Throughput-Aware Dynamic Task Offloading Under Resource Constant for MEC With Energy Harvesting Devices

Jing Mei, Longbao Dai, Zhao Tong, *Member, IEEE*, Xin Deng, and Keqin Li, *Fellow, IEEE*

***Abstract*—With the explosive increase of Internet of Things (IoT) devices, an increasing number of computation-intensive applications are emerging in IoT system. However, most IoT devices are limited by size and location, equipped with low-performance CPUs and low-capacity batteries, which cannot go well with computation-intensive applications. Mobile edge computing (MEC) is considered as a promising solution to provide computation-intensive and latency-sensitive services in IoT system, but it is still challenging to improve the throughput and extend the battery life of IoT devices under communication constraints. This paper focuses on the task offloading problem for an MEC system with multiple energy harvesting (EH) devices. To accommodate the system dynamics and ensure the system stability in terms of task queue and battery level, we apply Lyapunov optimization theory, and design a computation tasks maximum offloading algorithm to maximize the system throughput. The algorithm can determine the offloading decision in real-time without knowing any statistical information about the system. We first give a series of mathematical analysis to verify the system stability and discuss the performance of the algorithm. In addition, a number of simulation experiments are conducted to present the efficiency of the algorithm.**

***Index Terms*—Energy harvesting, Lyapunov optimization, mobile edge computing, task offloading.**

I. Introduction

With the explosive increase of Internet of Things (IoT) devices, e.g., mobile phones, laptops, smart terminals, and various sensors, it becomes a general trend to migrate computation-intensive applications to IoT devices [1], [2]. Processing such applications always requires powerful computing capacity. However, constrained by size and location, most IoT devices are equipped with low-performance CPU and low-capacity battery, which limits the computing capacity of IoT devices severely. Consequently, they cannot deal with computation-intensive applications effectively. A feasible method to address this problem is to offload such computation-intensive applications to other servers for execution. Mobile edge computing (MEC) provides such a technology that deploys offloading nodes at the margin of the network [3]. Via task offloading, both the computing capacity and the battery lifetime of IoT devices can be enhanced effectively.

Manuscript received 18 July 2022; revised 22 December 2022; accepted 31 January 2023. Date of publication 9 February 2023; date of current version 9 October 2023. This work was supported by the Program of National Natural Science Foundation of China (grant No. 62072174, 61502165), National Natural Science Foundation of Hunan Province, China (grant No. 2022JJ40278, 2020JJ5370), Scientific Research Fund of Hunan Provincial Education Department, China (Grant No. 22A0026). The associate editor coordinating the review of this article and approving it for publication was M. Reisslein. *(Corresponding author: Zhao Tong.)*

Jing Mei, Longbao Dai, Zhao Tong, and Xin Deng are with the College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China (e-mail: jingmei1988@163.com; awaken6758@gmail.com; tongzhao@hunnu.edu.cn; 202120293782@hunnu.edu.cn).

Keqin Li is with the College of Information Science and Engineering, Hunan University, Changsha 410012, China, also with National Supercomputing Center, Changsha 410082, Hunan, China, and also with the Department of Computer Science, State University of New York, New York, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TNSM.2023.3243629

Although task offloading can effectively extend the battery life and improve the computing capacity of IoT devices, it still faces several obstacles. First, for the tasks created by the devices, local execution and remote offloading to MEC (partially or fully) are two common ways to handle these tasks [4]. The energy consumption of IoT devices to process tasks locally or offload tasks remotely is different and it is affected by different factors. Energy consumption from local processing correlates with CPU frequency, while energy consumption of offloading tasks to MEC is extremely dependent on the channel state. To make task offloading optimal, we need to consider how to properly assign the tasks of each device under a stochastic channel state. Second, there is always competition for the limited computation and communication resources when a great deal of devices connect to the MEC server and begin to offload tasks at once [5]. Thus, it is a challenge to properly allocate the limited resources for each device such that the objective performance is maximized. Third, due to the system dynamics, e.g., channel state, task arrival rate, and energy harvesting efficiency, the system processing capability is changing over time. To achieve a long-term objective optimal while maintaining the system stability, the task offloading decision should be determined in real-time. This results in the problem being much more complicated.

In the majority of research, the task offloading decisions are based on the assumption that the task arrival rate, channel state, and other parameters were known in advance and remained unchanged, or could be precisely forecasted from the historical information. However, the task arrival rates of the different devices dynamically change over time. Estimating the task arrival rates necessitates a large amount of statistical work, and its status is difficult to anticipate precisely. Furthermore, the quality of the radio channel changes dynamically over time as well. It is impacted by a variety of parameters, such as device location and the degree of network congestion [6]. Exactly predicting the time-varying channel state is nearly impossible.

To address the aforementioned difficulties, this article utilizes Lyapunov optimization method and solves the dynamic

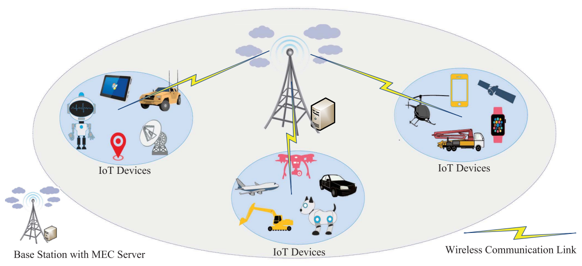


Fig. 1. System Architecture.

task offloading problem for the multi-IoT devices MEC system in a different scenario from the existing studies. In the scenario, each IoT device has an energy harvesting module. To keep the system running for a long time, the battery of each device keeps above a minimum charge. Thus, we set a value as the battery power threshold for each device. The system architecture as shown in Fig. 1. This study is similar to [7], but there are several differences as follows.

- The application scenarios are different. In [7], the IoT devices are not equipped with energy harvesting modules and the tasks are not assigned to execute locally. Thus, the devices cannot utilize renewable energy and only handle the tasks by remote offloading. In our work, IoT devices with energy harvesting modules and computing abilities are considered. Therefore, IoT devices can utilize renewable energy to provide a portion of the power. The tasks not only can be processed locally but also can be offloaded to MEC.
- The optimization objectives are different. Reference [7] is to minimize the IoT device's transmission energy while maintaining the system stability. In our work, the goal is to maximize the system throughput while keeping the battery level of each device above the threshold and maintaining the maintain stability of the task queue of each device.

To accommodate the system dynamics and maintain the system stability, we apply Lyapunov optimization theory to deal with this problem. The major contributions are summarized as follows:

- We consider the task offloading problem for a multi-IoT devices single-MEC scenario. In this scenario, each device is equipped with an energy harvesting module, which allows the user to use externally captured energy to process tasks. They are heterogeneous in terms of computing capacity, power consumption, and computation requirements. In our problem, the tasks, the network state, and the energy harvested in each time slot are dynamically changing and stochastic.
- We define the task offloading problem as a dynamic optimization problem that is conducted to maximize the system throughput while maintaining the system stability of the task queue and battery level in long-term scales. Meanwhile, setting a battery level threshold for each device can effectively extend the life of the device battery. Applying Lyapunov optimization theory, the original optimization problem which depends on further unknown information is transferred into a new problem that only depends on current system information. To address the this problem, we exploit a Computation Tasks Maximum Offloading Algorithm (CTMOA).
- We present a rigorous mathematical analysis on the performance of CTMOA, proving the task queue exists an upper bound. Specifically, we demonstrate that the battery constraint is fulfilled by the power management scheme generated by CTMOA. We also conduct many experiments to certify the effectiveness of CTMOA, observing the number of tasks processed by the CTMOA algorithm and the changing trend of the task queue and battery level under different parameters.

The rest of this article is organized as follows. In Section II, we present the related work in MEC. In Section III, we describe the system model in detail, and formulate the optimization problem rigorously. In Section IV, we apply Lyapunov optimization, and design a computation tasks maximum offloading algorithm called CTMOA to tackle with this problem. In Section V, we discuss the CTMOA algorithm performance by mathematical analysis. In Section VI, we estimate the CTMOA algorithm performance by several groups of experiments. In Section VII, we conclude this paper.

II. Related Work

There are extensive studies on the task offloading issue in MEC. In the single-device single-MEC application scenario, Liu et al. [8] investigated the two-timescale stochastic optimization problem. That is, in long time scales, the tasks are executed locally or offloaded to MEC via wireless channels, and in short time scales, the allocation of floading strategy is determined by the channel state. By quoting the Markov decision process, they proposed the dynamic task offloading strategy to address this problem. Unlike the application scenarios in [8], Dinh et al. [9] deployed a single mobile device (MD) which connected a group of edge servers scenario. For both fixed and flexible CPU frequencies, a series of optimization approaches were designed to improve the capability of MD and reduce the power consumption of MD. Different from the commonly studied resource management strategies [10], [11], Bi and Zhang [12] studied a multi-devices single MEC network model. All the devices are equipped with energy harvesting modules based on wireless power transfer (WPT). They goal is to maximize the computation rate of each device. To achieve this goal, a binary offloading strategy was proposed. Recently, Artificial Intelligence/Machine Learning (AI/ML) has made great progress [13]. Numerous algorithms [14], [16], [17], [18] have been developed. Du et al. [13] introduced a lot of state-of-the-art techniques based on AI/ML to support ultra-reliable and low-latency communications (URLLC) in future networks and illustrated intelligent terahertz techniques, such as AI/ML enabled terahertz channel estimation and spectrum management, which are considered revolutionary, to achieve an ultra-broadband transmission. Zhang et al. [14] used AI/ML technology to minimize latency costs and energy consumption. Zhou et al. [19] investigated the task offloading problem for a single MEC system that incorporates multiple antennas. Avoiding the device battery energy from falling below a predetermined threshold, and minimizing energy consumption and execution latency are their objectives. They convert the objective issue into a nonconvex problem quoting the Lyapunov optimization method, and then develop a dynamic offloading strategy to solve it. In an approximate cloud computing environment, Lyu et al. [20] took into account the computation offloading problem. A heuristic algorithm was proposed to speed up the task completion time and cut down on the energy consumption of MD. The rigorous mathematical analysis demonstrated that the optimization issue is NP-hard. Dai et al. [21] designed a particle swarm optimization algorithm to tackle the online task offloading problem on the Internet of Vehicles (IoV). The goal is to heighten resource utilization. According to simulation experiment results, the algorithm improves the global resource utilization from 71.8 to 94.5 percent. Overall, nearly all the studies listed above suppose that the channel states or task arrival rates are known or can be accurately predicted according to system information. The task arrival rates in MEC are stochastic, and channel conditions are affected by a series of parameters. Therefore, it is difficult to achieve accurate predictions.

