Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2279362, 10 pages
https://doi.org/10.1155/2022/2279362

Research Article

Offloading Cost Optimization in Multiserver Mobile Edge Computing Systems with Energy Harvesting Devices

Zheng Liu,[1] **Kun Jiang**,[1] **Xiuqiang Wu**,[2] **and Xianxiong Zeng**[3]

[1]*School of Computer and Science Engineering, Xi'an University of Technology, Xi'an 710048, China*
[2]*China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou, China*
[3]*Wuhan Zhongyuan Electronics Group Co., Ltd, Wuhan, China*

Correspondence should be addressed to Xiuqiang Wu; wuxiuqiang@ceprei.com

Received 8 July 2022; Revised 16 August 2022; Accepted 1 September 2022; Published 26 September 2022

Academic Editor: Lei Liu

In mobile edge computing (MEC) systems with energy harvesting, edge devices are powered by unstable energy harvested from the environment. To prolong the lifetime of edge devices, some computing tasks should be offloaded to MEC servers. However, computing services offered by MEC servers may be very costly. In this work, we aim to minimize total costs caused by computing services and dropping tasks while avoiding the devices running out energy. With the consideration of the unpredictability of the harvestable energy, we adopt the stochastic Lyapunov optimization framework to jointly manage energy and make task execution decisions (i.e., local executing, offloading, or dropping tasks) and develop an online algorithm which could help asymptotically obtain the optimal results for the whole system. The algorithm does not require any knowledge of the harvestable energy and the statistics of task arriving processes and can be easily implemented in a distributed manner. Numerical results corroborate that the proposed algorithm can try its best to push battery energy of edge devices to a preset parameter and effectively reduce the service costs and task drops.

1. Introduction

In mobile edge computing systems, mobile applications such as face recognition, image identification, and augmented reality are usually computational-intensive and consume a large quantity of energy of edge devices [1–3]. Harvesting energy from the environment provides a promising solution to address energy scarcity problem in MEC systems. However, in many circumstances, the harvestable energy is usually unpredictable and not sufficient to support edge devices executing all computing tasks locally. Some computing tasks need to be offloaded to edge servers.

Compared with edge devices, edge servers which are usually provided by service providers are more powerful and can assist edge devices to complete computing tasks nearby [4–6]. When edge devices offload their computing tasks to edge servers, there will be some service fee since it consumes bandwidth and computing resources of the servers [7, 8]. The service fee which is charged by service providers may be concerned to the types of tasks and can be time-varying. For example, the prices per bit of image identification may be different from augmented reality tasks, and when there are too many computing tasks being executed in a server, the service provider may raise the computation price of this server. For edge devices, more offloading means less energy consumption but more service costs. Hence, how to minimize service costs while guaranteeing edge devices never running out of energy will be a realistic problem.

Many offloading techniques [9–15] have been proposed to minimize task execution latency and energy consumption or achieve the trade-off between them [16, 17]. However, in terms of energy management, most of those works focus on saving the energy cost of edge computing systems which may cause excessive computation offloading. Since the battery energy of edge devices can be constantly replenished in energy harvesting MEC systems, making full use of harvestable energy for local computation can effectively reduce data transfer and service overhead. Moreover, they do not take the cost of computation service into account, and most

of them considered the problem from the perspective of static systems and single-user and single-server model.

In this paper, we consider a more general scenario with multiple energy harvesting mobile edge devices and multiple MEC servers. The harvestable energy of each edge devices is different and unpredictable. Time is slotted, and computation tasks are delay sensitive and with the execution deadline no greater than the length of a time slot. We aim to find a task offloading strategy which jointly manage edge device energy and minimize the total costs resulted by computation offloading (i.e., service costs) and task dropping. With the consideration of the energy profile as a stochastic process, we adopt the Lyapunov optimization framework with weight perturbation to achieve long-term stability of energy queues of edge devices. Based on the theory of drift-plus-penalty, an online algorithm which can obtain the trade-off between energy performance (i.e., edge devices' residual energy) and the total costs is developed. The algorithm makes greedy decisions in every time slot; hence, the complexity of the algorithm is low, and it can adapt to resource-constrained MEC systems very well. To summarize, the key contributions of this work are listed as follows:

(1) We consider both the heterogeneity and mobility of edge devices and model a MEC system with multiple energy harvesting edge devices and multiple MEC servers

(2) Instead of minimizing energy and delay costs, we define the total costs resulted by task offloading and dropping as the optimization object and formulate the computation offloading problem as a joint energy management and cost minimization problem

(3) We propose a low-cost computation offloading algorithm based on Lyapunov optimization framework to save the problem. This algorithm achieves nearly optimal costs meanwhile stabilizes the energy queues of edge devices and can be implemented in a distributed manner easily

The rest of this article is organized as follows. In Section 2, we conduct literature review. In Section 3, we present the system model and problem formulation. In Section 4, we present our algorithm and analysis. In Sections 5 and 6, we show the numerical results and conclude the article.

2. Related Work

In MEC systems, edge devices are allowed to offload some or all of the computation tasks to edge servers which releases the limited resources of mobile edge devices. However, due to the time-varying nature of wireless channels [18, 19], the mobility of edge devices, and the heterogeneity of computation intensive application tasks, the computation offloading and resource allocation problems remain challenging.

There have been many works [9–15, 19–21] focus on developing offloading strategies to reduce the energy consumption and computation latency. Chen et al. [10] developed an offloading strategy based on a self-adaptive particle swarm optimization algorithm to reduce the system energy consumption for DNN-based smart IoT systems. Zhao et al. [12] proposed a task scheduling and partial offloading method to minimize the energy consumption of edge devices under a deadline constraint. Considering that the wireless channel state and task arrival process are uncertain and dynamic, Chen et al. [13] developed an online and polynomial-time-complexity algorithm based on Lyapunov optimization techniques aiming at minimizing average transmission energy consumption while guaranteeing the average queueing latency. Yang et al. [15] considered both the heterogeneity of edge servers and the mobility of mobile devices and proposed an optimal offloading node selection strategy based on changes in available network bandwidth and location of mobile devices to minimize the offloading time. However, those strategies are mainly presented to MEC systems without energy harvesting and need to be farther improved to take full use of the energy harvested from the environment.

Energy harvesting technologies are supposed to be an efficient solution for MEC systems due to its self-sustainable nature [22, 23]. However, new challenges arise in jointly designing the energy management and offloading strategies since the harvestable energy is usually unpredictable in real world. Sun et al. [24] formulated the offloading problem in energy harvesting MEC systems as a joint minimization problem of energy consumption and delay in the long term and proposed a novel algorithm based on the reinforcement learning approach with noisy neural networks to solve the problem. Zhang et al. [25] proposed a continuous control-based deep reinforcement learning approach to minimize the execution time and energy consumption. Zhou et al. [26] proposed a Lyapunov optimization-based algorithm to minimize the time average of a weighted sum of energy consumption and execution delay meanwhile stabilize the battery energy queue. Most of those works simply use the minimization of energy consumption as one of the optimization objective and do not consider the costs of edge servers, which may lead those strategies offloading too much tasks to servers. At this point, the harvestable energy cannot be fully utilized, and the running mode of MEC systems is not fully characterized.

