Dynamic Energy-Saving Offloading Strategy Guided by Lyapunov Optimization for IoT Devices

Zhao Tong, *Member, IEEE*, Jinhui Cai, Jing Mei, Kenli Li, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—In the Internet of Everything era, various Internet of Things (IoT) devices have become popular, and the number of computing-intensive applications has increased substantially. As an emerging technology, mobile-edge computing (MEC) gives network edge nodes stronger computing and storage capabilities, bringing users a good Quality of Experience (QoE). By offloading some computing tasks to the edge for processing, the burden on IoT devices can be effectively reduced. However, this approach exacerbates the computing and storage resource depletion of the MEC server and the bandwidth and transmission cost of the wireless link used to offload computing tasks. Additionally, making an offloading decision online without future system status information is a considerable challenge. Therefore, we should study and design a reasonable offloading strategy to reduce the additional overhead, which is of significance. We establish a virtual queue model to describe the workload offloading problem of IoT devices in a two-layer MEC network. This is a stochastic optimization problem. Based on Lyapunov optimization, we transform the research problem into a deterministic optimization problem. A Lyapunov online energy consumption optimization algorithm (LOECOA) is proposed to effectively balance the system's queue backlog and energy consumption. Based on theoretical analysis and a large number of experimental and numerical results, our algorithm performs better on energy consumption while satisfying the system constraints under a dynamic task arrival rate.**

***Index Terms*—Energy consumption, Lyapunov optimization, mobile-edge computing (MEC), online calculation offloading.**

I. Introduction

With the maturity of fifth-generation (5G) mobile communication network technology and the continuous enhancement of social communication infrastructure, the network connection speed and the mutual perception ability of objects have greatly improved [1]. This setting has created convenient conditions for humanity to create data actively, accelerated the development of the Internet of Things (IoT), and triggered significant changes in mobile applications. With the rapid development of embedded sensors [2], wearable devices, and cameras, many new application requirements have emerged, such as face recognition (FR), autonomous driving, virtual reality (VR) games, and smart agriculture [3]. A common characteristic of these application scenarios is that they generate a large quantity of data to process [4], [5]. Currently, the contradictory relationship between IoT devices with limited computing power and computing-intensive applications has become an urgent problem. According to the annual Internet report released by Cisco in 2020, it is estimated that by 2023, there will be more than 29.3 billion IoT devices on the Internet. The large number of mobile devices will generate countless application data [6], [7]. IoT devices need powerful computing power to handle complex and diverse computing-intensive applications, and they also consume considerable energy. To provide a satisfactory Quality of Experience (QoE), people urgently need new technology.

Mobile-edge computing (MEC) has frequently been mentioned as a new technology to solve this issue. MEC refers to the deployment of servers on the edge side to provide content storage computing and distribution services, so that applications, services, and content can be deployed in a highly distributed environment that better meets IoT device terminals' expectations for low latency and high bandwidth [8]. In contrast to traditional cloud computing, which is far from user equipment, MEC is usually deployed near radio access points, such as base stations (BSs). It acts as a mobile cloud service platform that operates on the edge of the network, synchronizing services and functions that used to reside in cloud data centers to the edge of the network. By making reasonable offloading decisions to improve the user service experience, some tasks are processed on the edge side, which effectively reduces service delivery delays. Additionally, the substantial increase in network data has caused considerable load pressure on the network transmission link and the mobile core network. MEC can respond to users at the edge, reducing the bandwidth requirements of the backhaul network and core network [9].

A. Motivation

Computation offloading can reduce the processing and storage burden of IoT devices and improve the experience of

Manuscript received 16 November 2021; revised 24 February 2022 and 10 April 2022; accepted 16 April 2022. Date of publication 20 April 2022; date of current version 7 October 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB2104000; in part by the Program of the National Natural Science Foundation of China under Grant 62072174 and Grant 61502165; and in part by the National Natural Science Foundation of Hunan Province, China, under Grant 2020JJ5370. *(Corresponding author: Jing Mei.)*

Zhao Tong, Jinhui Cai, and Jing Mei are with the College of Information Science and Engineering, Hunan Normal University, Changsha 410081, Hunan, China (e-mail: tongzhao@hunnu.edu.cn; 202020291627@hunnu.edu.cn; jingmei1988@163.com).

Kenli Li is with the College of Information Science and Engineering and the National Supercomputing Center, Hunan University, Changsha 410082, Hunan, China.

Keqin Li is with the College of Information Science and Engineering and the National Supercomputing Center, Hunan University, Changsha 410082, Hunan, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA.

Digital Object Identifier 10.1109/JIOT.2022.3168968

mobile users. However, when computing-intensive tasks are transferred to the edge side for processing, a large amount of transmission energy will be generated in the process, which is one of the components of the system's total energy consumption. In the real world, the transmission energy of the equipment is determined by a combination of many factors, the most important of which is the channel quality: the better the channel status is, the faster the transmission rate will be. Moreover, the reduction in the time required to transmit data will reduce the amount of energy used for transmission. In contrast, when the channel conditions deteriorate, the transmission of the same amount of data will consume more energy. Therefore, offloading more computing tasks can effectively bring down the total energy consumption of the system when the channel status is good. However, such behavior may cause a surge in the queue backlog and make the system congested and unstable [10].

In some IoT application scenarios, devices are deployed in remote areas and do not have the conditions for frequent charging. Therefore, providing users with good services is important to optimize energy consumption when designing a simple and efficient task offloading strategy [11]. There are also some other challenges. First, the system is in a complex and changeable network environment. Its various status information changes dynamically with time and is affected by many factors, making it difficult to predict accurately in practice [12]. Second, we need to make the energy consumed by the system tend to the optimal solution while retaining the stability of the virtual queue. This requires an offloading strategy to make a tradeoff between the two indicators. Finally, the arrival flow of computing tasks for IoT devices is also difficult to predict. With the rapid growth in IoT devices, the scale of task offloading problems is becoming more substantial [13]. As a result, it is a huge struggle to adjust offloading decisions in the face of a complex environment.

B. Contributions

In this work, we investigate computational task offloading strategies in the context of a two-layer MEC network model. This article aims to make the long-term total energy consumed by the system tend to the optimal solution while ensuring the stability of the virtual queue of each IoT device. We model the problem and formulate it to form a stochastic optimization problem [14]. Using Lyapunov optimization technology, a dynamic offloading algorithm that can effectively reduce energy consumption is designed. The Lyapunov online energy consumption optimization algorithm (LOECOA) does not require any prior system state information. It dynamically makes offloading decisions and adjusts the parameter V to weigh the importance of energy consumption and queue backlog. Both the theoretical derivation and experimental results show the significance of the algorithm. The main contributions and originality of this article are summarized as follows.

1) We construct a two-layer MEC network model and establish the problem as a mathematical model of a multiuser computing offloading. The total energy consumption under the long-term stable system operation is taken as the optimization goal, and an online offloading algorithm called LOECOA is proposed. Under the premise that no prior system information is needed, the algorithm achieves good performance in the face of dynamic traffic arrival scenarios.
2) Specifically, we design that IoT devices have certain computing power in our model, and the generated tasks can be executed on both the MEC side and the local side. Energy consumption comes from task computation and transmission. The strategy could simultaneously maintain overall system stability, balance system energy, and queue backlog.
3) Through the Lyapunov optimization technology, we decouple the primal problem on the time slice and transform it into a deterministic upper bound problem. Under the condition that the long-term constraints of the system are satisfied, the numerical solution is obtained with the help of a heuristic algorithm. And through rigorous mathematical derivation, the difference between the numerical solution obtained and the analytical solution of the original problem is derived.
4) We set extensive experiments to prove the progressive optimality of LOECOA and verify our theoretical analysis results of the system energy consumed and queue backlog. In addition, the superiority of the algorithm is proven by comparison with several benchmark strategies. The results demonstrate that the LOECOA achieves a great improvement in reducing energy and can effectively preserve the system's stability.

The remainder of this article is structured as follows. Section II summarizes some recent developments in edge computing offloading. Section III introduces the system model used in this work and then gives a strictly defined form of the offloading problem. Section IV details how Lyapunov optimization can be applied to solve the problem presented in this article and proposes an algorithm called LOECOA. Sections V and VI present the theoretical analysis and simulation experiments, respectively. The superiority of LOECOA is verified from two aspects. In Section VII, we summarize the article.

