edge computing systems.

Fanzi Zeng (Member, IEEE) received the PhD degree in signal and information processing from Beijing Jiaotong University, Beijing, China, in 2005. Since 2005, he has been with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, where he is currently a professor. In 2009, he was with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, as a visiting scholar. His current research focuses on cognitive radio technology and artificial intelligence.

Haoran Kong received the BS degree in electronic information engineering from Chongqing Technology and Business University, Chongqing, China, in 2021, and the MS degree in electronic information from the Xi’an University of Science and Technology, Xi’an, China, in 2024. He is currently working toward the PhD degree in electronic information with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests focus on mobile edge computing and deep learning.

Jianghao Cai received the MS degree from Tianjin Chengjian University, Tianjin, China, in 2023, where he is currently working toward the PhD degree with the College of Computer Science and Electronic Engineer, Hunan University. His research interests lie in the area of Internet of Vehicles and mobile communications.

Hongbo Jiang (Senior Member, IEEE) received the PhD degree from Case Western Reserve University, in 2008. He is now a full professor with the College of Computer Science and Electronic Engineering, Hunan University. He was a professor with the Huazhong University of Science and Technology. His research concerns computer networking, especially algorithms and protocols for wireless and mobile networks. He is serving as the editor of the *IEEE/ACM Transactions on Networking*, the associate editor of the *IEEE Transactions on Mobile Computing*, and the associate technical editor of the *IEEE Communications Magazine*.

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 a SUNY distinguished professor with the State University of New York and a National distinguished professor with Hunan University (China). He has authored or coauthored more than 1130 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by the Chinese National Intellectual Property Administration. Since 2020, he has been among the world's top few most influential scientists in parallel and distributed computing regarding single-year impact (ranked #2) and career-long impact (ranked #4) based on a composite indicator of the *Scopus* citation database. He is listed in Scilit Top Cited Scholars (2023-2024) and is among the top 0.02% out of more than 20 million scholars worldwide based on top-cited publications. He is listed in ScholarGPS Highly Ranked Scholars (2022-2024) and is among the top 0.002% out of more than 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He received the *IEEE TCCLD Research Impact Award* from the *IEEE CS Technical Committee on Cloud Computing* in 2022 and the *IEEE TCSVC Research Innovation Award* from the *IEEE CS Technical Community on Services Computing* in 2023. He won the *IEEE Region 1 Technological Innovation Award (Academic)* in 2023. He was a recipient of the 2022-2023 *International Science and Technology Cooperation Award* and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a member of the SUNY Distinguished Academy. He is an AAAS fellow, an AAIA fellow, an ACIS fellow, and an AIIA fellow. He is a member of the European Academy of Sciences and Arts. He is a Member of Academia Europaea (Academician of the Academy of Europe).