3. System Model and Problem Formulation

In this section, we introduce the system models for a mobile energy harvesting MEC system. The system contains M edge devices and N edge servers. We use $\mathcal{M} \triangleq \{1, 2, \cdots, M\}$ and $\mathcal{N} \triangleq \{1, 2, \cdots, N\}$ to denote edge devices (i.e., users) and edge servers, respectively. Time is slotted in this paper, and τ is the length of each time slot. The time slot index is t and $t \in \mathcal{T}$, where $\mathcal{T} \triangleq \{1, 2, \cdots\}$. Edge devices keep traveling while edge servers are stationary. The distance between edge device m and edge server n is denoted by $d_{m,n}(t)$.

3.1. Task Model. Computation tasks in this paper are delay-sensitive, and the execution deadline of those tasks is no greater than the length of each time slot. All the tasks

generated by edge device m are with fixed length L_m bits. Each edge devices may have a different L_m. Whether tasks are generated in time slot t is denoted by the binary variable $\xi_m(t)$. When $\xi_m(t) = 1$, it means that there are tasks arriving on edge device m in time slot t and device m needs to make offloading decisions among local computing, task offloading, and dropping. $\xi_m(t)$ is with an independently identical distribution (i.i.d.) Bernoulli process with $\mathbb{E}(\xi_m(t)) = \lambda_m$, where $0 \le \lambda_m \le 1$. Let $\xi(t) = (\xi_m(t), m \in \mathcal{M}), t = 0, 1, 2, \cdots$ be the task generation vector in time slot t.

Denote $x_{ml}(t)$, $x_{m,n}(t)$, and $x_{md}(t)$ be the decision variables of computation offloading. $x_{ml}(t) = 1$ and $x_{md}(t) = 1$ indicated that the tasks are executed locally and dropped, respectively. $x_{m,n}(t) = 1$ indicated that edge device m chooses edge server n to offload tasks in time slot t. The computation offloading decision vector in time slot t can be defined as $x(t) = (x_{ml}(t), x_{md}(t), x_{m,n}(t), m \in \mathcal{M} \& n \in \mathcal{N})$. In time slot t, device m offloads tasks to one server at most, which means $\sum_{n=1}^{N} x_{m,n}(t) \le 1$. In every time slots, tasks must be executed locally, remotely, or dropped. Hence, when $\xi_m(t) = 1$, we have

$$x_{ml}(t) + \sum_{n=1}^{N} x_{m,n}(t) + x_{md}(t) = 1. \tag{1}$$

3.2. Local Computation Model. Assume that it takes c_m CPU cycles for edge device m to process 1 bit of computing task; then, the local computation delay is

$$D_{ml}(t) = \frac{c_m \cdot L_m}{f_m(t)}, \tag{2}$$

where $f_m(t)$ is the CPU frequency in time slot t. Considering that tasks must be executed before the next time slot, the local computation delay should satisfy

$$D_{ml}(t) \le \tau. \tag{3}$$

Assume that the dynamic voltage and frequency scaling technologies (DVFS) [27] are adopted in the edge devices, and the CPU frequencies scheduled for tasks remain the same in a single time slot. The computation power of local execution is

$$p_{ml}(t) = \kappa f_m^3(t), \tag{4}$$

where κ is a parameter that depends on the chip architecture. According to (2) and (4), if there are tasks to execute locally, the computation energy is

$$E_{ml}(t) = D_{ml}(t) \cdot p_{ml}(t) = \kappa \cdot c_m \cdot L_m \cdot f_m^2(t). \tag{5}$$

Due to the limited computing power of CPU, the CPU frequency should satisfy

$$0 \le f_m(t) \le f_m^{\max}. \tag{6}$$

3.3. Offloading Computation Model. The data associated with the tasks to be offloaded to servers are transmitted over wireless links. Since MEC servers are usually equipped with high performance CPUS, we assume that the task execution time of MEC server can be ignored for convenience. We also assume that the output results of those tasks are of small sizes and the delay for returning the results back is negligible [26, 28, 29]. Let $d_{m,n}(t)$ be the distance between edge device m and edge server n in time slot t; then, the channel power gain can be calculated by

$$g_{m,n}(t) = \gamma_{m,n}(t) \cdot g_0 \cdot \left(\frac{d_0}{d_{m,n}(t)}\right)^{\psi}, \tag{7}$$

where $\gamma_{m,n}(t)$ is the small-scale fading channel power gain, g_0 is the channel loss coefficient, and d_0 and ψ are reference distance and the channel loss exponent, respectively.

According to Shannon theorem, the maximum uplink transmission rate between m and n can be expressed by

$$r\left(g_{m,n}(t), p_{m,n}(t)\right) = w \cdot \log_2\left(1 + \frac{g_{m,n}(t) \cdot p_{m,n}(t)}{\sigma^2}\right), \tag{8}$$

where σ^2 is the noisy power and $p_{m,n}(t)$ is the transmission power of m in time slot t. The transmission delay of the offloading tasks between device m and n in time slot t can be computed by

$$D_{m,n}(t) = \frac{L_m}{r\left(g_{m,n}(t), p_{m,n}(t)\right)}. \tag{9}$$

Considering that the tasks must be finished before the end of the time slot and the transmission power of edge devices is limited, hence we have

$$D_{m,n}(t) \le \tau, \tag{10}$$

$$p_m^{\min} \le p_{m,n}(t) \le p_m^{\max}, \tag{11}$$

where $p_m^{\max}$ and $p_m^{\min}$ are the maximum and minimum transmission power of edge device m. The energy consumption for m offloading those tasks to server n in time slot t is

$$E_{m,n}(t) = p_{m,n}(t) \cdot D_{m,n}(t). \tag{12}$$

3.4. Energy Model. Since time is slotted in this paper, the process of energy harvesting is noncontinuous. Let $e_m(t)$ be the harvestable energy in time slot t. We assume that $e_m(t)$ is i.i.d and satisfies

$$0 \le e_m(t) \le E_{mh}^{\max}, \tag{13}$$

where $E_{mh}^{\max}$ is the maximum value of harvestable energy at each time slot. Let $e(t) = ((e_m(t), m \in \mathcal{M}))$ be the energy arriving vector at time slot t.