To address this challenge, many studies have been done in recent years to avoid the prediction of dynamic parameters by applying stochastic optimization methods. Among the stochastic optimization methods, the Lyapunov optimization method is the most universal one [7], [18], [19], [22], [23], [24], [25], [26], [27], [28]. Applying the Lyapunov optimization method, Chen et al. [7] used backpack theory to design an online offloading algorithm to address dynamic task offloading problem in MEC system. Minimizing energy consumption is their goal, but they only considered remote offloading, not local processing. Deep reinforcement learning (DRL) was used by [15], [16], [17], [18] to address the task offloading problem in MEC. Zhang et al. [15] presented two DRL based algorithms, i.e., hybrid decision-based actor-critic learning and multi-device hybrid decision-based actor-critic learning, to address the two challenges of MEC system with EH devices: continuous-discrete hybrid action spaces, and coordination among devices. Using a Markov decision process (MDP), Shi et al. [16] constructed the task offloading issue with the goal of maximizing the mean latency-aware utility of tasks in a period. Bi et al. [18] dedicated to improving the task processing capability in MEC network, and proposed an online strategy to maximize task offloading rate based on DRL. They formulated the maximizing the task offloading rates into a non-linear programming problem. Quoting the Lyapunov optimization method and DRL, a new online resource allocation algorithm was proposed to address this problem. Although this study discussed local processing, it did not consider energy harvesting. Mao et al. [22] exploited an online resource allocation approach in a multi-devices MEC system, using Gauss-Seidel theory to analyze the best transmit power, but this study also did not consider energy harvesting. Xia et al. [23] studied an EH-enabled MEC offloading system. According to perturbed Lyapunov methods and buyer/seller game theory, an online distributed algorithm was designed to optimize computing resources and battery energy and improve task offloading efficiency. Du et al. [29] designed an evolutionary game based service selection model for users, and proposes a Stackelberg differential game based cloud computing resource sharing mechanism to facilitate resource transactions among resource providers. However, they did not consider the battery energy threshold constraints of the devices, which do not effectively extend the battery life. Zhao et al. [24] investigated the dynamic offloading problem for a three-layer MEC network model. Compared to the EEDOA algorithm in [7], Zhao et al. further designed a dynamic offloading algorithm to optimize resource allocation and reduce power consumption. In this paper, the EH module is deployed in a base station (BS), which allows for the most efficient use of external energy. However, they also did not consider the battery energy threshold constraints.

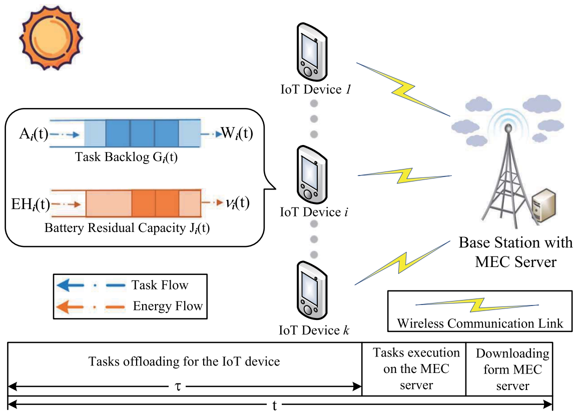

IoT Device 1
$A_i(t)$
$W_i(t)$
Task Backlog $G_i(t)$
$EH_i(t)$
$\nu_i(t)$
Battery Residual Capacity $J_i(t)$
IoT Device i
Base Station with MEC Server
Task Flow
Energy Flow
IoT Device k
Wireless Communication Link
Tasks offloading for the IoT device
τ
Tasks execution on the MEC server
Downloading form MEC server
t

Fig. 2. The System Model.

In most of the studies mentioned above, the devices with energy harvesting modules were not considered [6], [7], [8], [9], [18], [20], [21], [30], [31]. In [12], [16], [23], [24], [27], [29], the battery level constraints of the devices are not guaranteed, which makes the battery life significantly reduced. To address this situation, we propose an optimization framework for task offloading maximization problems with energy harvesting modules, while ensuring the system stability in terms of task queue and battery level.

III. System Model and Problem Formulation

A. System Model

One generic MEC system with one BS is explored in this study. Task offloading has two types offline offloading and online offloading. This article belongs to online task offloading. In general, the future information of the system cannot be predicted precisely. Within the coverage of the BS, there are k devices generating tasks dynamically. Local execution and remote offloading to MEC (partially or fully) are two common ways to handle these tasks. Let $K = \{1, 2, \ldots, k\}$ represent the index set of the k IoT devices inside the BS's coverage. As shown in Fig. 2, each IoT device owns a rechargeable battery of limited capacity and is equipped with an EH circuit component which can harvest external energy to power its operation [32]. Due to the dynamic nature of the system, the amount of harvested energy per unit time is stochastic. Thus, we divide time into a series of time slots $\mathcal{T} = \{0, 1, 2, \ldots\}$ and each time slot has the same length τ.

B. Task and Queuing Model

In per time slot τ, each device generates new computation tasks and the number of tasks is denoted by $A_i(t)$ (in bits). Each device maintains a queue buffer to store the generated tasks which wait to be processed. The first-come-first-served (FCFS) discipline is applied to the tasks to be processed in the queue. The total amount of tasks processed in slot *t* is denoted by $W_i(t)$. For device *i*, $G_i(t)$ denotes the queue length in current time slot *t*, and then the queue length in next slot is

$$G_i(t+1) = \max\{G_i(t) - W_i(t) + A_i(t), 0\}, \quad (1)$$

where $G_i(0) = 0$. It is certain that each device cannot process more tasks than what it has, so

$$W_i(t) \le G_i(t) + A_i(t) \quad (2)$$

holds always.

According to the definition of stability in [33], all the tasks satisfy

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\{G_i(t)\} \le \infty. \quad (3)$$

The task queue is stable.

C. Computation and Energy Model

Local computation: For device *i*, c_i denotes the required CPU cycles to process one bit of data, $f_{i,l}$ denotes the CPU cycle frequency, and $W_{i,l}(t)$ denotes the amount of tasks executed locally. Then, the time required to process $W_{i,l}(t)$ bits tasks in device *i* is $T_{i,l}(t) = c_i W_{i,l}(t)/f_{i,l}$. Since the tasks should be processed within current time slot, that is, $T_{i,l}(t) \le \tau$, we have

$$W_{i,l}(t) \le \frac{\tau f_{i,l}}{c_i}. \quad (4)$$

Let $E_{i,l}(t)$ denote the energy consumption for local execution of device *i* and then $E_{i,l}(t)$ is calculated as

$$E_{i,l}(t) = \frac{c_i P_{i,l} W_{i,l}(t)}{f_{i,l}}, \quad (5)$$

where $P_{i,l} = \xi_i (f_{i,l})^3$ is the computing power consumption (in watts). ξ_i represents the effective switched capacitance of the CPU, which is determined by the chip structure of device *i* [24], [34].

Task Transmission: In slot *t*, the transmit power and channel power gain of each IoT device *i* are indicated by $P_{i,o}$ and $h_i(t)$, respectively. Since the channel state is stochastic, channel power gain is dynamically changing over time. Therefore, we denote the achievable tasks communication speed as $R_i(t)$ (in bits/s), which is,

$$R_i(t) = B \log_2\left(1 + \frac{P_{i,o} h_i(t)}{B N_0}\right),$$

where *B* is the channel's bandwidth and N_0 is the noise power spectral density [7].

Let *S*(*t*) denote the number of available up-link channels. Based on realistic scenarios, *S*(*t*) is considered to dynamically vary over different time periods in this study. Similar to [35], the Time Division Multiple Access (TDMA) channel model is used in this study, which means that various devices can access a channel in TDMA at different times during a time slot. To improve the system performance, channel resources should be appropriately allocated to different devices. Then, the channel allocation decision is defined as $\boldsymbol{\pi}(t) = \{\pi_1(t), \ldots, \pi_k(t)\}$, where $\pi_i(t)$ represents the duration allocated to device *i*. The offloading tasks amount during slot *t* is denoted by

$$W_{i,o}(t) = R_i(t)\pi_i(t). \quad (6)$$

It is obvious that a device cannot offload more tasks in a time slot than that it has. Hence, $\pi_i(t)$ should satisfy

$$\pi_i(t) \le \frac{G_i(t) + A_i(t)}{R_i(t)}. \quad (7)$$

Based on the same assumption with [35], each device can only access a channel in TDMA at a time slot. So we have

$$0 \le \pi_i(t) \le \tau, \forall i \in K. \quad (8)$$

According to (7) and (8), we have (9).

$$0 \le \pi_i(t) \le \min\left\{\frac{G_i(t) + A_i(t)}{R_i(t)}, \tau\right\}. \quad (9)$$

Besides, the communication duration of all channels is not shorter than the total length of the channel resources allocated to all devices, which is,

$$\sum_{i=0}^{k} \pi_i(t) \le S(t)\tau. \quad (10)$$

Let $E_{i,o}(t)$ denotes energy consumed by device *i* for task transmission, which can be calculated by

$$E_{i,o}(t) = P_{i,o} \frac{W_{i,o}(t)}{R_i(t)} = P_{i,o}\pi_i(t), \quad (11)$$

where $P_{i,o}$ is transmission power of device *i*.