II. Related Work

Computation offloading is a key MEC technology, and there are many related research results, including the formulation of offloading decisions and the allocation of available resources. The core of the research on offloading decisions is how to make a suitable offloading decision for users, including the question of whether to offload and how much to offload. The results of the offloading decision are divided into three situations: 1) binary offloading; 2) partial offloading; and 3) local calculation. Mao *et al.* [15] studied a green energy MEC system with an energy collecting device. Additionally, they developed an efficient computational offloading strategy. Based on this model, a dynamic offloading algorithm (LODCO) is proposed to solve the question. The LODCO algorithm makes an offloading decision in every time slot. If the calculation task is executed locally, the mobile user is allocated CPU cycles; otherwise, the mobile user is allocated

transmission power. The calculation task is migrated to the MEC server. According to the results of simulation experiments, the running time of this algorithm can be shortened by 64%. In addition, the algorithm can effectively reduce the loss of data packets during calculation offloading. After completing the offloading decision, the next step is to consider the issue of reasonable resource allocation. The common research methods and models differ for different types of tasks. This part of the research is described in detail in [16] and [17].

The optimization objectives of offloading decisions can be split into three main types: 1) diminishing time delay; 2) diminishing consumed energy; and 3) weighing time delay and energy. Delays occur when the local side sends the calculation tasks to the MEC server, and its size directly affects the user's service experience [18]. To ensure QoE, many research papers aiming to reduce delay and energy have appeared, which involve many different optimization algorithms and application scenarios. Lv *et al.* [19] noted the problem of battery life in unmanned aerial vehicle (UAV) application scenarios. Based on traditional charging methods, the possibility of using green energy is considered. Combined with Lyapunov optimization technology, an online energy management solution was proposed, which can obviously save power resources, thereby raising the service provider's income. The optimal offloading scheme proposed in [20] to reduce time delay considers how to make offloading decisions when MEC has limited computing resources. To make better-offloading decisions in combination with other advanced methods, Bi *et al.* [21] creatively proposed a new computing offloading framework called LyDROO. This work unites the merits of both Lyapunov and DRL methods, and it has good performance in reducing delay overhead. Tong *et al.* [22] used the powerful adaptive capabilities of DRL [23], [24] to extend the application scenario to the cloud environment [25], [26]. A new offloading algorithm DDQNTS based on a dual-Q network was proposed in this article. By comparing several classic algorithms on the Google test set, it is found that DDQNTS can effectively improve the user's QoE on the premise of ensuring a high task completion rate. Anajemba *et al.* [27] presented a Lagrangian suboptimal convergent computation offloading algorithm, which has an excellent performance in energy efficiency and energy consumption reduction. Zhang *et al.* [28] developed a novel strategy on the basis of an intelligent optimization algorithm to optimize energy costs. In this scheme, the link states of a feedforward network and backhaul network are considered simultaneously. In the experiments conducted to verify the offloading scheme, the author compared random offloading and local offloading with other schemes. The offloading scheme based on AFSA has an excellent performance in terms of energy consumed. However, the disadvantage of this scheme is that the algorithm complexity is too high.

Considering complex computing tasks, such as FR systems and the Internet of Vehicles, both energy consumption and delay impact QoE, so energy consumption and delay are comprehensively considered when performing offloading tasks, which is an essential consideration for offloading decisions. Lyapunov optimization technology has excellent performance when optimizing multiple targets simultaneously. It was proposed by Neely and applied to network optimization theory [29]. Mao *et al.* [30] gave a solution based on Lyapunov theory to better balance the system energy consumed and task delay. An algorithm that can dynamically manage radio and computing resources is proposed to address changing computing requirements and wireless fading channels. The algorithm can automatically adapt to the changes of the system network state in each time slice and does not need to manually change some of the control quantities in it. It has the advantage of automatic learning and does not require many prior parameter inputs like most algorithms. It has good adaptability to real-time control of dynamically changing systems and, at the same time, ensures relatively low algorithm complexity.

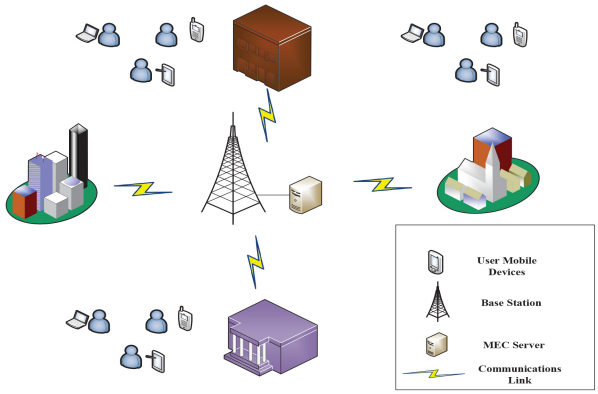


Fig. 1. MEC task offloading system model.

III. System Model and Formal Description

In this section, we construct a two-layer MEC network model and propose a stochastic optimization problem for task offloading in this model.

A. System and Workload Arrival Model

We consider an application scenario, which has some similar IoT devices, and a BS with an MEC server connected through the same local area network (LAN) to form a network system [31]. IoT devices are powered by batteries, and we regard the energy as sufficient. An MEC server provides services to nearby IoT devices, as depicted in Fig. 1. The edge server is connected to the BS over wired transmission, and the BS is given limited edge computing capabilities. Therefore, IoT devices can submit requests through wireless communication and offloading their computing tasks to the corresponding serving BS for processing. When tasks are offloaded for processing, the user equipment can obtain better QoE and reduce energy costs.

Let $N = \{1, 2, \ldots, n\}$ represent the index set of n IoT devices, and let time be a discrete-time model. We divide the runtime of the system into a series of time slots. It is divided into T parts of equal length, each of which has a slot length of τ, and each slot is indexed by $t \in \{0, 1, \ldots, T-1\}$.

Before the start of each period, the size of the computing task generated by device $i \in N$ is represented by $a_i(t)$ (in bits).

$a_i(t)$ is the Poisson distribution expected to be λ_i and satisfies an independent and identical distribution. The basis for our setting is that the Poisson distribution can approximate the probability distribution of the frequency of a discrete event in continuous time. Moreover, many scenes in the real world are related to the Poisson distribution. These settings make our model more practical and widely applicable.

B. Computation Task Model and Offloading Transport Model

To provide better service, tasks arriving in a time slot can be divided and processed, respectively. The CPU of the IoT device can compute the task, or it can also submit the task offloading request to a nearby MEC server for processing.

In time slot t, we define the computing task to be processed as $b_i(t)$. It consists of two parts: some are generated by local calculations, and the size is $b_{U,i}(t)$; and others submit computing offloading requests to the MEC server over wireless channels, and the size of this part is represented by $b_{M,i}(t)$. That is, it is clear that $b_i(t) = b_{U,i}(t) + b_{M,i}(t)$. Similarly, before the start of each period, the mobile device can obtain the environmental information of the current time slot, such as channel conditions, to make offloading decisions more intelligently.

In this model, the channel condition remains static in the current time slot while it changes from time slot to time slot. When the task is offloaded for processing, the data are sent from the IoT device to the BS through the wireless channel and then sent to the edge node through wired transmission. Usually, the energy consumption of wireless transmission accounts for a large proportion, so we solely consider the overhead of the wireless part. In wireless transmission, the data are uploaded through the uplink channel and processed, and then the result is returned through the downlink channel. Since the returned result size is trivial, the downlink energy consumption is also negligible. To simplify the model, this article does not consider the interference between communication channels and the restriction of the amount of subchannels. This assumption can be fulfilled through the use of new technologies. For example, 5G can provide sufficient communication resources. The impact of this simplification on the complexity of the offloading strategy is minimal.

For every IoT device i, its transmit power is defined as $P_i(t)$, and the channel gain of this time slot is represented by $h_i(t)$. Then, the achievable task transmission rate of data on orthogonal channels $r_i(t)$ is given by the Shannon capacity [14]–[16]

$$r_i(t) = \omega \log_2\left(1 + \frac{h_i(t)P_i(t)}{\sigma^2}\right) \tag{1}$$

where ω denotes the channel bandwidth allocated and σ^2 is the white Gaussian noise power. Therefore, the time taken on offloading cannot exceed the length of a time slot τ. Thus, the constraint condition that the size of $b_{M,i}(t)$ should follow is:

$$b_{M,i}(t) \le r_i(t)\tau \quad \forall i \in N. \tag{2}$$

TABLE I
SYMBOLS AND MEANING

Notation	Definition
C	maximum computing resources owned by the MEC server
L	the number of cycles required for the CPU to calculate a bit task
N	IoT devices set
$P_i(t)$	transmission power of device i
$Q_i(t)$	task queue backlog of device i
V	weight factor of average energy consumption to the drift
$W_M(t)$	MEC server energy consumption
$W_{T,i}(t)$	transmission energy consumption of device i
$W_{U,i}(t)$	calculation energy consumption of device i
$a_i(t)$	computing task arrival rate by device i
$b_i(t)$	the total number of computing tasks of device i
$b_{U,i}(t)$	local computing tasks of device i
$b_{M,i}(t)$	computing tasks offloading of device i
$f_i(t)$	CPU calculation rate of device i
$h_i(t)$	channel gain of device i
$r_i(t)$	wireless channel transmission rate of device i
$\Delta\Theta(t)$	Lyapunov function drift
γ	MEC server standby energy consumption
κ	energy cost factor
σ^2	white Gaussian noise power
τ	time slot length
ω	channel bandwidth