We denote the residual energy of device m's battery at the beginning of time slot t as $B_m(t)$ and $B(t) = (B_m(t), m \in \mathcal{M})$ be the residual energy vector at time slot t. $B_m(0)$ is

set to 0 J, and $B_m(t)$ satisfies

$$B_m(t) \le B_m^{\max}, \tag{14}$$

where $B_m^{\max}$ is the battery capacity of device m. To avoid running out of energy, the residual energy of edge devices should be stabilized. In a queueing network (e.g., a network with back-pressure routing), data queue is stable means that $\lim_{T\longrightarrow\infty} \sup (1/T)\sum_{t=0}^{T}\sum_{m=1}^{M} Q_m(t) < \infty$, where $Q_m(t)$ is the data queue of node m. However, we say that the energy queue of device m is stable if the following is met:

$$\lim_{T\longrightarrow\infty} \frac{1}{T}\sum_{t=0}^{T} B_m(t) \ge \theta_m - \varepsilon, \tag{15}$$

where θ_m is the perturbation parameter of device m and ε is the gap between θ_m and the long-term average value of $B_m(t)$. θ_m can be set in the 0th time slot and should no less than $V \cdot \alpha$ add the maximal energy which can be consumed by m in one slot [11]. V is the parameter trading off performance between energy and costs. Constraint (15) can effectively reduce the probability of running out of energy.

For simplicity, we assume that energy consumption happens only in local computation and task offloading. Then, the residual energy of m at time slot t is

$$B_m(t+1) = \min\left[\left[B_m(t) - \xi_m \cdot \left(x_{ml}(t)\cdot E_{ml}(t) - \sum_{n=1}^{N} x_{m,n}(t)\cdot E_{m,n}(t)\right)\right]^{+} + e_m(t), B_m^{\max}\right], \tag{16}$$

where $[x]^{+} = \max[x, 0]$. When $\xi_m(t) = 1$ and there is not enough energy to offload or execute the tasks locally, the tasks will be dropped. Considering that $e_m(t)$ is unpredictable, in order to guarantee the computation offloading decision can be executed, the residual energy should satisfy

$$B_m(t) \ge \xi_m \cdot \left(x_{ml}(t)\cdot E_{ml}(t) + \sum_{n=1}^{N} x_{m,n}(t)\cdot E_{m,n}(t)\right). \tag{17}$$

3.5. Problem Formulation. In our work, the primary goal is to minimize the total costs resulted by task offloading and dropping while guaranteeing the energy of the mobile device not be exhausted. Let $\cos t_{n,m}(t)$ be the price of server n computing one bit data for device m in time slot t and $\alpha > 0$ be the penalty of dropping tasks. Then, the total cost of the MEC system can be expressed as

$$\text{cost}(t) = \sum_{m=1}^{M} \text{cost}_m(t) = \sum_{m=1}^{M} \xi_m \cdot \left(\sum_{n=1}^{N} x_{m,n}(t)\cos t_{n,m}(t)\cdot L_m + \alpha \cdot x_{md}(t)\right). \tag{18}$$

Define the transmission power vector in time slot t as $p(t) \triangleq (p_{m,n}(t), m \in \mathcal{M} \& n \in \mathcal{N})$ and the local computing frequency vector as $f(t) = (f_m(t), m \in \mathcal{M})$. The operation vector of the MEC system in time slot t can be defined as $Op(t) = (x(t), p(t), f(t))$. Therefore, the computation offloading problem can be formulated as follows:

$$\begin{aligned} &\text{P1}: \min_{Op(t)} \lim_{T\longrightarrow\infty} \frac{1}{T}\sum_{t=0}^{T-1} \text{cost}(t) \\ &\text{s.t.}(1), (3), (6), (10), (11), (13), (14), (15), (17). \end{aligned} \tag{19}$$

Since tasks have clear deadlines in this paper, we do not use delay or latency as the optimization goal. Most of previous works [9–15, 30–35] only focus on reducing energy consumption and the computation latency and do not consider the overheads of the servers. However, task offloading consumes bandwidth resource and computing tasks on the server which leads to the costs of memory and energy of the servers. Besides, service providers usually have their own pricing strategy. The price may be related to task type, computational complexity, and the usage of bandwidth and computation resources in every time slot which can be time varying. Note that as long as $\cos t_{n,m}(t)$ is i.i.d, then our algorithm proposed in the following section can achieve near-optimality system performance based on drift-plus-penalty frameworks.

4. Dynamic Offloading Algorithm Design

In this section, we will propose an online algorithm based on the framework of Lyapunov optimization to solve the computation offloading problem formulated above. By dynamically making greedy offloading decisions, the algorithm can obtain near minimum costs while pushing the residual energy of edge devices towards the configurable parameter θ_m. Considering the residual energy of battery is not i.i.d, we advocate the weighted perturbation method [36] to solve this issue.

To stabilize the energy queue (i.e., the residual energy of battery), we define a virtual energy queue $\tilde{B}_m(t) \triangleq B_m(t) - \theta_m$. It is worth to note that when the virtual queue is stable, the energy queue is also stable. Let $L(t)$ denote the Lyapunov function, which is

$$L(t) \triangleq \frac{1}{2}\sum_{m=1}^{M} \tilde{B}_m^2(t). \tag{20}$$

Then, the conditional Lyapunov drift can be defined as follows:

$$\begin{aligned} \Delta(t) &\triangleq \mathbb{E}\left[L(t+1) - L(t)\middle|\tilde{B}_m(t)\right] = \frac{1}{2}\sum_{m=1}^{M}\left(\tilde{B}_m^2(t+1) - \tilde{B}_m^2(t)\right) \\ &= \sum_{m=1}^{M}\left(\tilde{B}_m(t)\cdot Z(t) + \frac{1}{2}Z^2(t)\right), \end{aligned} \tag{21}$$

where $Z(t) = e_m(t) - \xi(t)_m \cdot (x_{ml}(t)\cdot E_{ml}(t) - \sum_{n=1}^{N} x_{m,n}(t)\cdot E_{m,n}(t))$. Considering that $e_m(t)$, $E_{ml}(t)$, $\xi_m(t)$, and $E_{m,n}(t)$ are all equal or greater than 0 and $x_{ml}(t)$ and $x_{m,n}(t)$ can not both be 1, $Z^2(t) \le \max(e_m^2(t), E_{ml}^2(t), E_{m,n}^2(t))$.

Offloading tasks to server leads additional costs; hence, $\sum_{n=1}^{N} x_{m,n}(t) \cdot E_{m,n}(t) = 0$ holds when device m needs more energy to offload tasks than execute those tasks locally. Therefore, the inequation $Z^2(t) \le \max\left(e_m^2(t), E_{ml}^2(t), E_{m,n}^2(t)\right)$ holds. Then, the following inequality also holds:

$$\Delta(\mathrm{t}) \le C + \sum_{m=1}^{M} \tilde{B}_m(t) \cdot \left(e_m(t) - \xi_m \cdot \left(x_{ml}(t) \cdot E_{ml}(t) - \sum_{n=1}^{N} x_{m,n}(t) \cdot E_{m,n}(t)\right)\right), \tag{22}$$

where $C = (M/2) \max\left(\left(E_{mh}^{\max}\right)^2, \kappa \cdot c_m \cdot L_m \cdot \left(f_m^{\max}\right)^2\right)$.