According to the above definitions, the energy consumption of device *i* is defined as

$$\nu_i(t) = \frac{c_i P_{i,l} W_{i,l}(t)}{f_{i,l}} + \frac{P_{i,o} W_{i,o}(t)}{R_i(t)} + P_c\tau, \quad (12)$$

where P_c denotes the constant circuit power consumption [36], which is the lowest power consumption to maintain the basic operation of the IoT device.

The total amount of tasks that device *i* can process in slot *t* is

$$W_i(t) = W_{i,o}(t) + W_{i,l}(t).$$

D. Battery and EH Model

In this study, we presume that the energy used by IoT devices for offloading tasks only comes from the battery and the harvested energy in the current time slot can only be used in the next time slot. This model is adopted widely in existing studies [23], [25], [27]. $J_i(t)$ denotes the battery energy of device *i*. Consequently, $J_i(t+1)$ is defined as

$$J_i(t+1) = \min\{J_i(t) - \nu_i(t) + EH_i(t), J_{max}\}, \quad (13)$$

where $EH_i(t)$ is the energy harvested in time slot t, J_{max} is the battery capacity, and $\nu_i(t)$ satisfies

$$0 \leq \nu_i(t) \leq J_i(t), \tag{14}$$

which indicates that device i cannot consume more energy than $J_i(t)$ in time slot t. To keep the system running for a long time, the battery power should be kept above a minimum level. Hence, we set a threshold for the battery power of each device, which is,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\{J_i(t)\} \geq \sigma, \tag{15}$$

where σ is a predefined energy level threshold.

E. Problem Formulation

In this study, we are conducted to maximize the system throughput, maintain of the task queue of each device, and keep each device battery level above the threshold at the same time. This can be realized by jointly optimizing the task offloading decisions and resource allocation schemes.

The system throughput is measured by the amount of tasks processed by all devices in unit time. Due to the dynamics of system, we define the system throughput as the time-average amount of processed tasks of all devices, which is

$$\max_{\boldsymbol{\pi}(t), W_{i,l}(t)} \overline{W} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\{W(t)\},$$
$$\text{s.t. } (2), (3), (9), (10), (14) \text{ and } (15), \tag{16}$$

where

$$W(t) = \sum_{i=1}^{k} \{ W_{i,l}(t) + W_{i,o}(t) \}.$$

According to (12), $W_{i,l}(t)$ can be rewriten as

$$W_{i,l}(t) = \frac{\big(\nu_i(t) - P_{i,o}\pi_i(t) - P_c\tau\big) f_{i,l}}{c_i P_{i,l}}. \tag{17}$$

Combining (6) with (17), constraint (2) can be transformed into

$$\frac{\big(\nu_i(t) - P_{i,o}\pi_i(t) - P_c\tau\big) f_{i,l}}{c_i P_{i,l}} + R_i(t)\pi_i(t) \leq G_i(t) + A_i(t). \tag{18}$$

Letting $\zeta(t) = -W(t)$, (16) can be redefined as

$$\min_{\boldsymbol{\pi}(t), \nu_i(t)} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\{\zeta(t)\},$$
$$\text{s.t. } (3), (9), (10), (14), (15) \text{ and } (18), \tag{19}$$

where

$$\zeta(t) = \sum_{i=1}^{k} \left(\frac{\big(P_{i,o}\pi_i(t) - \nu_i(t)\big) f_{i,l}}{c_i P_{i,l}} - R_i(t)\pi_i(t) \right).$$

IV. Computation Tasks Maximum Offloading Algorithm

Since (19) depends on the information from time 1 to T, we first quote *Lyapunov's drift plus penalty* method [33] to transform (19) into a problem which only relies on the message about the current time. Then, a Computation Tasks Maximum Offloading Algorithm (CTMOA) is designed to solve it.

To transform our problem, a virtual queue M is constructed firstly for the battery energy level constraint (15), which is shown as

$$M_i(t+1) = \max\{M_i(t) + \sigma - J_i(t+1), 0\}, \tag{20}$$

where $M_i(t)$ is the virtual queue backlog in slot t. Noticed that M is different from J, since J is the battery energy of devices, while M is a virtual queue. The virtual queue backlog $M_i(t)$ reflects the degree that J_i is below the threshold σ by the end of time slot t, and $M_i(0) = 0$ for all devices. When the energy level of J is less than the threshold σ at t, the queue backlog of M increases, and vice verse. Hence, the energy level threshold constraint of J (15) is transformed into a stability constraint of the queue backlog of M. Until that, the Lyapunov optimization theory can be utilized to solve this problem.

A. Problem Transformation

To jointly control the task and energy queues, we define

$$\Theta(t) \triangleq \{G_1(t), \ldots, G_k(t), M_1(t), \ldots, M_k(t)\}$$

as the queue backlog vector. Thus, the *Lyapunov function* can be denoted as

$$L(\Theta(t)) = \frac{1}{2} \sum_{i=1}^{k} \Big(G_i^2(t) + \alpha M_i^2(t) \Big), \tag{21}$$

where α is the trade-off factor. According to [33], we can see that $L(\Theta(t))$ being "small" implies that both queue backlogs are "small" and $L(\Theta(t))$ being "large" implies that at least one queue backlog is "large". In this study, the average length of all the task queues is around 10^7, while the average length of all the virtual queues is around 10. Therefore, we set the trade-off factor α to 10^6.

In this study, $A_i(t)$ and $EH_i(t)$ can be deemed as two groups of random variables in time slot t. $G_i(t+1)$, $J_i(t+1)$ and $\zeta(t)$ are function values of random variables. Accordingly, the expectation operation is introduced when we use the Lyapunov optimization method. The *conditional Lyapunov drift* $\Delta(\Theta(t))$ is calculated as

$$\Delta(\Theta(t)) = \mathbf{E}\{L(\Theta(t+1)) - L(\Theta(t)) | \Theta(t)\}. \tag{22}$$

The *drift plus penalty* can be written as

$$\Delta(\Theta(t)) + V\mathbf{E}\{\zeta(t)|\Theta(t)\}, \tag{23}$$

where $V \geq 0$ is penalty weight, which reflects the trade-off between the queue backlog and the optimization objective. In this study, the goal is to maximize the system throughput. According to the Lyapunov optimization theory [33], (19) can be redefined as

$$\min_{\boldsymbol{\pi}(t),\boldsymbol{\nu}(t)} \quad \Delta(\Theta(t)) + V\mathbf{E}\{\zeta(t)|\Theta(t)\},$$
$$\text{s.t.} \quad (9), (10), (16) \text{ and } (19), \tag{24}$$

where $\boldsymbol{\nu}(t) = \{\nu_1(t), \ldots, \nu_k(t)\}$. We have Theorem 1 for an upper bound of (23).

Theorem 1: If $A_i(t)$, $W_i(t)$ and $EH_i(t)$ are respectively upper bounded by A_i^{max}, W_i^{max} and EH_i^{max} over time slots, the *drift-plus-penalty* value under any task offloading algorithm satisfies

$$\begin{aligned}\Delta(\Theta(t)) + V\mathbf{E}\{\zeta(t)\} &\le C_1 + C_2 + C_3 \\ &+ \alpha\sum_{i=1}^{k}\left(\frac{1}{2}\nu_i^2(t) + (M_i(t) + \sigma - J_i(t) - EH_i(t))\nu_i(t)\right) \\ &- \sum_{i=1}^{k} W_i(t)(G_i(t) + A_i(t)) + V\mathbf{E}\{\zeta(t)\},\end{aligned} \tag{25}$$

where

$$C_1 = \frac{1}{2}\sum_{i=1}^{k}\left((A_i^{max})^2 + (W_i^{max})^2 + 2G_i(t)A_i^{max}\right),$$

$$C_2 = \frac{\alpha}{2}\sum_{i=1}^{k}\left(\sigma^2 + J_{max} + (EH_i^{max})^2 + 2(J_{max})^2 EH_i^{max}\right),$$

and

$$C_3 = \alpha\sum_{i=1}^{k}(\sigma - J_i(t))M_i(t).$$

Proof: Taking square on (1) and (20) and exploiting $\left(\max\{x,0\}\right)^2 \le x^2$, we have

$$\begin{aligned}&G_i^2(t+1) - G_i^2(t) \\ &= (\max\{G_i(t) - W_i(t) + A_i(t), 0\})^2 - G_i^2(t) \\ &\le 2G_i(t)(A_i(t) - W_i(t)) + (A_i(t) - W_i(t))^2 \\ &\le A_i^2(t) + W_i^2(t) + 2G_i(t)A_i(t) - 2W_i(t)A_i(t) \\ &\quad - 2G_i(t)W_i(t) \\ &\le (A_i^{max})^2 + (W_i^{max})^2 + 2G_i(t)A_i^{max} \\ &\quad - 2W_i(t)(A_i(t) + G_i(t)),\end{aligned} \tag{26}$$

and

$$\begin{aligned}&M_i^2(t+1) - M_i^2(t) \\ &= (\max\{M_i(t) + \sigma - J_i(t+1), 0\})^2 - M_i^2(t) \\ &\le 2M_i(t)(\sigma - J_i(t+1)) + (\sigma - J_i(t+1))^2 \\ &\le 2M_i(t)\sigma + \sigma^2 + (\min\{J_i(t) - \nu_i(t) + EH_i(t), J_{max}\})^2 \\ &\quad - 2(M_i(t) + \sigma)\min\{J_i(t) - \nu_i(t) + EH_i(t), J_{max}\} \\ &\le 2M_i(t)\sigma + \sigma^2 - 2(M_i(t) + \sigma)(J_i(t) - \nu_i(t)) \\ &\quad + (J_i(t) - \nu_i(t) + EH_i(t))^2 \\ &\le 2M_i(t)\sigma + \sigma^2 - 2M_i(t)J_i(t) + 2M_i(t)\nu_i(t) \\ &\quad - 2\sigma J_i(t) + 2\sigma\nu_i(t) + (J_i(t) - \nu_i(t) + EH_i(t))^2 \\ &\le 2(M_i(t) + \sigma - J_i(t) - EH_i(t))\nu_i(t) + 2(\sigma - J_i(t))M_i(t) \\ &\quad + \nu_i^2(t) + \sigma^2 + (J_{max})^2 + \left(EH_i^{max}\right)^2 + 2J_{max}EH_i^{max}.\end{aligned} \tag{27}$$