The transmission energy consumed by offloading tasks for IoT device i is given as follows:

$$W_{T,i}(t) = P_i(t)\frac{b_{M,i}(t)}{r_i(t)}. \tag{3}$$

Each IoT device has a virtual queue to store tasks that have arrived waiting to be assigned. The task queue backlog size of IoT device i is $Q_i(t)$. To be realistic, each device cannot offloading more than the number of tasks it has at present, so

$$b_{M,i}(t) \le Q_i(t) - b_{U,i}(t) \quad \forall i \in N. \tag{4}$$

Recall that at time slot t, the arrival number of tasks for each queue is $a_i(t)$, and the processing amount is $b_i(t)$. Then, the evolution process of the queue backlog can be given as follows:

$$Q_i(t+1) = \max\{Q_i(t) - b_i(t) + a_i(t), 0\}. \tag{5}$$

During the period of time when the user submits the task, the average number of tasks arriving in each time slot should not exceed the average number of tasks processed, which is as follows:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E[a_i(t) - b_i(t)] \le 0, i \in N. \tag{6}$$

In this way, the queue will not grow indefinitely, the system will remain relatively stable, and the following discussion will be valuable. The definitions of the main symbols appearing in this article are listed in Table I.

C. Energy Consumption Model

In this work, the total energy consumption of the system consists of three components: 1) local side; 2) transmission consumption; and 3) MEC side.

1) Local Side: The energy consumption is generated by computing tasks. Let the IoT device CPU computing rate in time slot t be $f_i(t)$, and let L be the number of cycles required for the CPU to calculate a bit task. Then, the computing task size $b_{U,i}(t)$ of device i in time slot t [30] can be given as follows:

$$b_{U,i}(t) = \frac{f_i(t)\tau}{L}. \tag{7}$$

The energy consumption calculation is as follows:

$$W_{U,i}(t) = \kappa f_i^3(t) \tag{8}$$

where κ represents the expected energy consumption of CPU cycles required to perform a computational task. The energy consumed by the device for performing local calculations in time slot t is given by $W_{UC}(t) = \sum_{i=1}^{n} W_{U,i}(t)$.

2) Transmission Consumption: The achievable transmission speed is given above, so the communication energy consumption of device i is as follows:

$$W_{T,i}(t) = P_i(t)\frac{b_{M,i}(t)}{r_i(t)}. \tag{9}$$

At time slot t, the total communication energy consumption of all IoT devices to offload tasks to the MEC server is $W_{UT}(t) = \sum_{i=1}^{n} W_{T,i}(t)$.

3) MEC Side: The MEC server has relatively sufficient computing resources and can process tasks offloaded from different devices in parallel. Then, the energy consumption of the MEC server at time slot t is as follows:

$$W_M(t) = \gamma + \beta \sum_{i=1}^{n} b_{M,i}(t) \tag{10}$$

where the amount of energy consumed by the MEC server in standby is γ, independent of the workload. β is the power factor required for the MEC server to calculate the 1 bit task. C represents the maximum computing resources owned by the MEC server, then

$$\sum_{i=1}^{n} b_{M,i}(t) \le C. \tag{11}$$

Therefore, based on the above calculation, the total energy consumed by the system in time slot t is as follows:

$$W(t) = W_{UC}(t) + W_{UT}(t) + W_M(t). \tag{12}$$

D. Energy Consumption Optimization Problem

In accordance with the definition of the above parameters, we discuss the issue of minimizing energy consumption in the context of long-term system stability in this section. The problem is formalized as follows:

$$P1: \min_{f_i(t), b_{M,i}(t)} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E\{W(t)\} \tag{13}$$

$$\text{s.t.} \quad \bar{Q} \triangleq \lim_{T\to\infty} sup \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{n} E\{Q_i(t)\} < \infty \tag{14}$$

$$f_i(t) \le f_{\max} \quad \forall i \in N \tag{15}$$

$$P_i(t) \le P_{\max} \quad \forall i \in N$$
$$(2),\ (4),\ \text{and}\ (11). \tag{16}$$

Constraint (14) is a stability constraint, which means that the total length of the IoT device queue cannot grow infinitely for a long time, with a bounded time-averaged backlog. In this way, the system is in a long-term stable state. In addition, constraint (15) indicates that the upper limit of the IoT device's CPU calculation rate in each time slot is $f_{\max}$. Constraint (16) indicates that $P_{\max}$ is the peak value that the transmit power of the IoT device can reach.

Problem $P1$ is a typical optimization problem with randomness. The generation of computing tasks and the state of the wireless channel in the system change over time. Since it is difficult to obtain information about the system's future, in reality, solving problem $P1$ offline is a complex problem.

In the next section, we use the Lyapunov optimization framework to figure out a solution without knowing future information.

IV. Lyapunov Online Energy Consumption Optimization Algorithm

This section designs a new online offloading algorithm called LOECOA using the Lyapunov optimization framework. LOECOA makes the optimal offloading decision based on the current system status information without relying on future information. It fully considers the tradeoff between system energy consumed and queue backlog, and it achieves infinitely close to optimal energy consumption under the premise of system stability.

Lyapunov optimization refers to the establishment of a queue model for the constraints of stochastic optimization problems. We decompose the operations that should be taken to meet these long-term constraints into each time slice, thus decoupling the problem from the time perspective. Additionally, the size of the queue is considered when solving the objective function. This allows the system to maintain a relatively stable state to achieve our optimization goals.

A. Lyapunov Optimization Framework

To indicate the current state of the system and the degree of congestion, we define a queue backlog vector $\Theta(t)$, i.e., $\Theta(t) = (Q_1(t), Q_2(t), \ldots, Q_n(t))$. For vector $\Theta(t)$, we define the quadratic Lyapunov function as follows:

$$L(\Theta(t)) = \frac{1}{2} \sum_{i=1}^{n} \omega_i Q_i^2(t) \tag{17}$$

where $\{\omega_i\}_{i=1}^{n}$ is the weight of each queue, and in this article, the value set to 1. Clearly, the Lyapunov function is nonnegative, and $L(\Theta(t))$ is zero only if the backlog of all queues $Q_i(t)$ is zero. The size of $L(\Theta(t))$ denotes the relative size of the backlog of IoT device queues, and we define $L(\Theta(0)) = 0$. Then, we define the Lyapunov drift function as follows:

$$\Delta(\Theta(t)) \triangleq E\{L(\Theta(t+1)) - L(\Theta(t)) | \Theta(t)\}. \tag{18}$$

This means that the mathematical expectation of the Lyapunov function changes when the current state of slot t is $\Theta(t)$, which is the key index of system stability.

Lemma 1: For the Lyapunov function $L(\Theta(t))$, when $E\{L(\Theta(t))\} < \infty$, there is a constant

$B = (1/2)\sum_{i=1}^{n}(a_{\max}^2 + b_{\max}^2)$. ε can be regarded as a positive number slightly greater than zero. The upper bound of Lyapunov drift can be expressed as follows:

$$\Delta(\Theta(t)) \triangleq E\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\} \leq B - \varepsilon\sum_{i=1}^{n} Q_i(t). \tag{19}$$

Proof: According to (5):

1) if $Q_i(t)-b_i(t)+a_i(t) \geq 0$, it can obtain $Q_i(t+1) = Q_i(t)-b_i(t)+a_i(t)$, thus $Q_i(t+1)^2 = (Q_i(t)-a_i(t)+b_i(t))^2$, $i \in N$;

2) if $Q_i(t)-b_i(t)+a_i(t) < 0$, it can obtain $Q_i(t+1) = 0 > Q_i(t)-b_i(t)+a_i(t)$, thus $Q_i(t+1)^2 = 0 < (Q_i(t)-a_i(t)+b_i(t))^2$, $i \in N$.