Define the virtual energy queue vector of time slot $\tilde{B}(t) \triangleq (\tilde{B}_m(t), m \in \mathscr{M})$, and then, the drift-plus-penalty function can be express as

$$\Delta_v(t) \triangleq \Delta(t) + V \cdot \mathbb{E}\left[\mathrm{cost}(t) \middle| \tilde{B}(t)\right], \tag{23}$$

where $\mathbb{E}[x \mid y]$ is the expectation of x under the condition of y. Combine (22) and (23), we have

$$\Delta_v(t) \le \sum_{m=1}^{M} \tilde{B}_m(t) \cdot \left(e_m(t) - \xi_m \cdot \left(x_{ml}(t) \cdot E_{ml}(t) - \sum_{n=1}^{N} x_{m,n}(t) \cdot E_{m,n}(t)\right)\right) \cdot + V \cdot \mathbb{E}\left[\mathrm{cost}(t) \middle| \tilde{B}(t)\right] + C. \tag{24}$$

Then, P1 can be converted to P2 which approximately minimize the right-hand side of (24) subject to every constraints of P1.

$$\mathrm{P2}: \min_{Op(t)} \xi_m \sum_{m=1}^{M} \left(\underbrace{\tilde{B}_m(t) \cdot e_m(t)}_{I1} + \underbrace{V \cdot \alpha \cdot x_{md}(t)}_{I3} - \underbrace{\tilde{B}_m(t) \cdot x_{ml}(t) \cdot E_{ml}(t)}_{I4} + \underbrace{\sum_{n=1}^{N} x_{m,n}(t)\left(V \cdot L_m \cdot \cos \mathrm{t}_{n,m}(t) - \tilde{B}_m(t) \cdot E_{m,n}(t)\right)}_{I2} \right). \tag{25}$$

4.1. Calculations of Decision Parameters. In each time slots, all the edge devices m with $\xi_m = 1$ should firstly compute decision-related parameters and then make greedy decisions from offloading, locally computing, and dropping tasks based on those parameters. For locally executing, the device m should determine the CPU frequency of locally computing and the minimum energy cost of finishing the tasks. For offloading, the device m needs to calculate the minimum energy cost of communication associated with task offloading for every server n and correspondent transmission power.

4.1.1. Calculations for Locally Executing. According to equation (5), $E_{ml}(t)$ is monotone increasing with $f_m(t)$ when $f_m(t) \ge 0$. Combine (2) and (3), we have $f_m(t) \ge (c_m \cdot L_m/\tau)$. Therefore, $E_{ml}(t) \ge \kappa \cdot (c_m^3 \cdot L_m^3/\tau^2)$ holds according to (5). Denote the minimum energy cost for locally executing and the correspondent frequency as $E'_{ml}(t)$ and $f'_m(t)$; then, we have $E'_{ml}(t) = \kappa \cdot (c_m^3 \cdot L_m^3/\tau^2)$ and $f'_m(t) = (c_m \cdot L_m/\tau)$.

4.1.2. Calculations for Offloading. For every server n in m's communication range, m solves $E'_{m,n}(t) = \min\{E_{m,n}(t)\}$, s.t. (10) and (11). Note that the function $E_{m,n}(t) = f(p_{m,n}(t))$ has no obvious monotonicity; the minimum energy cost of offloading from m to n can be obtained by following traversal method. For $p_{m,n}(t)$ from $p_m^{\min}$ to $p_m^{\max}$, use Δp as the step length, calculate all $E_{m,n}(t)$ which satisfies the condition (10) according to (9) and (12), and obtain the minimum energy cost $E'_{m,n}(t)$ and the correspondent transmission power $p'_{m,n}(t)$.

4.2. Computation Offloading Algorithm. According to P2, items $I1$, $I2$, $I3$, and $I4$ are related to energy harvesting, task offloading, task dropping, and locally executing, respectively, and only an item of $I2$, $I3$, and $I4$ can be active in one time slot. Considering all those items are irrelevant, P2 can be converted into minimizing $I1$, $I2$, $I3$, and $I4$, respectively. To find the minimum values of those items, it needs to be discussed in the three cases which are $\tilde{B}_m(t) = 0$, $\tilde{B}_m(t) < 0$, and $\tilde{B}_m(t) > 0$.

When $\tilde{B}_m(t) = 0$, $I1$ and $I4$ equals to 0. Assume that the edge device m decides to offload tasks to one of those servers, then $\sum_{n=1}^{N} x_{m,n}(t) = 1$ and $I2 > 0$ holds. Assume that the edge device m decides to drop tasks, then $x_{md}(t) = 1$ and $I2 > 0$ holds. Hence, executing tasks locally leads $I2 + I3 + I4$ to be minimum (equals to 0). Note that there will be enough energy to execute the tasks locally when $\tilde{B}_m(t) = 0$; therefore, the strategy under the condition $\tilde{B}_m(t) = 0$ can be described as follows: use the optimal frequency $f'_m(t)$ to execute the tasks locally, store all or part of the harvestable energy in the battery, and update the residual energy $B_m(t+1)$ according to (16).

When $\tilde{B}_m(t) < 0$, obtain $E'_{ml}(t)$, $f'_m(t)$, $E'_{m,n}(t)$, and $p'_{m,n}(t)$ using the method mentioned above. Compute costs for every server n using $cost'_{n,m}(t) = V \cdot L_m \cdot \mathrm{cost}_{n,m}(t) - \tilde{B}_m(t) \cdot$