Let

$$C_1 = \frac{1}{2}\sum_{i=1}^{k}((A_i^{max})^2 + (W_i^{max})^2 + 2G_i(t)A_i^{max}),$$

$$C_2 = \frac{\alpha}{2}\sum_{i=1}^{k}\left(\sigma^2 + (J_{max})^2 + \left(EH_i^{max}\right)^2 + 2J_{max}EH_i^{max}\right),$$

and

$$C_3 = \alpha\sum_{i=1}^{k}(\sigma - J_i(t))M_i(t).$$

Summing over all the IoT devices on (26) and (27), and taking conditional expectation, it has

$$\begin{aligned}&\Delta(\Theta(t)) + V\mathbf{E}\{\zeta(t)\} \le C_1 + C_2 + C_3 + V\mathbf{E}\{\zeta(t)\} \\ &\quad + \alpha\sum_{i=1}^{k}\left(\frac{1}{2}\nu_i^2(t) + (M_i(t) + \sigma - J_i(t) - EH_i(t))\nu_i(t)\right) \\ &\quad - \sum_{i=1}^{k}\left(\frac{(\nu_i(t) - P_{i,o}\pi_i(t))f_{i,l}}{c_iP_{i,l}} + R_i(t)\pi_i(t)\right)(A_i(t) + G_i(t)).\end{aligned}$$

The lemma is proven. ■

Following Theorem 1, we can see that C_1 and C_2 are constants during all time slots, and C_3 is also a constant in a specific time. Thus, in the objective function, C_1, C_2 and C_3 minimization operation is not required. The optimal problem is reformulated as

$$\begin{aligned}\mathbf{P:} \quad &\min_{\boldsymbol{\pi}(t),\boldsymbol{\nu}(t)} \sum_{i=1}^{k}\varphi_i(t)\pi_i(t) + \sum_{i=1}^{k}\left(\frac{\alpha}{2}\nu_i^2(t) + \psi_i(t)\nu_i(t)\right), \\ \text{s.t.} \quad &0 \le \pi_i(t) \le \min\left\{\frac{G_i(t) + A_i(t)}{R_i(t)}, \tau\right\}, \\ &\sum_{i=0}^{k}\pi_i(t) \le S(t)\tau, \\ &\frac{\left(\nu_i(t) - P_{i,o}\pi_i(t) - P_c\tau\right)f_{i,l}}{c_iP_{i,l}} + R_i(t)\pi_i(t) \\ &\le G_i(t) + A_i(t).\end{aligned} \tag{28}$$

Here,

$$\varphi_i(t) = (G_i(t) + A_i(t) + V)\left(\frac{P_{i,o}f_{i,l}}{c_iP_{i,l}} - R_i(t)\right)$$

and

$$\begin{aligned}\psi_i(t) = &\ \alpha(M_i(t) + \sigma - J_i(t) - EH_i(t)) \\ &- \frac{(G_i(t) + A_i(t) + V)f_{i,l}}{c_iP_{i,l}}.\end{aligned}$$

B. Computation Tasks Maximum Offloading Algorithm

Analyzing the above problem, $\boldsymbol{\pi}(t)$ and $\boldsymbol{\nu}(t)$ are the optimization variables in the optimization problem. Since **P** exists the dynamic coupling between the energy consumed by IoT device and the offloading duration of IoT device, finding the optimal values of decision variables is actual difficult. Thus, a Computation Tasks Maximum Offloading Algorithm (CTMOA) is proposed.

This algorithm can minimizes the *drift plus penalty*'s upper bound. Through the observation of **P**, we find that the value of

$\nu_i(t)$ hinges on the range of $\pi_i(t)$. Thus, two sub-problems can be separated out of the objective optimization problem and each can be addressed separately. Next, we notice that the portion of **P** associated with the variable $\pi_i(t)$. Thus, the problem **P** is transformed into **SP1**, and we apply the knapsack theory to find a sub-optimal solution of $\boldsymbol{\pi}(t)$ in **SP1**. After solving the problem **SP1**, the problem **P** is equal to **SP2**.

Offloading Strategy Allocation: Similar to [7], problem **SP1** can be treated as the divisible knapsack problem, which can be expressed as

$$\begin{aligned}\textbf{SP1:}\ &\min_{\boldsymbol{\pi}(t)} \sum_{i=1}^{k} \varphi_i(t)\pi_i(t),\\ \text{s.t.}\ & 0 \le \pi_i(t) \le \min\left\{\frac{G_i(t)+A_i(t)}{R_i(t)}, \tau\right\},\\ & \sum_{i=0}^{k} \pi_i(t) \le S(t)\tau, \end{aligned} \tag{29}$$

where

$$\varphi_i(t) = (G_i(t) + A_i(t) + V)\left(\frac{P_{i,o}f_{i,l}}{c_i P_{i,l}} - R_i(t)\right),$$

and $S(t)\tau$ is the knapsack capacity. From the constraints it is clear that $\min\{\tau, (G_i(t) + A_i(t))/R_i(t)\}$ is the size and $\varphi_i(t)$ is the unit value of each item in the knapsack problem. The solving process of **SP1** is shown in Lines 5-16 in Alg. 1.

Local Computation Allocation: After determining the optimal $\boldsymbol{\pi}(t)$ in **SP1**, the value ranges of $\nu_i(t)$ can be fixed. Then, we formulate **SP2**.

$$\begin{aligned}\textbf{SP2:}\ &\min_{\boldsymbol{\nu}(t)} \sum_{i=1}^{k}\left(\frac{\alpha}{2}\nu_i^2(t) + \psi_i(t)\nu_i(t)\right),\\ \text{s.t.}\ & \pi_i(t)P_{i,o} + P_c\tau \le \nu_i(t) \le \min\left\{J_i(t), \pi_i(t)P_{i,o}\right.\\ &\left. + P_c\tau + \frac{(G_i(t)+A_i(t)-R_i(t)\pi_i(t))c_iP_{i,l}}{f_{i,l}}\right\},\\ & \pi_i(t)P_{i,o} + P_c\tau \le \nu_i(t) \le \pi_i(t)P_{i,o} + P_{i,l}\tau + P_c\tau \end{aligned} \tag{30}$$

where

$$\psi_i(t) = \alpha(M_i(t) + \sigma - J_i(t) - EH_i(t)) - \frac{(G_i(t)+A_i(t)+V)f_{i,l}}{c_iP_{i,l}}.$$

Objective function in **SP2** is a quadratic function. According to the quadratic function theorem, we can obtain the theoretical optimum, which is $\nu_i(t) = -\frac{1}{\alpha}\psi_i(t)$. Considering that the value range of $\nu_i(t)$ is limited as s.t., the actual optimum of $\nu_i(t)$ is discussed as Lines 17 through 25 in the Alg. 1.

In this section, this study first construct a "drift plus penalty" function to integrate the long-term queue stability constraint and the long-term energy power constraint into the optimization objective. The original problem is transformed into a new optimization problem which only depends on the information of current time slot and next time slot. And then, to further eliminate the dependence on the information of next time slot, we find the upper bound for the "drift plus penalty" function, and turn to minimize the upper bound instead of minimizing the "drift plus penalty" function. By deforming the original problem twice, we eliminate the dependence of the original problem on future information. The algorithm is designed to solve the optimization problem after twice transformation, so it can determine the offloading decision in real-time without knowing any statistical information about the system.