In summary

$$Q_i(t+1)^2 \leq (Q_i(t)-b_i(t)+a_i(t))^2, i \in N. \tag{20}$$

Expanding inequality (20) and accumulating all n queues based on it, we obtain

$$\frac{1}{2}\sum_{i=1}^{n} Q_i(t+1)^2 \leq \frac{1}{2}\sum_{i=1}^{n} Q_i(t)^2 + \frac{1}{2}\sum_{i=1}^{n}(a_i(t)-b_i(t))^2 + \sum_{i=1}^{n} Q_i(t)(a_i(t)-b_i(t)).$$

Shifting the term of the above inequality and performing the expectation operation on both sides simultaneously, we obtain

$$\begin{aligned}\Delta(\Theta(t)) &= \frac{1}{2}\sum_{i=1}^{n} E\left\{Q_i(t+1)^2|\Theta(t)\right\} - \frac{1}{2}\sum_{i=1}^{n} E\left\{Q_i(t)^2|\Theta(t)\right\} \\ &\leq \frac{1}{2}\sum_{i=1}^{n} E\left\{(a_i(t)-b_i(t))^2|\Theta(t)\right\} \\ &\quad + \sum_{i=1}^{n} E\{Q_i(t)(a_i(t)-b_i(t))|\Theta(t)\} \\ &\leq B + \sum_{i=1}^{n} E\{Q_i(t)(a_i(t)-b_i(t))|\Theta(t)\}\end{aligned} \tag{21}$$

where B is a constant and $B = (1/2)\sum_{i=1}^{n}(a_{\max}^2 + b_{\max}^2)$. Since for any $i \in N$, $a_i(t) \leq a_{\max}$ and $b_i(t) \leq b_{\max}$, where $a_{\max}$ and $b_{\max}$ are used to represent the maximum value of $a_i(t)$ and $b_i(t)$. After a simple inequality calculation

$$\frac{1}{2}\sum_{i=1}^{n} E\left\{(a_i(t)-b_i(t))^2|\Theta(t)\right\} \leq \frac{1}{2}\sum_{i=1}^{n}\left(a_{\max}^2 + b_{\max}^2\right).$$

Recalling the premise (6) mentioned above and substituting it into (21), we obtain the formula. ■

According to Lemma 1, taking the telescoping sum on time slice $t \in \{0, 1, \ldots, T-1\}$, we have

$$E\{L(\Theta(T))\} - E\{L(\Theta(0))\} \leq BT - \varepsilon\sum_{t=0}^{T-1}\sum_{i=1}^{n} E\{Q_i(t)\}.$$

Bringing $L(\Theta(0)) = 0$ into it, and through simple inequality operations, we can obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{n} E\{Q_i(t)\} < \frac{B}{\varepsilon}. \tag{22}$$

This shows that the backlog of all IoT device task queues has an upper limit and cannot grow indefinitely. Therefore, the stability of the system is guaranteed.

Thus, optimization problem $P1$ is equivalently transformed into $P2$

$$P2: \min_{f_i(t), b_{M,i}(t)} \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\{W(t)\} \quad \text{s.t.} \quad (2),\ (4),\ (6),\ (11),\ (15),\ \text{and}\ (16). \tag{23}$$

In the Lyapunov optimization framework, we have introduced the concept of penalty terms to achieve the equilibrium of dual-objective optimization on the basis of system stability. Therefore, problem $P2$ is further transformed into the following form:

$$P3: \min_{f_i(t), b_{M,i}(t)} \Delta(\Theta(t)) + VE\{W(t)|\Theta(t)\} \quad \text{s.t.} \quad (2),\ (4),\ (6),\ (11),\ (15),\ \text{and}\ (16). \tag{24}$$

The first item reduces the queue backlog and keeps the system stable, and the second item is our optimization goal. The Lyapunov control parameter V is a weight coefficient used to weigh the importance of these two goals. The theoretical basis for this is given in the strict proof as follows.

Additionally, $P3$ is still a difficult issue. It needs information at the present and the future, so we decompose the optimization into each time slot to achieve the purpose of optimization by solving the minimum value of $\Delta(\Theta(t)) + VE\{W(t)|\Theta(t)\}$.

Next, we simplify problem $P3$ by minimizing the upper bound of Lyapunov drift plus a penalty. Based on (21), we have

$$\begin{aligned}&\Delta(\Theta(t)) + VE\{W(t)|\Theta(t)\} \\ &= E\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\} + VE\{W(t)|\Theta(t)\} \\ &\leq B + VE\{W(t)|\Theta(t)\} + \sum_{i=1}^{n} E\{Q_i(t)(a_i(t)-b_i(t))|\Theta(t)\}.\end{aligned} \tag{25}$$

By means of the iterated expectation theorem, we can equivalently transform problem $P3$ into $P4$. Since B is a constant

$$\begin{aligned}P4: \min_{f_i(t), b_{M,i}(t)} \quad &\left[VW(t) + \sum_{i=1}^{n} Q_i(t)(a_i(t)-b_i(t))\right] \\ &b_{M,i}(t) \leq r_i(t)\tau \quad \forall i \in N \\ &b_{M,i}(t) \leq Q_i(t) - b_{U,i}(t) \quad \forall i \in N \\ &\sum_{i=1}^{n} b_{M,i}(t) \leq C \\ &f_i(t) \leq f_{\max} \quad \forall i \in N \\ &P_i(t) \leq P_{\max} \quad \forall i \in N.\end{aligned} \tag{26}$$

Now, we need to use only the current information of the system. Solving for the minimum value of (26) in each time slot, we obtain the desired offloading decision.

Algorithm 1 Online Calculation Offloading Algorithm

```
for t = 1 to T do
    for all i ∈ N do
        Calculate f_i(t), b_{M,i}(t) by solving P4;
        Update Q_i(t) according to
        Q_i(t+1) = max{Q_i(t) − b_i(t) + a_i(t), 0};
    end for
end for
```

B. Optimal Offloading Policy

We need to further solve problem $P4$ from Algorithm 1. Thus, we design an optimization algorithm, LOECOA. The upper limit of the Lyapunov drift plus penalty term in each slot is minimized to determine the optimal task offloading strategy. Experiments indicate that LOECOA can effectively decrease system energy consumption and maintain a low IoT device queue backlog.

To better solve problem $P4$, we expand (26) to obtain

$$\begin{aligned}
&VW(t) + \sum_{i=1}^{n} Q_i(t)(a_i(t) - b_i(t)) \\
&= V(W_{UC}(t) + W_{UT}(t) + W_M(t)) \\
&\quad + \sum_{i=1}^{n} Q_i(t)\big(a_i(t) - b_{U,i}(t) - b_{M,i}(t)\big) \\
&= V\left(\kappa \sum_{i=1}^{n} f_i^3(t) + \sum_{i=1}^{n} P_i(t)\frac{b_{M,i}(t)}{r_i(t)} + \gamma + \beta \sum_{i=1}^{n} b_{M,i}(t)\right) \\
&\quad + \sum_{i=1}^{n} Q_i(t)\big(a_i(t) - b_{U,i}(t) - b_{M,i}(t)\big) \\
&= \sum_{i\in N}\Bigg[\kappa f_i^3(t)V - Q_i(t)\frac{f_i(t)\tau}{L} \\
&\qquad + \left(\frac{VP_i(t)}{r_i(t)} + \beta V - Q_i(t)\right)b_{M,i}(t)\Bigg]. \qquad (27)
\end{aligned}$$

Through (27), we find that the optimization problem can be divided into two different smaller problems and solved, as follows.

1) Local Calculation: By separating the variables, the part related to the local computation can be expressed as follows:

$$\min_{f_i(t)} \left(\kappa f_i^3(t)V - Q_i(t)\frac{f_i(t)\tau}{L}\right). \qquad (28)$$

It is not difficult to find that the objective function is a convex function, so the optimal value should be obtained at the extreme points or boundaries, i.e.

$$f_i^*(t) = \min\left\{\sqrt{\frac{Q_i(t)\tau}{3\kappa VL}},\ f_{\max}\right\}. \qquad (29)$$

After we solve the optimal local CPU calculation rate $f_i^*(t)$, by substituting it into (7), we can calculate the local calculation amount $b_{U,i}(t)$ of each IoT device.

2) Offloading Calculation: After decoupling $b_{M,i}(t)$ from (27), the optimal value of $b_{M,i}(t)$ can be obtained by solving

$$\min_{b_{M,i}(t)} \left(\frac{VP_i(t)}{r_i(t)} + \beta V - Q_i(t)\right)b_{M,i}(t). \qquad (30)$$

To make it easier to solve, we can transform the problem into maximizing the opposite number

$$\max_{b_{M,i}(t)} H_i(t)b_{M,i}(t) \qquad (31)$$

where

$$H_i(t) = Q_i(t) - \frac{VP_i(t)}{r_i(t)} - \beta V. \qquad (32)$$

Equation (31) can be considered in terms of a knapsack problem. The maximum computing resources owned by the MEC server C represents the backpack capacity, and weight coefficient $H_i(t)$ can be regarded as the item's unit value. The optimal solution is to sequentially pick the highest value nonnegative items into the backpack until the backpack is full or the nonnegative items are all put out.