Lyapunov-based computation offloading algorithm.
Input: $(\xi(t), e(t), t \in \mathcal{T})$, $G = (\mathcal{M}, \mathcal{N})$, $(\text{cost}_{n,m}(t), d_{m,n}(t), \gamma_{m,n}(t), m \in \mathcal{M}, n \in \mathcal{N}, t \in \mathcal{T})$, σ, $\kappa, G = (\mathcal{M}, \mathcal{N}), g_0$, $(B_m(0), B_m^{\max}, f_m^{\max}, p_m^{\max}, L_m, c_m, m \in \mathcal{M})$.
Output: The offloading strategy of different edge devices.
for each time slot t**do**
  for each device m with $\xi_m = 0$, harvest all the energy and update $B_m(t+1)$.
  end for
  for each device m with $\xi_m = 1$**do**
    obtain $B_m(t)$ and $f'_m(t)$, then compute $\tilde{B}_m(t)$.
    if$\tilde{B}_m(t) \geq 0$, use $f'_m(t)$ for local computing.
      harvest all the energy $e_m(t)$, store it in battery, and update $B_m(t+1)$.
    else compute $E'_{ml}(t)$ according to (2),(3),(5) and (6).
      for all the n, device m**do**
        solve $E'_{m,n}(t) = \min\{E_{m,n}(t)\}$ s.t.(10), (11),
        obtain $E'_{m,n}(t)$ and $p'_{m,n}(t)$ accordingly,
        compute $\text{cost}'_{n,m}(t) = V \cdot \text{L}_m \cdot \cos \text{t}_{n,m}(t) - \tilde{B}_m(t) \cdot E'_{m,n}(t)$.
      end for
      find minimum element in $\{\text{cost}'_{n,m}(t), V \cdot \alpha, -\tilde{B}_m(t)E'_{ml}(t), n \in \mathcal{N}\}$, s.t.(13).
      if the element is $\text{cost}'_{n^*,m}(t)$, choose sever $n^*$and use $p'_{m,n^*}(t)$ to offloading tasks.
      else if the element is $-\tilde{B}_m(t)E'_{ml}(t)$, use $f'_m(t)$ for local computing.
      else drop the tasks.
      end if
      harvest all the energy, and update $B_m(t+1)$.
    end if
  end for
end for

Algorithm 1

$E'_{m,n}(t)$. Find the minimum item under the constraint (17) in the set $\{cost'_{n,m}(t), V \cdot \beta, -\tilde{B}_m(t) \cdot E'_{ml}(t), n \in \mathcal{N}\}$. If the result is $cost'_{n^*,m}(t)$, then choose server n^* to offload tasks with the power $p'_{m,n^*}(t)$. Else if the results goes to $-\tilde{B}_m(t) \cdot E'_{ml}(t)$, then use the frequency $f'_m(t)$ to locally executing tasks. Otherwise, drop the tasks. Store all the harvestable energy in the battery and update the residual energy $B_m(t+1)$ accordingly.

When $\tilde{B}_m(t) > 0$, $I3 \geq 0$, $I4 \leq 0$, and the sign of $I2$ is not guaranteed. Hence, the computation offloading decision only can be locally executing or offloading. Solve $E'_{m,n}{}'(t) = \arg\min V \cdot \text{L}_m \cdot \cos \text{t}_{n,m}(t) - \tilde{B}_m(t) \cdot E'_{m,n}{}'(t)$ and obtain $E'_{m,n}{}'$ for all the server n. $E'_{m,n}{}'$ can be obtained by using iterative calculation method mentioned above similarly. After that, solve $E'_{m,n^*}{}'(t) = \arg\min V \cdot \text{L}_m \cdot \cos \text{t}_{n^*,m}(t) - \tilde{B}_m(t) \cdot E'_{m,n^*}{}'(t)$ and obtain $p'_{m,n^*}{}'(t)$, $E'_{m,n^*}{}'(t)$, and n^*. According to (5), using $f_m^{\max}$ to execute tasks locally leads the highest energy consumption. Hence, if $V \cdot \text{L}_m \cdot \cos \text{t}_{n^*,m}(t) - \tilde{B}_m(t) \cdot E'_{m,n^*}{}'(t) < E'_{ml}{}'(t)$, choose server n^* and use the transmission power $p'_{m,n^*}{}'(t)$ to offloading tasks. Otherwise, use frequency $f_m^{\max}$ to execute tasks locally. In terms of energy harvesting, abandon all the harvestable energy.

Based on the Lyapunov optimization framework, the computation offloading strategy above can obtain nearly the minimum long time average costs and achieve the minimum costs when $V \longrightarrow +\infty$. However, the strategy always tries to push the energy queue to θ_m which may lead edge devices to try their best to consume energy when $\tilde{B}_m(t) > 0$. For example, edge devices will choose the server with the longest distance to offload tasks when the charges of servers $(\cos \text{t}_{n,m}(t), n \in \mathcal{N})$ are the same and use the largest frequency to execute tasks locally. The harvestable energy also will be discarded to increase energy consumption. In fact, executing tasks locally with $f'_m(t)$ is not only beneficial to save energy but also reduce the service fee of the servers when $\tilde{B}_m(t) \geq 0$. Besides, harvesting all the harvestable energy can efficiently increase the residual energy of edge devices and ability to cope with energy scarcity (e.g., harvesting little energy for a long time). To summarize and improve the computation offloading above, we propose Lyapunov-based computation offloading algorithm (i.e., Algorithm 1) as follows.

The complexity of Algorithm 1 is $O(m * n)$ in each time slot t. It is very lightweight and can be easily implemented in a distributed manner where each device only needs to know the distance and costs of its neighbor servers and can make offloading decisions locally. In this paper, we do not consider the interface between links. Note that Algorithm 1 will try its best to reduce offloading to achieve a lower service cost; hence, the transmission collision will also be reduced. Moreover, maximal matching-based algorithms [37, 38] can be adapted to our algorithm in the cases when the links interfere with each other.

5. Performance Evaluation

To validate the effectiveness of our proposed approach, we conduct simulations in this section. We consider a MEC

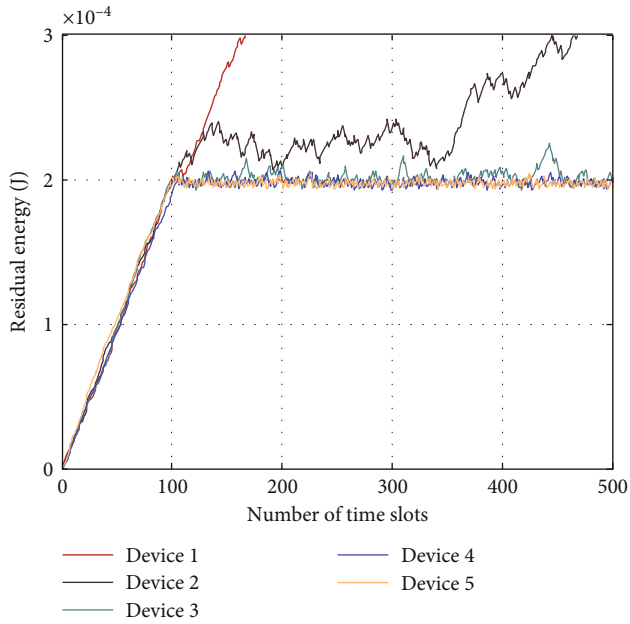

Figure 1: Residual energy of edge devices vs. time as $V = 10^{-8}$.