Algorithm 1 Computation Tasks Maximum Offloading Algorithm (CTMOA)

```
Input: $S(t)\tau, A_i(t), J_i(t), G_i(t), P_c, P_{i,o}, P_{i,l}, f_{i,l}, \alpha, c_i$;
Output: optimal solution $\boldsymbol{\pi}^*(t), \boldsymbol{\nu}^*(t)$;
1: for all $i \in K$ do
2:     Calculate $R_i(t), \varphi_i(t)$ and $\psi_i(t)$;
3:     $\pi_i^*(t) \leftarrow 0$;
4: end for
5: $\varphi_i(t) \leftarrow$ Sort all devices in the ascending order of $\varphi_i(t)$;
6: $\mathbb{C} \leftarrow S(t)\tau$;
7: while $\mathbb{C} > 0$ do
8:     for all $i \in K$ do
9:         if $\varphi_i(t) < 0$ then
10:            $\pi_i^*(t) \leftarrow \min\left\{\mathbb{C}, \frac{J_i(t)-P_c\tau}{P_{i,o}}, \frac{G_i(t)+A_i(t)}{R_i(t)}\right\}$;
11:            $\mathbb{C} \leftarrow \mathbb{C} - \pi_i^*(t)$;
12:        else
13:            break
14:        end if
15:    end for
16: end while
17: for all $i \in K$ do
18:    if $-\frac{1}{\alpha}\psi_i(t) < \max\{\pi_i^*(t)P_{i,o}, J_i(t) - P_c\tau\}$ then
19:        $\nu_i(t) \leftarrow \max\{\pi_i^*(t)P_{i,o}, -\frac{1}{\alpha}\psi_i(t)\}$;
20:    else
21:        $\nu_i(t) \leftarrow \frac{G_i(t)+A_i(t)-R_i(t)\pi_i(t)}{f_{i,l}} c_iP_{i,l} + P_{i,o}\pi_i(t)$;
22:        $\nu_i(t) \leftarrow \min\{\nu_i(t), J_i(t) - P_c\tau, P_{i,l}\tau + P_{i,o}\pi_i(t)\}$;
23:    end if
24:    $\nu_i^*(t) \leftarrow \nu_i(t) + P_c\tau$;
25: end for
```

V. Performance Analysis for CTMOA

We first validate the CTMOA's performance by mathematical analysis in this section. It is proved that CTMOA brings the system throughput approach to the optimal value. Also, it ensures that the virtual and task queues have upper bounds.

For the sake of proving that the battery power satisfies the minimum threshold constraint in long time scales, the following Lemma 1 is deduced.

Lemma 1: The constraint

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} J_i(t) \ge \sigma$$

is satisfied if the virtual queue in (20) is upper bounded. Assume the virtual queue M has an upper constraint of M>0.

Proof: Assume the virtual queue M_i has an upper constraint of $M_i^{max} > 0$ for any device $i \in K$. Then, we can obtain

$$M_i(t)/t \leq M_i^{max}/t \rightarrow 0(if \ \ t \rightarrow \infty). \tag{31}$$

Eq. (20) can be rewritten as

$$\begin{aligned} M_i(t+1) &= \max\{M_i(t) + \sigma - J_i(t+1), 0\} \\ &= \begin{cases} M_i(t) + \sigma - J_i(t+1) \\ \quad \text{if} \quad M_i(t) \geq J_i(t+1) - \sigma; \\ 0 \\ \quad \text{if} \quad M_i(t) < J_i(t+1) - \sigma; \end{cases} \end{aligned} \tag{32}$$

With (32), we have

$$\begin{aligned} M_i(t+1) - M_i(t) &= \begin{cases} \sigma - J_i(t+1) \\ \quad \text{if} \quad M_i(t) \geq J_i(t+1) - \sigma \\ -M_i(t) \\ \quad \text{if} \quad M_i(t) < J_i(t+1) - \sigma \end{cases} \\ &= \max\{\sigma - J_i(t+1), -M_i(t)\} \\ &\geq \sigma - J_i(t+1). \end{aligned} \tag{33}$$

We average both sides of (33) over $t = 0$ to $T - 1$ and let T converge to $+\infty$. The following is obtained

$$\begin{aligned} \lim_{T\rightarrow\infty} \frac{M_i(T)}{T} &\geq \sigma - \lim_{T\rightarrow\infty} \sum_{t=1}^{T} \frac{J_i(t)}{T} \\ &= \sigma - \lim_{T\rightarrow\infty} \sum_{t=0}^{T-1} \frac{J_i(t)}{T}. \end{aligned} \tag{34}$$

Combining (31) with (34), we have

$$\lim_{T\rightarrow\infty} \frac{1}{T} \sum_{t=0}^{T-1} J_i(t) \geq \sigma.$$

The lemma is proven. ■

On the basis of Lemma 1, we further depict Theorem 2.

Theorem 2: For the given V, the virtual queue length has upper bound for any device $i \in K$.

Proof: According to (25), we have

$$\begin{aligned} \Delta(\Theta(t)) + V\mathbf{E}\{\zeta(t)|\Theta(t)\} &\leq M_1 + V\mathbf{E}\{\zeta(t)|\Theta(t)\} \\ &+ \alpha \sum_{i=1}^{k} \Big(\frac{1}{2}\nu_i^2(t) + (M_i(t) + \sigma - J_i(t) - EH_i(t))\nu_i(t)\Big), \end{aligned} \tag{35}$$

where $M_1 = \frac{1}{2}\sum_{i=1}^{k}\big((A_i^{max})^2 + (W_i^{max})^2 + 2G_i(t)A_i^{max}\big) + \frac{\alpha}{2}\sum_{i=1}^{k}\{\sigma^2 + J_{max} + (EH_i^{max})^2 + 2J_{max}EH_i^{max}\} + \alpha\sum_{i=1}^{k}(\sigma - J_i(t))M_i(t)$. The inequality's R.H.S. in (35) is convex, i.e., the first order derivative is increasing.

Let

$$\begin{aligned} y = \alpha \sum_{i\in K} \left(\frac{1}{2}\nu_i^2(t) + (M_i(t) + \sigma - J_i(t) - EH_i(t))\nu_i(t)\right) \\ + V\mathbf{E}\{\zeta(t)\}. \end{aligned}$$

and

$$\begin{aligned} \frac{\partial y}{\partial \nu_i(t)} &= \alpha(\nu_i(t) + M_i(t) + \sigma - J_i(t) - EH_i(t)) \\ &- \frac{Vf_{i,l}}{c_i P_{i,l}}. \end{aligned} \tag{36}$$

Obviously, the first order derivative $\frac{\partial y}{\partial \nu_i(t)}$ is increasing. y is convex. In the proposed system model, $\nu_i(t)$ denotes the total energy consumption for local processing and remote offloading of device i in time slot t. Our strategy is to determine the communication resource allocation $\pi_i(t)$ for each device firstly, and then determine the total energy consumption $\nu_i(t)$. According to (12), we can see that once the optimal total energy consumption $\nu_i^*(t)$ is calculated, the local computation energy consumption can be obtained and thus the local processing task can be calculated. Therefore, $\nu_i(t) \geq \pi_i(t)P_{i,o}$. Then, we evaluate the value of the first order derivative at $\pi_i(t)P_{i,o}$, i.e., $\frac{\partial y}{\partial \nu_i(t)}\Big|_{\nu_i(t)=\pi_i(t)P_{i,o}}$.

Case 1: When $\frac{\partial y}{\partial \nu_i(t)}\Big|_{\nu_i(t)=\pi_i(t)P_{i,o}} < 0$, the optimal consumption power $\nu_i^*(t) > \pi_i(t)P_{i,o}$ and $\frac{\partial y}{\partial \nu_i(t)}\Big|_{\nu_i(t)=\nu_i^*(t)} \leq 0$. According to (36), we have

$$\begin{aligned} M_i(t) &\leq J_i(t) + EH_i(t) - \sigma + \frac{Vf_{i,l}}{\alpha c_i P_{i,l}} \\ &\leq J_{max} + EH_i^{max} + \frac{Vf_{i,l}}{\alpha c_i P_{i,l}}. \end{aligned} \tag{37}$$

Case 2: When $\frac{\partial y}{\partial \nu_i(t)}\Big|_{\nu_i(t)=\pi_i(t)P_{i,o}} \geq 0$, the optimal consumption power $\nu_i^*(t) = \pi_i(t)P_{i,o}$. According to (13), we have

$$J_i(t+1) = \min\big\{J_i(t) - \pi_i(t)P_{i,o} + EH_i(t), J_{max}\big\}. \tag{38}$$

Definitely existing a t_0 satisfies **Case 1** in long time scales, i.e.,

$$M_i(t_0) \leq J_i(t_0) + EH_i(t_0) - \sigma + \frac{Vf_{i,l}}{\alpha c_i P_{i,l}}. \tag{39}$$

According to (20), if $J_i(t+1) \leq \sigma$, we have

$$M_i(t) \leq M_i(t+1) \leq M_i(t) + \sigma. \tag{40}$$

When $\nu_i^*(t) = \pi_i(t)P_{i,o}$ and $M_i(t)$ satisfies (37), we know that $J_i(t+1) \geq \sigma$ with at most T_i time slots. Thus, $\exists t > t_0$ and

$$\begin{aligned} M_i(t) &\leq J_i(t) + EH_i(t) - \sigma + \frac{Vf_{i,l}}{\alpha c_i P_{i,l}} + T_i\sigma \\ &\leq J_{max} + EH_i^{max} + \frac{Vf_{i,l}}{\alpha c_i P_{i,l}} + T_i\sigma, \end{aligned} \tag{41}$$

where $T_i = \sigma/|EH_i^{min} - \pi_i^{min}P_{i,o}|$ are fixed constant.

In all, according to (37) and (41), $M_i(t)$ is upper bounded. This completes the proof. ■

Let $\overline{G}$ be the time average queue length of all the devices, which is defined as

$$\overline{G} = \lim_{T\rightarrow\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i\in K} \mathbf{E}\{G_i(t)\}.$$

Lemma 2: For any request arrival rate $\lambda \in \Lambda$, Λ represents the capacity of the MEC system. There is an optimal policy π^* that satisfies the following conditions.

$$\mathbf{E}\left\{\zeta^{\pi^*}(t)\right\} = \zeta^*(\lambda), \tag{42}$$

where $\zeta^*(\lambda)$ represents the optimal total offloading tasks when the tasks arrival rate is λ.

Proof: Carassiodor's theorem can prove Lemma 2 [33]. Thus, we not describe it in detail here. ■

We define the upper boundary of ζ as $\hat{\zeta}$ and the lower boundary of ζ as $\check{\zeta}$. According to Lemma 2 and Theorem 2, we can propose Theorem 3.