The specific steps are as follows.

1) Initialize the computing task size $b_{M,i}(t)$ of each IoT device offloading to the edge side, and the size $D(t)$ of all available computing resources of the MEC server is C.
2) Collect information, such as the current task offloading queue backlog $Q_i(t)$, wireless channel transmit power $P_i(t)$, and rate $r_i(t)$ of each IoT device.
3) Sort all IoT devices according to the principle of weighting coefficient $H_i(t)$ from high to low.
4) Determine the size of the offloading tasks assigned to the MEC server calculation for each device in turn, starting from the first in the sorted order

$$b_{M,i}^*(t) = \begin{cases} \min\{\eta_i(t), D(t)\}, & \text{if } H_i(t) \geq 0 \\ 0. & \text{if } H_i(t) < 0. \end{cases} \qquad (33)$$

According to the constraints (2) and (4), $H_i(t)$ is denoted by $\eta_i(t) = \min\{r_i(t)\tau, Q_i(t) - b_{U,i}(t)\}$.

5) Update the size of the remaining available computing resources based on $D(t) = D(t) - b_{M,i}(t)$.

At this point, we obtain the approximate solution of problem $P4$. The algorithm pseudocode is shown as Algorithm 2.

V. Algorithm Performance Theoretical Analysis

In this section, we conduct a rigorous theoretical analysis to verify the performance of LOECOA. After mathematical derivation, we find that the algorithm can ensure the stability of the IoT device virtual queue while making the system energy consumption approach the optimal solution.

We adopt the difference between the Lyapunov optimization and thalytical solution of the problem $P1$ by analyzing it from the solution of the objective function and the stability of the task offloading queue, respectively.

Theorem 1: By adopting the LOECOA, the average system energy consumption satisfies

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E\{W(t)|\Theta(t)\} \leq W^* + \frac{B}{V} \qquad (34)$$

Algorithm 2 LOECOA

Input: device and channel parameters such as $a_i(t)$, $P_i(t)$, $h_i(t)$; queue state such as $Q_i(0) \leftarrow 0$;
Output: optimal solution $f_i(t), b_{M,i}(t)$;
1: **for all** $i \in N$ **do**
2: Calculate the value of $\sqrt{\frac{Q_i(t)\tau}{3\kappa VL}}$;
3: Set the $f_i(t)$ according to (7);
4: Calculate the $b_{U,i}(t)$;
5: **end for**
6: Initialize $\forall i \in N$ $b_{M,i} = 0, D(t) = C$;
7: **for all** $i \in N$ **do**
8: Obtain the $P_i(t), r_i(t)$;
9: Calculate the $H_i(t)$;
10: **end for**
11: Sort all the devices according to the principle of weighting coefficient $H_i(t)$ from high to low;
12: **for all** $i \in N$ **do**
13: Solve the offloading decision $b_{M,i}(t)$ in sorted order according to (33);
14: Update $D(t)$ according to $D(t) = D(t) - b_{M,i}(t)$;
15: **end for**

and the average queue backlog satisfies

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{n} E\{Q_i(t)|\Theta(t)\} \le \frac{B + V(W_{\max} - W^*)}{\varepsilon} \quad (35)$$

where W^* and $W_{\max}$ are the optimal system energy consumption and the maximum system energy consumption, respectively. B and ε are the same as defined in (19), and V is the Lyapunov control parameter.

Proof: For problem $P1$, it defines all random information, such as $a_i(t)$ and $h_i(t)$ as $\omega(t)$, and all decision factors are defined as $\alpha(t)$ [29], [32]. For any random information set $\omega(t)$, there is a stable decision set $\alpha^*(t)$, which enables the mobile device to make a reasonable offloading decision in every time slot. The energy consumed under the optimal decision is W^*, which is

$$W\big(\alpha^*(t)|\omega(t)\big) = W^*.$$

According to the above theory, (25) can be rewritten as follows:

$$\begin{aligned}
&\Delta(\Theta(t)) + VW(t) \\
&\le B + VW(\omega(t), \alpha(t)) \\
&\quad + \sum_{i=1}^{n} Q_i(t)(a_i(\omega(t), \alpha(t)) - b_i(\omega(t), \alpha(t))) \\
&\le B + VW\big(\omega(t), \alpha^*(t)\big) \\
&\quad + \sum_{i=1}^{n} Q_i(t)\big(a_i\big(\omega(t), \alpha^*(t)\big) - b_i\big(\omega(t), \alpha^*(t)\big)\big).
\end{aligned}$$

If we take the expectation of both sides of this inequality, we obtain

$$\begin{aligned}
&E\{\Delta(\Theta(t)) + VW(t)|\Theta(t)\} \\
&\le E\Bigg\{B + VW\big(\omega(t), \alpha^*(t)\big) \\
&\qquad + \sum_{i=1}^{n} Q_i(t)\big(a_i\big(\omega(t), \alpha^*(t)\big) - b_i\big(\omega(t), \alpha^*(t)\big)\big)|\Theta(t)\Bigg\} \\
&= B + VE\big\{W\big(\omega(t), \alpha^*(t)\big)\big\} + \sum_{i=1}^{n} E\{Q_i(t)|\Theta(t)\} \\
&\quad \times \sum_{i=1}^{n} E\big\{a_i\big(\omega(t), \alpha^*(t)\big) - b_i\big(\omega(t), \alpha^*(t)\big)\big\} \\
&\le B + VW^*.
\end{aligned}$$

We take the telescoping sum on time slice $t \in \{0, 1, \ldots, T-1\}$. Thus, we have

$$L(\Theta(T)) - L(\Theta(0)) + V\sum_{t=0}^{T-1} E\{W(t)|\Theta(t)\} \le BT + VTW^*$$

where $L(\Theta(T))$ is greater than zero. If we divide both sides of this inequality by VT, we obtain

$$\frac{1}{T}\sum_{t=0}^{T-1} E\{W(t)|\Theta(t)\} \le W^* + \frac{B}{V}.$$

There is always a set of decision sets $\alpha'(t)$ and a constant ε, such that the following holds:

$$\sum_{i=1}^{n} E\big\{a_i\big(\omega(t), \alpha'(t)\big) - b_i\big(\omega(t), \alpha'(t)\big)\big\} \le -\varepsilon.$$

Substituting the above formula into (25), it can be obtained that

$$\begin{aligned}
\Delta(\Theta(t)) + VW(t) &\le B + VW\big(\omega(t), \alpha'(t)\big) \\
&\quad + \sum_{i=1}^{n} Q_i(t)\big(a_i\big(\omega(t), \alpha'(t)\big) - b_i\big(\omega(t), \alpha'(t)\big)\big).
\end{aligned}$$

We scale the $W(t)$ on both sides of the formula, taking $W(t)$ to the left side as W^* and taking $W(t)$ to the right side as $W_{\max}$. Therefore

$$\begin{aligned}
&\Delta(\Theta(t)) + VW^* \\
&\le B + VW_{\max} + \sum_{i=1}^{n} Q_i(t)\big(a_i\big(\omega(t), \alpha'(t)\big) - b_i\big(\omega(t), \alpha'(t)\big)\big).
\end{aligned}$$

At this point, we take the mathematical expectation on both sides of the formula and scale them to obtain

$$E\{\Delta(\Theta(t))\} + VW^* \le B + VW_{\max} - \varepsilon\sum_{i=1}^{n} E\{Q_i(t)|\Theta(t)\}.$$

After performing simple operations on both sides of the inequality

$$E\{\Delta(\Theta(t))\} \le B + V\big(W_{\max} - W^*\big) - \varepsilon\sum_{i=1}^{n} E\{Q_i(t)|\Theta(t)\}.$$

Similar to the above proof, we take the telescoping sum on time slice $t \in \{0, 1, \ldots, T-1\}$ and obtain

$$E\{L(\Theta(T))\} - E\{L(\Theta(0))\}$$

$$\leq BT + V\big(W_{\max} - W^*\big)T - \varepsilon \sum_{t=0}^{T-1} \sum_{i=1}^{n} E\{Q_i(t)|\Theta(t)\}.$$

If we divide both sides of the inequality by εT, we have

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{n} E\{Q_i(t)|\Theta(t)\} \leq \frac{B + V(W_{\max} - W^*)}{\varepsilon} - \frac{E\{L(\Theta(T))\}}{\varepsilon T}$$

where $L(\Theta(T))$ is a nonnegative number. Thus, scaling the inequality again, we obtain

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{n} E\{Q_i(t)|\Theta(t)\} \leq \frac{B + V(W_{\max} - W^*)}{\varepsilon}.$$

The proof of the theorem is thus complete. ■

The theorem demonstrates that the upper limit of the average system energy consumed is inversely proportional to the weight coefficient V; in contrast, the upper limit of the average queue backlog is directly proportional to V. Thus, to maintain the queue load's stability, a smaller V should be selected. However, when parameter V is reduced, the system energy consumed increases. Therefore, the system energy consumed and stability have a tradeoff relationship of $[O(1/V), O(V)]$. When conducting simulation experiments, we can choose an appropriate V to preserve the system's stability under the given energy budget; the reverse is also the same.