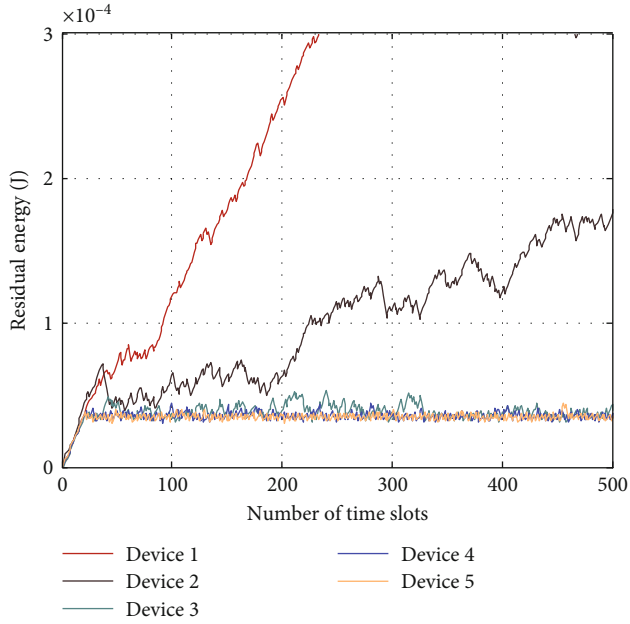

Figure 2: Residual energy of edge devices vs. time as $V = 10^{-6}$.

system with 5 mobile edge devices and 3 edge servers. For simplicity, all the harvestable energy of edge devices is i.i.d and obeys the same distribution. The harvestable energy $e_m(t)$ in every time slot t is $\lambda(t) * 4 * 10^{-7}$ J, where $\lambda(t)$ is subject to the Poisson distribution with $\mathbb{E}[\lambda(t)] = 5$. The task arriving process of each edge devices is assumed to be a Bernoulli process. The task arriving probabilities of device 1 to device 5 are $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. The maximum CPU frequencies of all the edge devices are 2 GHz. $L_m = 1000$ bit for all devices $m \in \mathcal{M}$. The time slot length τ is 4 ms, and $B_m(0)$ is 0 J for all $m \in \mathcal{M}$. We also assume $\kappa = 10^{-28}$ and $c_m = 1000$.

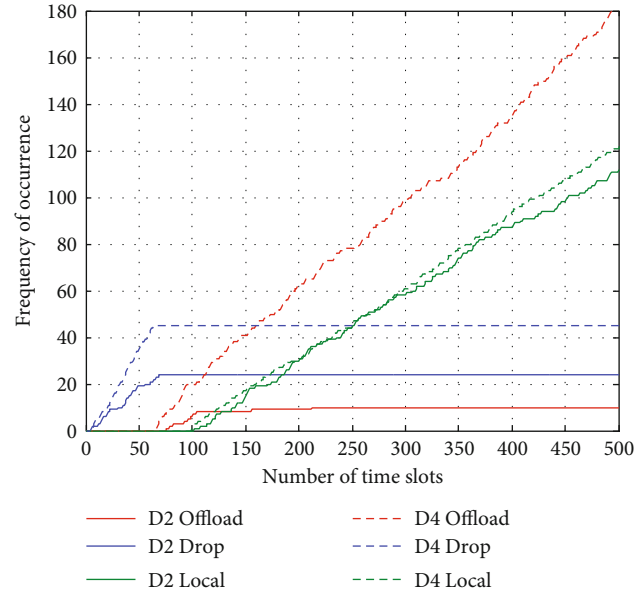

Figure 3: Computation offloading processes of devices 2 and 4.

To embody the mobility sufficiently, we assume that the distance between any edge device and any server takes a random value from $[10, 70]$ metres in every time slots. We assume that $d_0 = 5$ meters, $\sigma^2 = 10^{-13}$ W, $\cos t_{n,m}(t) = 10^{-6}$ per bit and $\gamma_{m,n}(t) = 1$ for $m \in \mathcal{M}$ and $n \in \mathcal{N}$, $\omega = 1$ MHz, $p_m^{\min} = 1$ mw and $p_m^{\max} = 50$ mw for $m \in \mathcal{M}$, and $\psi = 4$. θ_m is set to $2 * 10^{-4}$ J.

Figures 1 and 2 show the residual energy processes of all 5 edge devices under the condition $V = 10^{-8}$ and $V = 10^{-6}$, respectively. α is set to $2 * 10^{-3}$, and simulations are run for 500 time slots. From Figure 1, we can see that the residual energy of devices 3, 4, and 5 increases at first and then stabilizes around the perturbation parameters θ_m, respectively, which exactly keeps the advantages of our algorithm. The reason why the residual energy of devices 1 and 2 is with growing tendencies is that the harvestable energy is enough to sustain all the tasks being executed locally. Figure 2 shows that as V grows to 10^{-6}, the residual energy of devices 3, 4, and 5 can still reach a steady state with values which are much lower than that of $V = 10^{-8}$. Note that the average residual energy of edge devices decreases as V increases; we will show that the value of V also influences the total costs of the MEC system and can trade-off energy performances and costs later.

Figure 3 demonstrates computation offloading processes of devices 2 (D2) and 4 (D4) under $V = 10^{-8}$. At the beginning, both devices keep harvesting energy and dropping tasks to push their residual energy to θ_m (the residual energy processes are shown in Figure 1). When their residual energy increases to a certain value, both devices stop dropping tasks and begin to offload tasks to servers for remote execution. As time goes on, both devices will begin to execute tasks locally when their residual energy goes to some value (e.g., near by θ_m when $V = 10^{-8}$). Since the harvestable energy of device 2 is enough to execute all tasks locally, device 2 executes nearly all tasks some time slots later (nearly the 100th time slot). To

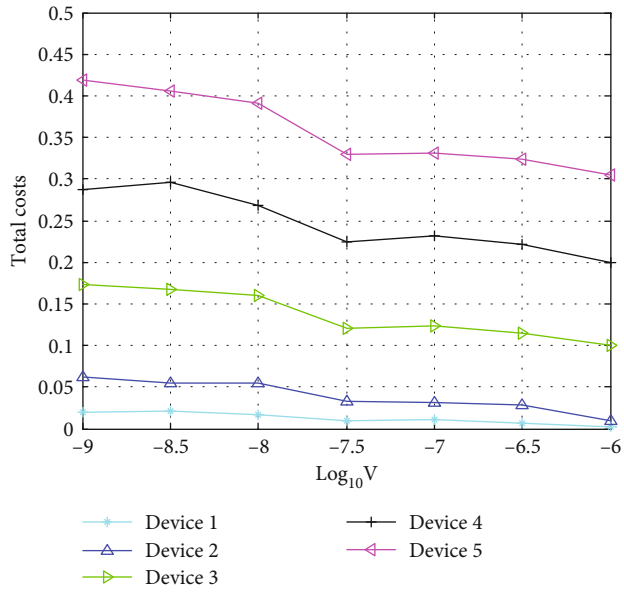

Figure 4: Total costs of edge devices vs. V.