Theorem 3: If $\exists\varepsilon$ satisfies $\lambda + \varepsilon \in \Lambda$, $\overline{G}$ satisfies

$$\overline{G} \leq \frac{M_2 + V\left(\hat{\zeta} - \check{\zeta}\right)}{\rho^*}, \tag{43}$$

where $M_2 = M_1 + \alpha\sum_{i=1}^{k}\left(\frac{1}{2}(J_i(t))^2 + \left(M_i^{max} + \sigma\right)J_i(t)\right)$ and $\rho^* = \sum_{i\in K} W_i^{min}$.

In addition, function ζ of CTMOA satisfies

$$\zeta^{CTMOA} \leq \frac{M_2}{V} + \zeta^*. \tag{44}$$

Proof: Suppose that $\lambda + \varepsilon \in \Lambda$ is satisfied by the offloading policy π'. Quoting Lemma 2, we have

$$\mathbf{E}\left\{\zeta^{\pi'}(t)\right\} = \zeta^*(\lambda + \varepsilon), \tag{45}$$

According to (25) and Theorem 2, we have

$$\begin{aligned}
&\Delta(\Theta(t)) + V\mathbf{E}\{\zeta(t)\} \leq M_1 + V\mathbf{E}\left\{\zeta^{\pi'}(t)\right\}\\
&\quad + \alpha\sum_{i=1}^{k}\left(\frac{1}{2}\nu_i^2(t) + (M_i(t) + \sigma - J_i(t) - EH_i(t))\nu_i(t)\right)\\
&\quad - \sum_{i\in K}\mathbf{E}\left\{W_i^{\pi'}(t)(A_i(t) + G_i(t))|\Theta(t)\right\}\\
&\leq V\mathbf{E}\left\{\zeta^{\pi'}(t)\right\} + \alpha\sum_{i=1}^{k}\left(\frac{1}{2}(J_i(t))^2 + (M_i(t) + \sigma)J_i(t)\right)\\
&\quad - \sum_{i\in K}\mathbf{E}\left\{G_i(t)\left(W_{i,l}^{l\pi'}(t) + W_{i,o}^{t\pi'}(t)\right)|\Theta(t)\right\} + M_1.
\end{aligned} \tag{46}$$

Substituting (45) to the R.H.S of (46) and letting $M_2 = M_1 + \alpha\sum_{i=1}^{k}\left(\frac{1}{2}(J_i(t))^2 + \left(M_i^{max} + \sigma\right)J_i(t)\right)$, we have

$$\begin{aligned}
\Delta(\Theta(t)) + V\mathbf{E}\{\zeta(t)\} &\leq M_2 + V\zeta^*(\lambda + \varepsilon)\\
&\quad - \rho^*\sum_{i\in K}\mathbf{E}\{G_i(t)\},
\end{aligned} \tag{47}$$

where ρ^* is minimum value of tasks processed in all time, which is satisfied to

$$\rho^* = \sum_{i\in K} W_i^{min} \leq \sum_{i\in K}\left(W_{i,l}^{l\pi'}(t) + W_{i,o}^{t\pi'}(t)\right).$$

Moving $V\mathbf{E}\{\zeta(t)\}$ to the R.H.S of (47) and sorting it out, we have

$$\begin{aligned}
\Delta(\Theta(t)) &\leq M_2 + V(\zeta^*(\lambda + \varepsilon) - \mathbf{E}\{\zeta(t)\})\\
&\quad - \rho^*\sum_{i\in K}\mathbf{E}\{G_i(t)\}\\
&\leq M_2 + V\left(\hat{\zeta} - \check{\zeta}\right) - \rho^*\sum_{i\in K}\mathbf{E}\{G_i(t)\}.
\end{aligned} \tag{48}$$

For generality, we suppose that the queue length of devices are empty at initial state of the system, i.e., $G_i(0) = 0, \forall i \in K$. Thus, $L(\Theta(0)) = 0$. Summing both sides of (48) from slot 0 to $T-1$, we have

$$\begin{aligned}
\rho^*\sum_{t=0}^{T-1}\sum_{i\in K}\mathbf{E}\{G_i(t)\} &\leq T\left(M_2 + V\left(\hat{\zeta} - \check{\zeta}\right)\right) - L(\Theta(T))\\
&\leq T\left(M_2 + V\left(\hat{\zeta} - \check{\zeta}\right)\right).
\end{aligned} \tag{49}$$

Dividing both sides of (49) by $T\rho^*$, we can obtain (43) in Theorem 3.

Considering that there exists a fact $\mathbf{E}\{G_i(t)\} \geq 0$, summing both sides of (48) over time slots and deflating the inequality, we have

$$V\sum_{t=0}^{T-1}\mathbf{E}\{\zeta(t)\} \leq T(M_2 + V\zeta^*(\lambda + \varepsilon)). \tag{50}$$

Dividing both sides of (50) by *VT*, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{E}\{\zeta(t)\} \leq \frac{M_2}{V} + \zeta^*(\lambda + \varepsilon). \tag{51}$$

Letting $T \to \infty, \varepsilon \to 0$, we can obtain (44).

This completes the proof. ■

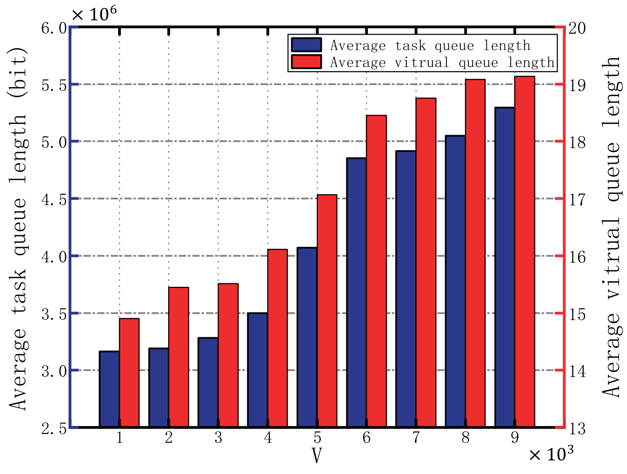
TABLE I
SIMULATION EXPERIMENT PARAMETERS

$f_{i,l} \sim U[0.5, 1]$ GHz	$c_i \sim U[1000, 3000]$ cycle/bits
$P_{i,o} \sim U[300, 500]$ mW	$EH_i(t) \sim U[1.5, 2.5]$ J
$h_i(t) \sim E(1)$ [22]	$S(t) \sim U[k/2, k]$
$N_0 = 10^{-6}$ W/Hz	$\xi = 10^{-27}$ [24, 31]
$\tau = 1$ s	$B = 1$ MHz

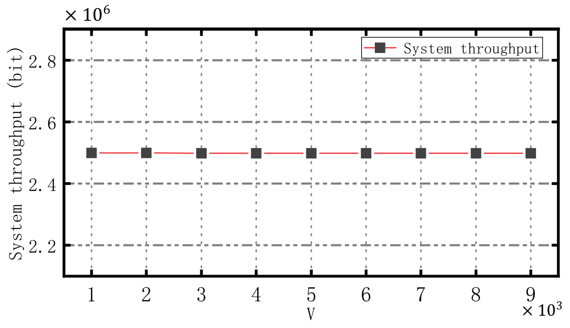
VI. SIMULATION EXPERIMENTS FOR CTMOA

In this section, we run a series of experiments to estimate the performance of CTMOA. The efficiency of this system is analyzed by modifying different parameters. Different 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 industrial manufacturing scenes. Different energy arrival rate reflects the energy harvested by the device, which is usually measured by the amount of energy. We set different device numbers and device energy thresholds to check the robustness of the system.

In each time slot, the amount of tasks that arrive at device i is set to be uniformed distributed within [1000, 4000] bits. The different number of IoT devices are considered, and k is set as 60 to 120 with an increment of 10. The values of other parameters are listed in Table I.

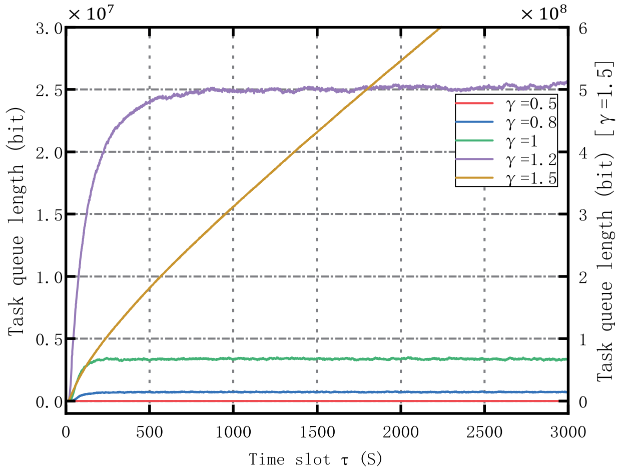


(a) Average queue length

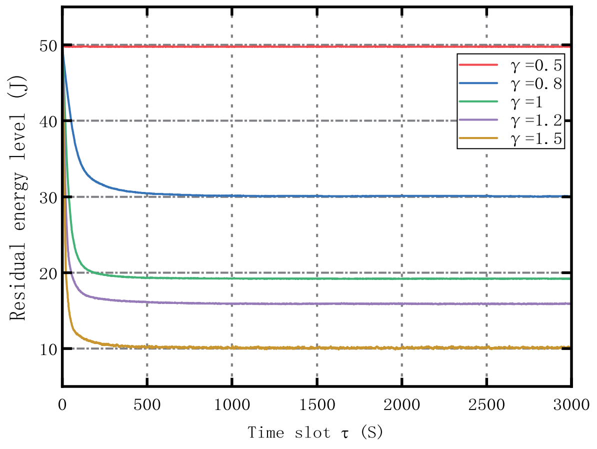


(b) System throughput

Fig. 3. Average queue length and system throughput with different values of V.