Finally, we analyze the performance of LOECOA in terms of time complexity and cost efficiency. On the one hand, according to Algorithm 1, LOECOA must ensure that each IoT device's requests are processed within a unit time slot. The time complexity of the outer loop is $O(n)$. In Algorithm 2, the offloading priority of all devices must be sorted before assignment. The algorithm we use is quicksort, and the time complexity is $O(\log n)$. Thus, the total time complexity of the LOECOA is $O(n \log n)$. On the other hand, the space complexity of our proposed algorithm is relatively low. When the program runs on the computer, the memory changes of the system are very small. This is because the operation of the algorithm does not require a large amount of storage space and only processes the collected data, which is simple and efficient. When running the algorithm, the CPU calculation speed changes from 3.21 to 4.06 GHz. The change is also acceptable.

VI. Simulation Results and Analysis

We evaluate the proposed LOECOA in terms of the average queue backlog and total system energy consumed. The theoretical analysis is verified by the results of a MATLAB simulation experiment.

First, we verify the theoretical results of the task offloading queue stability. Then, we analyze the influence of parameters, such as the Lyapunov weight coefficient V, computing task arrival rate $a_i(t)$, and the number of IoT devices n on the algorithm. Finally, it can be concluded that LOECOA has superiority in terms of energy consumption under the premise of ensuring system stability.

In the experiment, we first simulated a scenario with 30 IoT devices and one MEC server to perform offloading task calculations. We simulated 1500 time slots, where the length of each time slot was $\tau = 1$ ms. We use some specific statistical distributions to simulate events in the real environment, such as the arrival rate of offloading tasks, channel status, and device transmission power. The data size of the offloading task arrival is set to the expected Poisson distribution of 1000 bits, i.e., $a_i(t) \sim P(1000)$. For the wireless channel, we simplify it as a Rayleigh fading model, and $h_i(t)$ can be regarded as a random variable with a mathematical expectation of 1 that obeys an exponential distribution, i.e., $h_i(t) \sim E(1)$. We set the transmission power of each IoT device as $P_i(t) \sim U[0.01, 1]$ W. In addition, we set $\kappa = 1 \times 10^{-27}$, $\sigma^2 = 1 \times 10^{-12}$ W, $P_{\max} = 1$ W, $f_{\max} = 1.5$ GHz, $L = 737.5$ cycles/byte [33]. Although some parameters are set according to a fixed statistical distribution in the experiment, LOECOA actually does not require prestatistical information.

A. System Stability Evaluation

We randomly select four of the 30 IoT devices to verify if their queue backlog changes are stable. Fig. 2(a) shows the change in the backlog of task offloading queues for four randomly selected IoT devices as the simulation time increases. The analysis shows that the backlog of IoT device queues increases rapidly over time and eventually stabilizes. At the beginning of the simulation, the number of calculation tasks is small, and most tasks are calculated locally; as the number of calculation tasks increases, the queue backlog increases rapidly. The system begins to offload tasks to the edge side for processing. Due to the significant improvement of edge processor performance, tasks are processed quickly, which gradually stabilizes the backlog of queues. The time required for the backlog of the IoT device queue to stabilize is relatively short, reflecting LOECOA's good performance.

Fig. 2(b) reveals the growth of the average queue backlog of all IoT devices, which shows a gradual rise and then plateaus. This is because in this system the change trend of the queue backlog of a single device is roughly the same and the stability is good, which shows that the offloading system is stable for a long time.

Next, we investigated whether the system queue backlog can remain stable when the task arrival rate changes dynamically. We randomly select four IoT devices and observe their queue backlog changes when the arrival rate of computing tasks changes.

In Fig. 3, the arrival rate obeys a Poisson distribution with a mean value of 1500 at the beginning. At the 200th step, we reduced the task arrival rate by 100, i.e., the arrival rate obeyed a Poisson distribution with a mean value of 1400. As shown in Fig. 3, the queue backlog dropped significantly. At the 400th step, the arrival rate was increased by 100. The queue backlog quickly returned to its original level. On the basis of this experiment, we can conclude that LOECOA has a strong adaptive ability to handle real application scenarios with sudden changes in network traffic.

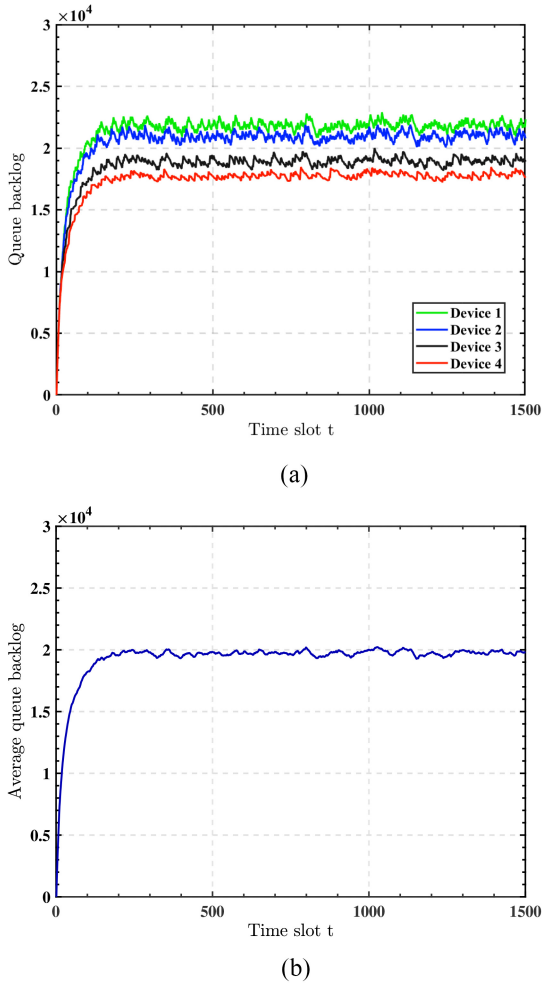


Fig. 2. Queue backlog stability experiment. (a) Queue backlog. (b) Average queue backlog.

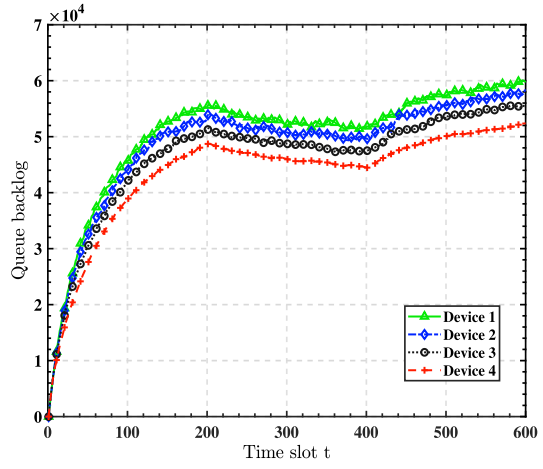


Fig. 3. Queue backlog with dynamic arrival rates.

B. Effects of System Parameters

Next, to verify the robustness of LOECOA, we select several sets of key parameters for ablation experiments. In each set of experiments, the effects of changes in a single parameter on two performance indicators are observed, and the adaptability of LOECOA to changes in different parameters is explored.

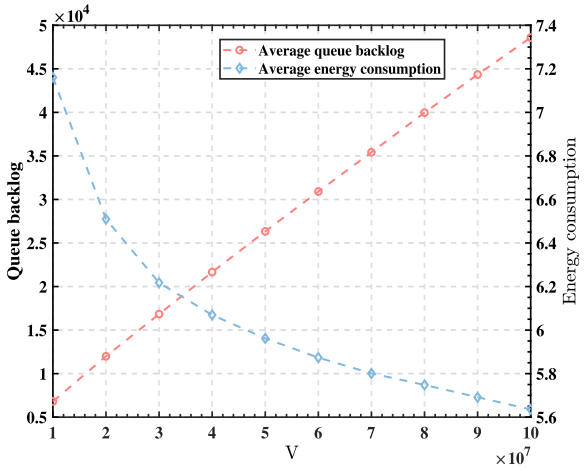


Fig. 4. Effect of V on the system energy consumption and queue backlog.