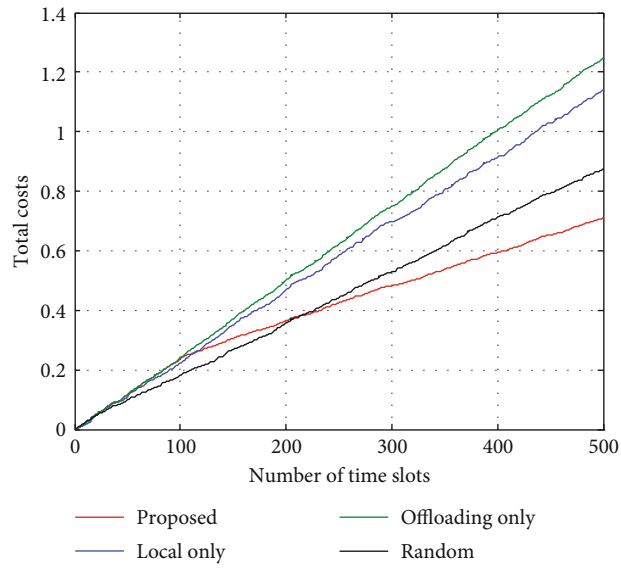

Figure 5: Total costs by different algorithms.

stabilize the energy queue, part of the tasks of device 4 is executed locally while others are offloaded to servers some time later.

According to our energy queue stability definition (showed in equation (15)), the convergence time should be $O(\theta_m - \varepsilon)$, where ε is a positive number. Hence, the convergence time is no larger than $O(\theta_m)$ which is approximately equal to $\theta_m/\mathbb{E}(e_m(t))$. In our simulation, the harvestable energy $e_m(t)$ in every slot t is $\lambda(t) * 4 * 10^{-7}$ J, where $\lambda(t)$ is subject to the Poisson distribution with $\mathbb{E}[\lambda(t)] = 5$ and θ_m is set to $2*10^{-4}$ J; hence, the convergence time is about 100 time slots. Although the residual energy of device 1 and device 2 seems never converging to a fixed value in Figure 1, the offloading decision processes have reached a

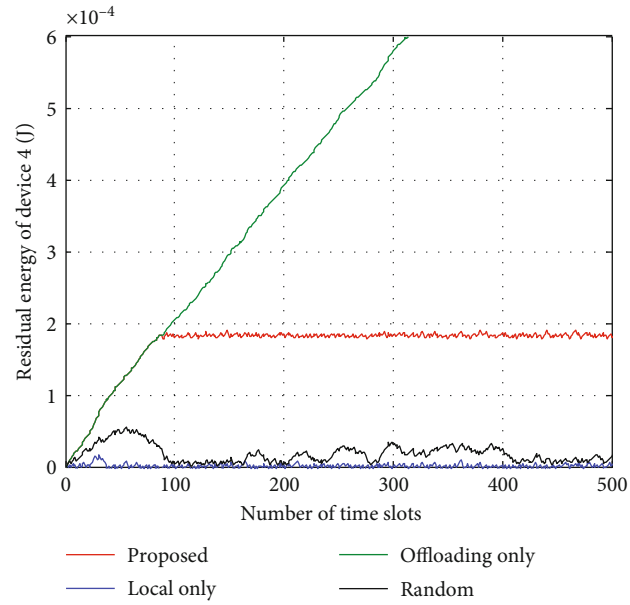

Figure 6: Residual energy processes of different algorithms over device 4.

stable state at about the 100th time slot. As can be seen from Figure 3, D2 stops dropping and offloading and executes nearly all tasks after the 100th time slot.

Figure 4 shows the total costs of all the edge devices with different parameter V. We can see that the total costs decrease with the increasement of V. As shown in Figure 1, the long-time average residual energy of edge devices decreases as the parameter V increases. Therefore, parameter V does trade-off energy performances and the total costs.

To evaluate the performance of our algorithm in terms of reducing the total system costs, we compare our proposed algorithm against three other strategies: strategy where all tasks are executed locally with the optimal frequency f'_{ml} (local only), strategy where all tasks are offloaded to the serve with the minimum service costs (offloading only), and the strategy where edge devices randomly make choice from locally executing and offloading and choose random server to offloading when they have decided to execute tasks remotely (random). Figure 5 demonstrates the total costs of all the edge devices vs. time under the condition $V = 10^{-7}$. The results show that our algorithm performs significantly better in comparison to other three comparison algorithms.

Figure 6 shows the residual energy dynamic of device 4 by using different strategies which are the proposed algorithm, local only, offloading only, and random strategy. The parameter V is set to 10^{-7}. Since the harvestable energy of devices 1 and 2 is sufficient to execute all computing tasks locally, the residual energy will keep growing until it reaches $B_m^{\max}$. Devices 3 and 5 have a similar energy process as that of device 4; hence, we only show the results of device 4. Figure 6 demonstrates that our algorithm can use the harvestable energy more effectively and stabilize the residual energy to a preset value.

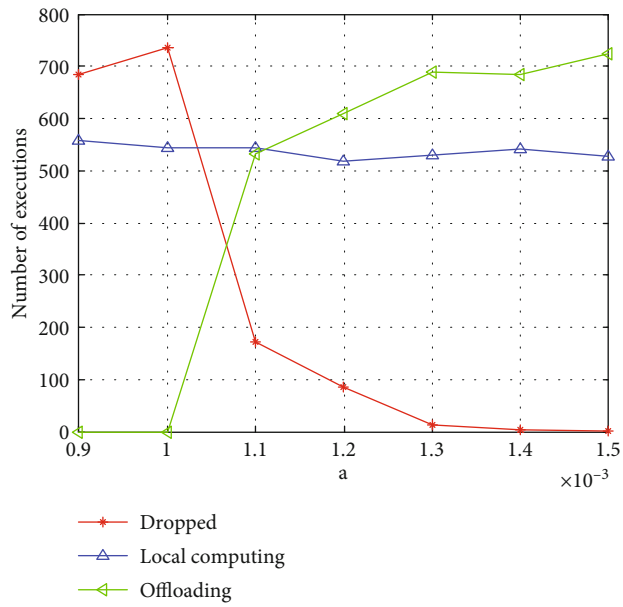

Figure 7: The influence of α on offloading decisions.

Figure 7 shows the effects of the parameter α on computation offloading decisions under the condition $V = 10^{-7}$. Y axis is the total number of execution times of all the edge devices. As α goes larger, the number of task dropping will decrease, and the number of offloading will increase. When the tasks have high intertask dependencies, we can set α to a large value to avoid task dropping.

6. Conclusions

In this paper, an online and lightweight algorithm has been proposed for solving the computation offloading problems in energy harvesting MEC systems with multiuser and multiserver. Based on stochastic Lyapunov optimization framework, the proposed algorithm can minimize the total system costs and guarantee the devices' residual energy stabilized to a preset value. It needs no prior knowledge of the harvestable energy and the task arrival rates and can make trade-off between system costs and the residual energy by a parameter V. The algorithm is also light-weighted and can be easily implemented in a distributed manner since each device can make offloading decisions locally only based on the distance and service costs of its neighbor servers. Our evaluation data shows that the proposed algorithm can take fully use of the harvestable energy and effectively reduce the total system costs caused by computing services and task droppings.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This paper is supported by the Natural Science Foundation of Shaanxi Province of China (Project No. 2020JQ-647).