(a) Task queue length



(b) Residual energy level

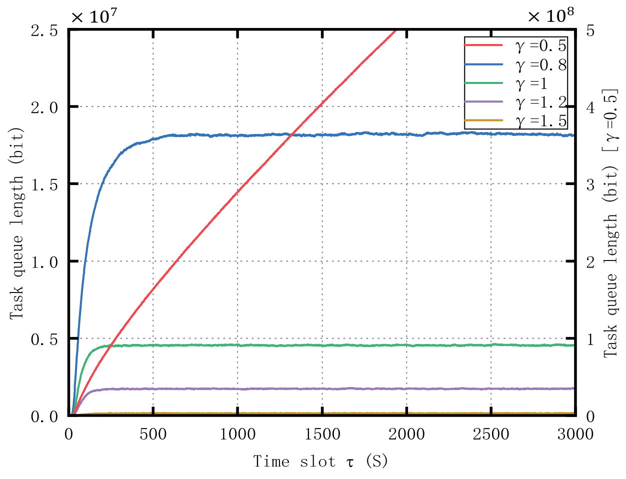
Fig. 4. Task queue length and residual energy level with different task arrival rates.

A. Analysis on Trade-Off Parameter V

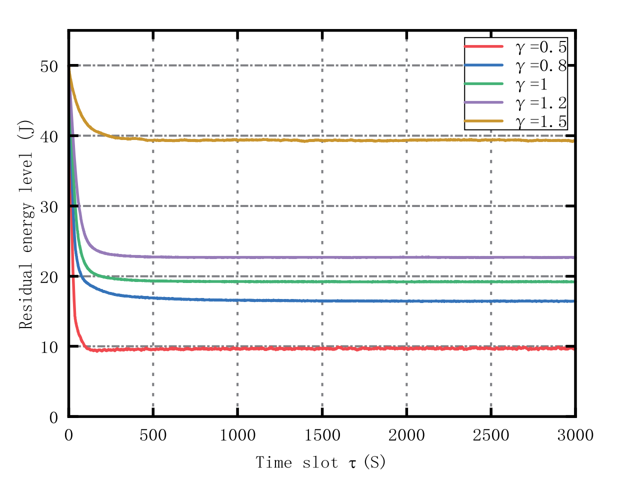
In this group of experiments, we plot the average queue length and the system throughput under different values of V. The system throughput and the queue stability are balanced utilizing the parameter V. Fig. 3(a) shows the task queue and the virtual queue length per time slot under different V. As V becomes larger, both the two types of queues show an upward trend. This is because the effect of the queue length on the objective function becomes smaller compared with the system throughput when V increases. Fig. 3(b) shows the system throughput when the system reaches a steady state. Parameter V is a trade-off between the queue length and the system throughput. However, from the experimental results shown in Fig. 3(a), we can observe that the change of V affects the queue length obviously, yet the system throughput does not show obvious change with the increment of V in Fig. 3(b). That is because one goal of our problem is to maintain the queue length stability. When the queue length reaches the steady state, the average system throughput should be equal to the average amount of task arriving. When the throughput is greater or less than the average amount of task arriving at each device for a long period of time, it will lead to unstable the task queue and battery energy level. Therefore, under the premise of ensuring system stability, the system throughput will eventually approach the average task amount level of device arrival regardless of the value of V. In the actual scenario, setting a moderate V can reduce the size of the queue backlog required by the devices on the basis of the network congestion degree and the task arrival rates of the devices. This phenomenon is similar to that presented in [18].

B. Analysis on Task Arrival Rate

With different task arrival rates, this set of experiments depicts the task queue state and the residual battery energy. The amount of arrived data $A_i(t)$ are multiplied by a coefficient γ where $\gamma = 0.5, 0.8, 1, 1.2$ and 1.5, respectively. In Fig. 4(a), the scale axis of the curve $\gamma = 1.5$ is the right scale axis, and this set of curves shows that the task queue length grows as the arrival rate grows. This is because the amount of generated tasks by the devices increases with the arrival rate monotonically in all situations. However, when $\gamma = 1.5$, the devices cannot handle so many tasks, which makes the queue length unstable. This is because there are not sufficient power, computation, and communication resource to support the processing of these tasks. Fig. 4(b) shows that the residual battery power of the device decreases as the arrival rate increases. The battery power level is well maintained around the threshold. This is because the CTMOA algorithm prior ensures the stability of the battery energy. When $\gamma = 1.5$,

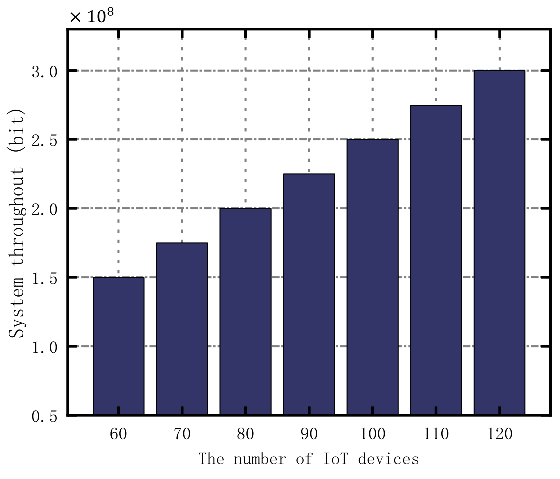


(a) Task queue length

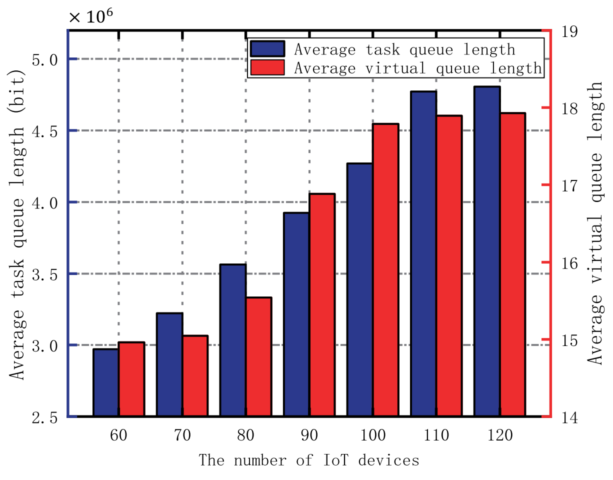


(b) Residual energy level

Fig. 5. Task queue length and residual energy level with different energy harvesting rates.



(a) System throughput



(b) Average queue length

Fig. 6. System throughput and average queue length with different numbers of IoT devices.

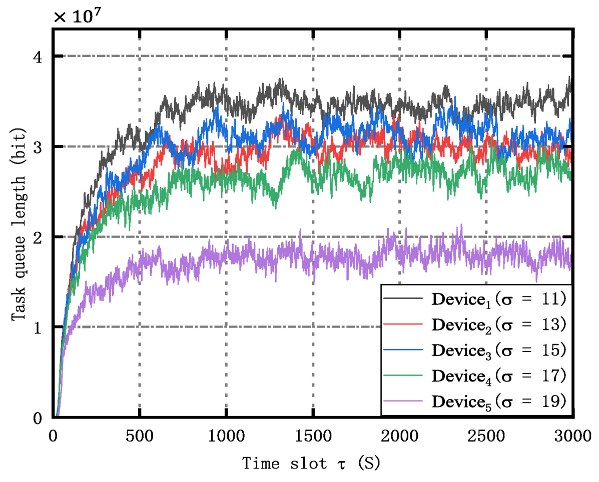
owing to the numerous tasks arriving, the energy required to offload tasks is almost inadequate and the battery is kept at a minimum energy level.

C. Analysis on Energy Harvesting Rate

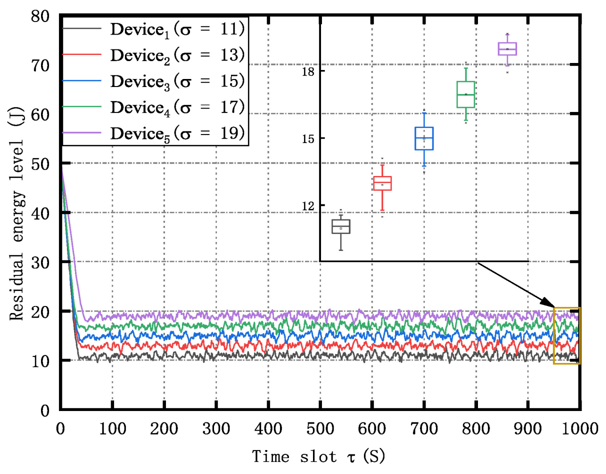
In this group of experiments, we plot the task queue length and the residual battery energy with different energy harvesting rates. $EH_i(t)$ is multiplied by a coefficient γ where $\gamma = 0.5, 0.8, 1, 1.2$ and 1.5, respectively. In Fig. 5(a), the scale axis of the curve $\gamma = 0.5$ is the right scale axis. This set of curves shows that the residual battery energy of the device increases and the task queue length of the device decreases as the energy harvesting rate increases. This is because energy harvested by the devices increases with the harvesting rate monotonically in all situations. The devices have enough power to support processing these tasks so that the task queue backlog decreases. However, when $\gamma = 0.5$, the devices do not have sufficient power to handle these tasks, and the task queue is unstable. Hence, the curve $\gamma = 0.5$ is only maintained around the threshold in Fig. 5(b). In addition, the two groups of experiments mentioned above show that CTMOA enables the system residual energy level to converge quickly to around the threshold. This demonstrates that CTMOA can dynamically adjust offloading decisions and adapt to the change in the arrival rate.