1) Effect of Parameter V: We pick several different sets of V values. The average queue backlog and energy consumption for the corresponding systems at different values of V are calculated, and the results are plotted in the same figure. The changes in system energy consumed and queue backlog when selecting different V values are shown in Fig. 4. When the value of V increases, the system energy consumed is significantly reduced. This is consistent with the law embodied in (34). However, the queue backlog grows substantially when V increases. This phenomenon conforms to the theoretical derivation in (35). It adversely affects the stability of the system and considerably reduces the QoE of users. Therefore, we demonstrate again through experiments that there is an $[O(1/V), O(V)]$ tradeoff between the system energy consumption and the queue backlog. By regulating the Lyapunov weight coefficient V, the importance of both is changed.

2) Impact of Parameter a(t): Fig. 5 depicts the changes in the system energy consumed and queue backlog under different task arrival rates. These experiments set three different $a(t)$ values, 800, 1000, and 1200. In Fig. 5(a), the system energy consumed increases as the task arrival rate increases. There are more tasks to calculate, so more energy is consumed for task processing. Similarly, Fig. 5(b) shows that the queue backlog increases as the task arrival rate increases. The reason for this phenomenon is that when the number of tasks increases, the system's transmission capacity does not increase, so the task queue backlog of IoT devices increases. In addition, the system consumed energy and the queue backlog of the algorithm converge quickly under different task arrival rates. Therefore, LOECOA can adjust the offloading decision online to adapt to changes in the arrival rate so that the system quickly reaches stability.

Next, we explore the effect of the parameter $a(t)$ on the performance of LOECOA from two aspects: 1) the proportion of offloaded computing tasks and 2) the CPU computing rates of IoT devices. By analyzing the data in Table II, we can conclude that the more tasks there are to be processed, the lower

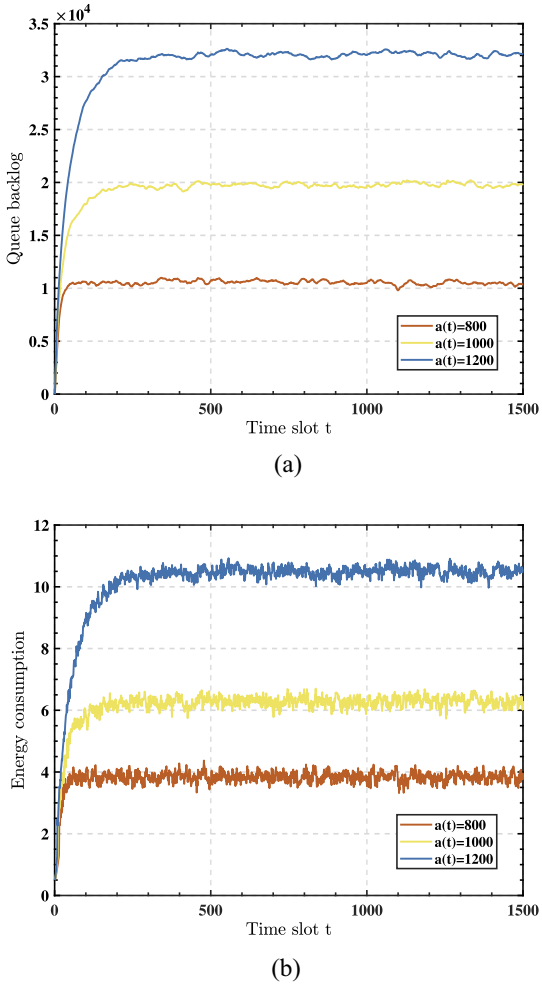


Fig. 5. Queue backlog and system energy consumption with different task arrival rates. (a) Queue backlog. (b) System energy consumption.

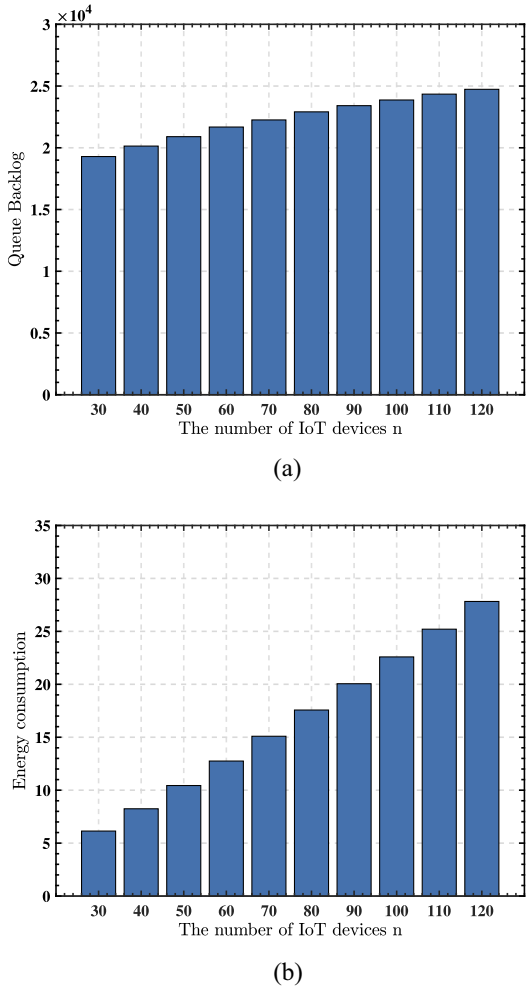


Fig. 6. Queue backlog and system energy consumption with different numbers of IoT devices. (a) Queue backlog. (b) System energy consumption.

TABLE II
AVERAGE OFFLOADING PORTION AND AVERAGE CPU CALCULATION RATE OF IOT DEVICES

Arrival rate	poissrnd(800)	poissrnd(1000)	poissrnd(1200)
Offloading portion	30.4%	26.9%	23.9%
CPU calculation rate	3.66×10^8	4.98×10^8	6.32×10^8

the proportion of data offloaded to the MEC server for calculation. Since the computing resources of the MEC server are limited, when the task arrival rate increases, a large number of computing requests occurring at the edge node at the same time will cause excessive queuing delay. This can lead to system instability, so offloading decisions choose to put more workload on the local side. Similarly, when the amount of data arriving increases, the transmission capacity of the system does not increase. As the data to be processed increase, the backlog of equipment queues increases and the requirements for local computing power increase. To maintain stable system operation, the calculation rate of IoT devices also increases accordingly.

3) Impact of Parameter n: To explore the stability of the system and the robustness of the algorithm in the scenario where the number of devices increases, we increase the number of devices from 30 to 120 in sequence, with an increase of 10 each time. Fig. 6 plots the system consumed energy and queue backlog under different quantities of equipment.

Fig. 6(a) shows that the queue backlog also slowly increases. The reason is similar to the phenomenon in Fig. 5 above. As the amount of equipment increases, so does the number of computing tasks to be processed. The computing power of the edge side and the transmission capacity of the system are limited. To keep the system stable, the computing rate of IoT devices increases. This process can effectively relieve the pressure and realize the slow growth of the queue backlog, reflecting the excellent performance of LOECOA. Fig. 6(b) reveals that as the number of IoT devices grows, the energy consumed by the system also increases.

C. Comparison Experiment

To further evaluate the performance of LOECOA, we compare LOECOA with two baseline algorithms.

1) *Local Only:* In slot t, computing tasks are executed on IoT devices at the maximum CPU rate.

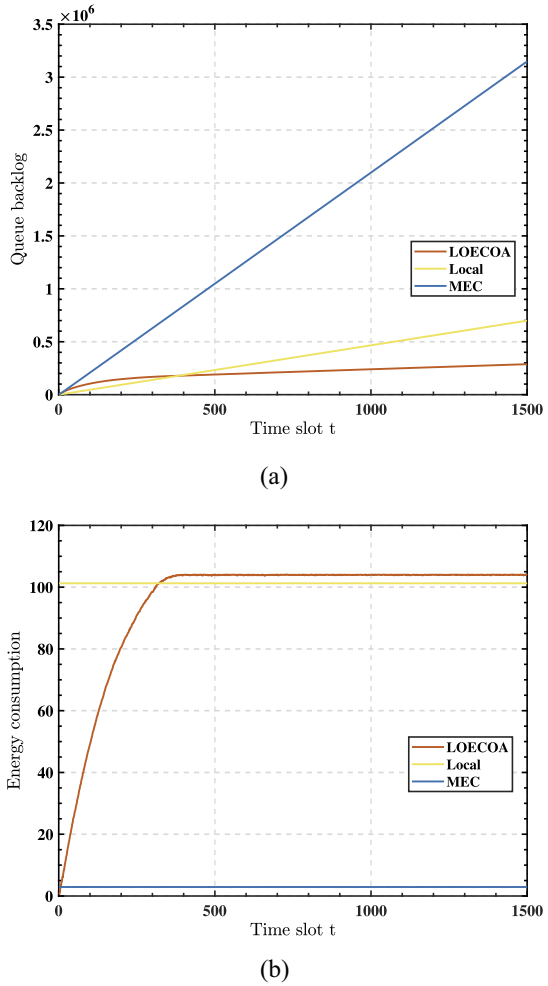


Fig. 7. Queue backlog and system energy consumption with three different strategies. (a) Queue backlog. (b) System energy consumption.