References

[1] M. Li, X. Zhou, T. Qiu, Q. Zhao, and K. Li, "Multi-relay assisted computation offloading for multi-access edge computing systems with energy harvesting," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10941–10956, 2021.

[2] J. Lu, Q. Li, B. Guo et al., "A multi-task oriented framework for mobile computation offloading," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 187–201, 2022.

[3] A. Boukerche, S. Guan, and R. E. De Grande, "A task-centric mobile cloud-based system to enable energy-aware efficient offloading," *IEEE Transactions on Sustainable Computing*, vol. 3, no. 4, pp. 248–261, 2018.

[4] R. Xie, Q. Tang, S. Qiao, H. Zhu, F. Richard Yu, and T. Huang, "When serverless computing meets edge computing: architecture, challenges, and open issues," *IEEE Wireless Communications*, vol. 28, no. 5, pp. 126–133, 2021.

[5] J. Li, R. Wang, and K. Wang, "Service function chaining in industrial Internet of Things with edge intelligence: a natural actor-critic approach," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, pp. 1–10, 2022.

[6] X. Li, R. Xie, F. R. Yu, T. Huang, and Y. Liu, "Advancing software-defined service-centric networking toward in-network intelligence," *IEEE Network*, vol. 35, no. 5, pp. 210–218, 2021.

[7] E. El Haber, T. M. Nguyen, and C. Assi, "Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3407–3421, 2019.

[8] H. Tout, A. Mourad, N. Kara, and C. Talhi, "Multi-persona mobility: joint cost-effective and resource-aware mobile-edge computation offloading," *IEEE/ACM Transactions on Networking*, vol. 29, no. 3, pp. 1408–1421, 2021.

[9] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 1, pp. 1–14, 2022.

[10] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 683–697, 2022.

[11] H. Zhao, W. Du, W. Liu, T. Lei, and Q. Lei, "QoE aware and cell capacity enhanced computation offloading for multi-server mobile edge computing systems with energy harvesting devices," in *IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 671–678, Guangzhou, China, 2018.

[12] J. Zhao, L. Deng, Y. Liu, and J. Sun, "Energy-efficient partial offloading in mobile edge computing under a deadline constraint," in *International Conference on Intelligent Technology and Embedded Systems (ICITES*, pp. 14–21, Chengdu, China, 2021.

[13] 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 Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1050–1060, 2021.

[14] S. Mao, L. Liu, N. Zhang et al., "Reconfigurable intelligent surface-assisted secure mobile edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 6647–6660, 2022.

[15] G. Yang, L. Hou, X. He, D. He, S. Chan, and M. Guizani, "Offloading time optimization via Markov decision process in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2483–2493, 2021.

[16] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: a survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.

[17] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: architecture, advances and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[18] H. Han, L. Fang, W. Lu, W. Zhai, Y. Li, and J. Zhao, "A GCICA grant-free random access scheme for M2M communications in crowded massive MIMO systems," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 6032–6046, 2022.

[19] D. Zhai, C. Wang, R. Zhang, H. Cao, and F. R. Yu, "Energy-saving deployment optimization and resource management for UAV-assisted wireless sensor networks with NOMA," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 6609–6623, 2022.

[20] X. Guo, L. Liu, Z. Chang, and T. Ristaniemi, "Data offloading and task allocation for cloudlet-assisted ad hoc mobile clouds," *Wireless Networks*, vol. 24, no. 1, pp. 79–88, 2018.

[21] H. Zhu, Q. Wu, X. Wu, Q. Fan, P. Fan, and J. Wang, "Decentralized power allocation for MIMO-NOMA vehicular edge computing based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12770–12782, 2022.

[22] A. Bozorgchenani, S. Disabato, D. Tarchi, and M. Roveri, "An energy harvesting solution for computation offloading in fog computing networks," *Computer Communications*, vol. 160, no. 1, pp. 577–587, 2020.

[23] X. He, Y. Chen, and K. K. Chai, "Delay-aware energy efficient computation offloading for energy harvesting enabled fog radio access networks," in *IEEE 87th Vehicular Technology Conference (VTC Spring)*, pp. 1–6, Porto, Portuga, 2018.

[24] Z. Sun, M. Zhao, and M. R. Nakhai, "Computation offloading in energy harvesting powered MEC network," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, Montreal, QC, Canada, 2021.

[25] J. Zhang, J. Du, C. Jiang, Y. Shen, and J. Wang, "Computation offloading in energy harvesting systems via continuous deep reinforcement learning," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, Dublin, Ireland, 2020.

[26] W. Zhou, L. Xing, J. Xia, L. Fan, and A. Nallanathan, "Dynamic computation offloading for MIMO mobile edge computing systems with energy harvesting," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 5172–5177, 2021.

[27] Y. Mao, C. S. You, J. Zhang, K. B. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communication Surveys & Tutorials*, vol. 70, no. 2017, pp. 2322–2358, 2017.

[28] Y. Zuo, S. Jin, S. Zhang, Y. Han, and K.-K. Wong, "Delay-limited computation offloading for MEC-assisted mobile blockchain networks," *IEEE Transactions on Communications*, vol. 69, no. 12, pp. 8569–8584, 2021.

[29] W. Zhang, G. Zhang, and S. Mao, "Joint parallel offloading and load balancing for cooperative-MEC systems with delay constraints," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4249–4263, 2022.

[30] M. Wu, W. Qi, J. Park, P. Lin, L. Guo, and I. Lee, "Residual energy maximization for wireless powered mobile edge computing systems with mixed-offloading," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4223–4228, 2022.

[31] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.

[32] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3341–3356, 2020.

[33] S. Guan, A. Boukerche, and A. Loureiro, "Novel sustainable and heterogeneous offloading management techniques in proactive cloudlets," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 2, pp. 334–346, 2021.

[34] J. Feng, L. Liu, Q. Pei, and K. Li, "Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2687–2700, 2022.

[35] J. Feng, W. Zhang, Q. Pei, J. Wu, and X. Lin, "Heterogeneous computation and resource allocation for wireless powered federated edge learning systems," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3220–3233, 2022.

[36] M. J. Neely and L. Huang, "Dynamic product assembly and inventory control for maximum profit," in *IEEE Conference on Decision and Control (CDC)*, pp. 2805–2812, Atlanta, USA, 2010.

[37] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 302–315, 2006.

[38] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 556–564, Tel Aviv, Israel, 2000.