D. Analysis on the Number of IoT Devices

In this group of experiments, we plot the task queue length and the system throughput with different number of the devices. The IoT devices is set from 60 to 120 in step of 10. Fig. 6(a) shows that the system throughput rises as amount of devices rises. Fig. 6(b) shows both the two types of queues backlog per time slot under the different numbers of the devices. With the number of devices increases, both the two types of queue show an upward trend. This is because there is no change in the system computation and communication resources. The tasks that are not offloaded in time are saved in the task queue, which incurs the queue backlog increasing. As the queue buffer grows, more energy is needed to offload these tasks, which is why the virtual queue length grows. From Fig. 6(a) and Fig. 6(b), it is clear that CTMOA adapts to different IoT device volumes and allows for maximum task offloading while keeping the queue stable.



(a) Task queue length



(b) Residual energy queue level

Fig. 7. Task queue length and residual energy level with different energy level threshold and single IoT device.

E. Analysis on the Energy Level Threshold

In this group of experiments, we plot the changing trend of the task queue length and the residual battery energy of five devices whose energy level thresholds σ are set as 11, 13, 15, 17, and 19, respectively. Fig. 7(a) shows that CTMOA can effectively stabilize the queue length of single device. After a period of time (1000 time slots), CTMOA ensures that the task queue stabilizes under all cases. In the ideal case that the local processing ability, channel state, transmission power consumption, and amount of tasks arriving are consistent across devices, the task queue length of a single IoT device increases as σ increases. This is because the energy level threshold of the IoT device affects the consumed energy, and further affects the number of tasks offloaded. However, in reality, all devices are heterogeneous, the channel state is unpredictable, and the amount of tasks arriving is random. The amount of the offloaded tasks of each device is different. Accordingly, the task queue of each device does not stabilize at the expected length. Fig. 7(b) shows that the residual energy of the individual IoT device is well maintained at different energy level thresholds. The residual energy rapidly converges with time and eventually reaches the energy level threshold. We also show the box plot of the last 500 time slots, which shows that the medium value is around the energy level threshold and the degree of fluctuation in the residual energy of each IoT device is very stable. From Fig. 7(a) and Fig. 7(b), we can see that under different energy level thresholds, CTMOA can control the residual energy level of all the devices effectively for all cases. Therefore, CTMOA can improve the system throughput effectively. Moreover, CTMOA enables the system to make offloading decisions dynamically. The residual battery energy is brought around the threshold in different threshold situations.

VII. Conclusion

In this paper, we investigate the task offloading problem for an MEC system with multiple EH devices. For the dynamic system status and stochastic energy harvesting, we apply Lyapunov optimization theory, and exploit a computation task maximum offloading algorithm (CTMOA) to maximize the system throughput while maintaining the system stability and device battery energy level constraint in long time scales. The CTMOA algorithm can determine the offloading decision in real-time without knowing any statistical information about the system. The simulation experiment results demonstrate that CTMOA can increase the system throughput while guaranteeing system stability and the device battery energy level constraint.

References

[1] D. Chen et al., "S2M: A lightweight acoustic fingerprints-based wireless device authentication protocol," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 88–100, Feb. 2017.

[2] H. Duan, Y. Zheng, C. Wang, and X. Yuan, "Treasure collection on foggy islands: Building secure network archives for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2637–2650, Apr. 2019.

[3] 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. IEEE Int. Conf. Web Serv. (ICWS)*, 2019, pp. 193–200.

[4] K. Zhang, X. Gui, D. Ren, J. Li, J. Wu, and R. Dongsheng, "Survey on computation offloading and content caching in mobile edge networks," *J. Softw.*, vol. 30, no. 8, pp. 2491–2516, 2019.

[5] R. Xie, X. Lian, Q. Jia, T. Huang, and Y. Liu, "Survey on computation offloading in mobile edge computing," *J. Commun.*, vol. 39, no. 11, pp. 138–155, 2018.

[6] Z. Sheng, C. Mahapatra, V. C. M. Leung, M. Chen, and P. K. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 114–126, Jan.–Mar. 2018.

[7] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1050–1060, Jul.–Sep. 2021.

[8] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 1451–1455.

[9] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[10] B. P. Rimal, D. Pham Van, and M. Maier, "Mobile-edge computing versus centralized cloud computing over a converged FiWi access network," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 498–513, Sep. 2017.

[11] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

[12] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[13] 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.

[14] J. Zhang, Y. Shen, Y. Wang, X. Zhang, and J. Wang, "Dual-timescale resource allocation for collaborative service caching and computation offloading in IoT systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1735–1746, Feb. 2023.

[15] J. Zhang, J. Du, Y. Shen, and J. Wang, "Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9303–9317, Oct. 2020.

[16] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.

[17] Q. An, J. Wang, Z. Zhang, and Y. Shen, "Information-based bit allocation for cooperative visual sensing in vehicular networks," *IEEE Trans. Veh. Technol.*, early access, Sep. 30, 2022, doi: 10.1109/TVT.2022.3211176.

[18] 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.

[19] W. Zhou, L. Xing, J. Xia, L. Fan, and A. Nallanathan, "Dynamic computation offloading for MIMO mobile edge computing systems with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 5172–5177, May 2021.

[20] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[21] S. Dai, M. Li Wang, Z. Gao, L. Huang, X. Du, and M. Guizani, "An adaptive computation offloading mechanism for mobile health applications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 998–1007, Jan. 2020.

[22] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[23] 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.

[24] F. Zhao, Y. Chen, Y. Zhang, Z. Liu, and X. Chen, "Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 2154–2165, Jun. 2021.

[25] C. Qiu, Y. Hu, and Y. Chen, "Lyapunov optimized cooperative communications with stochastic energy harvesting relay," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1323–1333, Apr. 2018.

[26] Z. Tong, J. Cai, J. Mei, K. Li, and K. Li, "Dynamic energy-saving offloading strategy guided by Lyapunov optimization for IoT devices," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 19903–19915, Oct. 2022.

[27] M. Guo, W. Wang, X. Huang, Y. Chen, L. Zhang, and L. Chen, "Lyapunov-based partial computation offloading for multiple mobile devices enabled by harvested energy in MEC," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 9025–9035, Jun. 2022.

[28] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163–2176, Feb. 2021.

[29] J. Du, C. Jiang, A. Benslimane, S. Guo, and Y. Ren, "SDN-based resource allocation in edge and cloud computing systems: An evolutionary Stackelberg differential game approach," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1613–1628, Aug. 2022.

[30] J. Ren, K. M. Mahfujul, F. Lyu, S. Yue, and Y. Zhang, "Joint channel allocation and resource management for stochastic computation offloading in MEC," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8900–8913, Aug. 2020.

[31] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 570–584, Apr.–Jun. 2020.

[32] P. Cai, F. Yang, J. Wang, X. Wu, Y. Yang, and X. Luo, "JOTE: Joint offloading of tasks and energy in fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3067–3082, Apr. 2020.

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

[34] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[35] X. Lyu et al., "Optimal schedule of mobile edge computing for Internet of Things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov. 2017.

[36] 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.

[37] M. Merluzzi, P. di Lorenzo, and S. Barbarossa, "Latency-constrained dynamic computation offloading with energy harvesting IoT devices," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2019, pp. 750–755.

[38] M. Sun, X. Xu, Y. Huang, Q. Wu, X. Tao, and P. Zhang, "Resource management for computation offloading in D2D-aided wireless powered mobile-edge computing networks," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8005–8020, May 2021.

[39] T. D. P. Perera, D. N. K. Jayakody, S. K. Sharma, S. Chatzinotas, and J. Li, "Simultaneous wireless information and power transfer (SWIPT): Recent advances and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 264–302, 1st Quart., 2018.

[40] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[41] Y. Ye, R. Q. Hu, G. Lu, and L. Shi, "Enhance latency-constrained computation in MEC networks using uplink NOMA," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2409–2425, Apr. 2020.

Jing Mei received the Ph.D. degree in computer science from Hunan University, China, in 2015. She is currently an Associate Professor with the College of Information Science and Engineering, Hunan Normal University. Her research interests include cloud computing, fog computing and mobile edge computing, high performance computing, task scheduling, and resource management. She has published more than 20 research articles in international conference and journals, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEM, IEEE TRANSACTIONS ON SERVICE COMPUTING, *Cluster Computing*, *Journal of Grid Computing*, and *Journal of Supercomputing*.

Longbao Dai received the B.S. degree in computer science and technology from the Hunan University of Science and Engineering, Yongzhou, China, in 2021. He is currently pursuing the M.S. degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research interests focus on distributed parallel computing, modeling and resource pricing and allocation in mobile edge computing systems, and combinatorial optimization.

Zhao Tong received the B.Sc. degree in computer science from the Beijing Institute of Technology in 2007, and the Ph.D. degree in computer science from Hunan University, China, in 2014. He is currently an Associate Professor with the College of Information Science and Engineering, Hunan Normal University. His research interests include modeling and scheduling for parallel and distributed computing systems, parallel system reliability, and parallel algorithms.

Xin Deng received the B.S. degree in computer science and technology from Hengyang Normal University, Hengyang, China, in 2020. She is currently pursuing the M.S. degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. Her research interests focus on distributed parallel computing, modeling and resource pricing and allocation in mobile edge computing systems, and game theory.

Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor of Computer Science with the State University of New York. He is also a National Distinguished Professor with Hunan University, China. He has authored or coauthored over 890 journal articles, book chapters, and refereed conference papers. He holds nearly 70 patents announced or authorized by the Chinese National Intellectual Property Administration. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing. He has received several best paper awards. He is among the world's top 5 most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He is an AAAS Fellow and an AAIA Fellow. He is also a member of Academia Europaea (Academician of the Academy of Europe).