2) *MEC Only:* In slot t, the generated computing tasks are transmitted to the MEC server with the maximum transmission power and then executed in the MEC server with the maximum computing power.

In Fig. 7, we have drawn the change curve of the system energy consumption and queue backlog for different strategies. The LOECOA proposed in this article can effectively reduce the queue backlog and maintain the stability of the system at roughly the same level of energy consumption. This is because LOECOA can dynamically make offloading decisions to adapt to changes in system status. In addition, the energy consumption of our algorithm is similar to that of only local computing. However, what is different from the previous work is that our system energy consumption includes the noncomputing part of the MEC server. Only local calculations do not involve this part, so the system energy is higher than local calculations only. From the comparison of the two figures, only part of the calculation tasks is processed due to the limited transmission capacity of the MEC server calculation. Therefore, the energy consumption is relatively low, and the price of a large queue backlog is paid. It can be concluded from the comparative experiment that LOECOA has advantages in effectively reducing system energy consumption and queue backlog.

VII. Conclusion

In this article, we investigate minimizing energy costs in MEC systems in cooperation scenarios. By solving the problem with Lyapunov optimization technology, we propose an online dynamic computing offloading algorithm that can reasonably formulate an offloading strategy and determine the number of calculation tasks for the local and MEC servers. This algorithm rarely requires prior statistical knowledge of the environment, and it can achieve the purpose of balancing the dual optimization goals. The experimental results show that LOECOA can obtain near-optimal system energy consumed while maintaining a stable queue backlog of IoT devices. Moreover, the algorithm shows good robustness in the face of dynamic changes.

References

[1] 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.

[2] M. Mittal, C. Iwendi, S. Khan, and A. R. Javed, "Analysis of security and energy efficiency for shortest route discovery in low-energy adaptive clustering hierarchy protocol using levenberg-marquardt neural network and gated recurrent unit for intrusion detection system," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, p. e3997, 2021.

[3] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2017, pp. 1–7.

[4] 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 Services (ICWS)*, 2019, pp. 193–200.

[5] Y. Nan *et al.*, "Adaptive energy-aware computation offloading for cloud of things systems," *IEEE Access*, vol. 5, pp. 23947–23957, 2017.

[6] 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.

[7] S. A. Latif *et al.*, "Ai-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems," *Comput. Commun.*, vol. 181, pp. 274–283, Jan. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366421003662

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

[9] X. Renchao, L. Xiaofei, J. Qingmin, and Y. L. T. Huang, "Survey on computation offloading in mobile edge computing," *J. Commun.*, vol. 39, no. 11, pp. 138–155, 2018.

[10] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. 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.

[11] P. Shu *et al.*, "eTime: Energy-efficient transmission between cloud and mobile devices," in *Proc. IEEE INFOCOM*, 2013, pp. 195–199.

[12] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[13] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "An online algorithm for task offloading in heterogeneous mobile clouds," *ACM Trans. Internet Technol.*, vol. 18, no. 2, pp. 1–25, 2018.

[14] R. Lin *et al.*, "Distributed optimization for computation offloading in edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 8179–8194, Dec. 2020.

[15] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

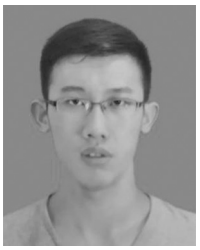
[16] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[17] 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.
[18] N. Zhang, S. Guo, Y. Dong, and D. Liu, "Joint task offloading and data caching in mobile edge computing networks," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107446.
[19] L. Lv *et al.*, "Contract and lyapunov optimization based load scheduling and energy management for UAV charging stations," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1381–1394, Sep. 2021.
[20] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.
[21] 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.
[22] Z. Tong, F. Ye, B. Liu, J. Cai, and J. Mei, "DDQN-TS: A novel bi-objective intelligent scheduling algorithm in the cloud environment," *Neurocomputing*, vol. 455, pp. 419–430, Sep. 2021.
[23] S. Liao, W. Wu, J. Jiao, S. Wu, and Q. Zhang, "Fairness-improved resource allocation for qos-guaranteed satellite-based Internet of Thing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2021, pp. 1–6.
[24] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep q-learning," *Inf. Sci.*, vol. 512, pp. 1170–1191, Feb. 2020.
[25] Y. Qin, W. Han, Y. Yang, W. Yang, and B. Liu, "Joint energy optimization on the server and network sides for geo-distributed data centers," *J. Supercomput.*, vol. 77, pp. 7757–7790, Jan. 2021.
[26] Z. Tong, X. Deng, H. Chen, and J. Mei, "DDMTS: A novel dynamic load balancing scheduling scheme under sla constraints in cloud computing," *J. Parallel Distrib. Comput.*, vol. 149, pp. 138–148, Mar. 2021.
[27] J. H. Anajemba, T. Yue, C. Iwendi, M. Alenezi, and M. Mittal, "Optimal cooperative offloading scheme for energy efficient multi-access edge computation," *IEEE Access*, vol. 8, pp. 53931–53941, 2020.
[28] H. Zhang, J. Guo, L. Yang, X. Li, and H. Ji, "Computation offloading considering fronthaul and backhaul in small-cell networks integrated with MEC," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2017, pp. 115–120.
[29] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Communication Networks*, vol. 3. Williston, VT, USA: Morgan & Claypool, 2010, pp. 1–211.
[30] Y. Mao, J. Zhang, S. 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.
[31] 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, Jan.-Mar. 2020.
[32] M. J. Neely and L. Huang, "Dynamic product assembly and inventory control for maximum profit," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, 2010, pp. 2805–2812.
[33] Y. Chen, S. Deng, H. Zhao, Q. He, Y. Li, and H. Gao, "Data-intensive application deployment at edge: A deep reinforcement learning approach," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2019, pp. 355–359.

Zhao Tong (Member, IEEE) received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2014.

He was a Visiting Scholar with Georgia State University, Atlanta, GA, USA, from 2017 to 2018. He is currently an Associate Professor with the College of Information Science and Engineering, Hunan Normal University, Changsha, the Young Backbone Teacher of Hunan Province, China. He has published more than 25 research papers in international conferences and journals, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Information Sciences*, *Future Generation Computer Systems*, *Neural Computing and Applications*, *Journal of Parallel and Distributed Computing*, and PDCAT. His research interests include parallel and distributed computing systems, resource management, big data, and machine learning algorithm.

Dr. Tong is a Senior Member of the China Computer Federation.



Jinhui Cai received the B.E. degree from Qilu University of Technology (Shandong Academy of Sciences), Jinan, China, in 2019. He is currently pursuing the M.S. degree with the College of Information Science and Engineering, Hunan Normal University, Changsha, China.

His research interests include cloud edge integration, task offloading, objective optimization, machine learning, and artificial intelligence.

Mr. Cai is a Student Member of the China Computer Federation.

Jing Mei received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2015.

She is currently a Lecturer with the College of Information Science and Engineering, Hunan Normal University, Changsha. She has published more than 15 research articles in international conferences and journals, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE-SC, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Cluster Computing*, *Journal of Gastric Cancer*, *The Journal of Supercomputing*, and ICPP. Her research interests include parallel and distributed computing, cloud computing, and combinatorial optimization.

Dr. Mei is a member of the China Computer Federation.

Kenli Li (Senior Member, IEEE) received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2003.

He was a Visiting Scholar with the University of Illinois at Urbana–Champaign, Champaign, IL, USA, from 2004 to 2005. He is currently a Cheung Kong Professor of Computer Science and Technology with Hunan University, Changsha, China, where he is the Assistant President, the Dean of the College of Information Science and Engineering, and the Director of National Supercomputing Center. He has published more than 350 research papers in international conferences and journals, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, HPCA, SC, MM, AAAI, DAC, and ICPP. His major research interests include parallel and distributed processing, high-performance computing, and big data management.

Dr. Li is currently serving or has served as an Associate Editor for IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He is a Fellow of the China Computer Federation.

Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor of Computer Science with the State University of New York, New Paltz, NY, USA. He has published over 510 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU–GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of Things, and cyber–physical systems.

Dr. Li is currently serving or has served on the editorial boards of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, and IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.