Journal of Systems Architecture 153 (2024) 103194

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Lyapunov-guided deep reinforcement learning for delay-aware online task offloading in MEC systems

Longbao Dai [a], Jing Mei [a,*], Zhibang Yang [b], Zhao Tong [a,1], Cuibin Zeng [a], Keqin Li [c,2]

[a] College of Information Science and Engineering, Hunan Normal University, Changsha, Hunan, 410081, China
[b] Hunan Province Key Laboratory of Industrial Internet Technology and Security, Changsha University, Changsha 410022, China
[c] Department of Computer Science, State University of New York, New Paltz, New York 12561, USA

ARTICLE INFO

Keywords:
Deep reinforcement learning
Lyapunov optimization
Mobile edge computing
Task offloading

ABSTRACT

With the arrival of 5G technology and the popularization of the Internet of Things (IoT), mobile edge computing (MEC) has great potential in handling delay-sensitive and compute-intensive (DSCI) applications. Meanwhile, the need for reduced latency and improved energy efficiency in terminal devices is becoming urgent increasingly. However, the users are affected by channel conditions and bursty computational demands in dynamic MEC environments, which can lead to longer task correspondence times. Therefore, finding an efficient task offloading method in stochastic systems is crucial for optimizing system energy consumption. Additionally, the delay due to frequent user–MEC interactions cannot be overlooked. In this article, we initially frame the task offloading issue as a dynamic optimization issue. The goal is to minimize the system's long-term energy consumption while ensuring the task queue's stability over the long term. Using the Lyapunov optimization technique, the task processing deadline problem is converted into a stability control problem for the virtual queue. Then, a novel Lyapunov-guided deep reinforcement learning (DRL) for delay-aware offloading algorithm (LyD2OA) is designed. LyD2OA can figure out the task offloading scheme online, and adaptively offload the task with better network quality. Meanwhile, it ensures that deadlines are not violated when offloading tasks in poor communication environments. In addition, we perform a rigorous mathematical analysis of the performance of Ly2DOA and prove the existence of upper bounds on the virtual queue. It is theoretically proven that LyD2OA enables the system to realize the trade-off between energy consumption and delay. Finally, extensive simulation experiments verify that LyD2OA has good performance in minimizing energy consumption and keeping latency low.

1. Introduction

As 5G technology and the Internet of Things (IoT) advance, the proliferation of smart terminal devices has surged. With this comes a growing demand for processing delay-sensitive and compute-intensive (DSCI) applications [1]. This poses new challenges to the traditional cloud computing paradigm. The centralized structure of cloud data centers frequently falls short in addressing the business requirements for low latency and high throughput associated with processing DSCI-type tasks [2]. To address these challenges, a rising idea called Mobile Edge Computing (MEC) sinks computing, data storage, and many other services to the network's edge [3].

MEC deploys computing resources and services on base stations, edge servers, and other devices near end-users, and leverages the distributed nature of these devices to process tasks [4]. By moving computing tasks and data processing closer to the network's edge, MEC reduces the latency and network congestion that can occur when routing all data to remote cloud data centers. This edge-based approach supports DSCI-type applications including virtual reality, audio and video streaming, in-vehicle communications, and online gaming for optimal performance and improved user experience [5].

MEC performs well in handling DSCI-type applications, but there are still some problems. Firstly, the wireless channel condition in MEC systems has a direct influence on how much energy is used for transmission tasks by IoT devices [6], i.e., when the connection is unreliable, more energy is used to transmit data. Although the system can employ the greedy algorithm to choose a time slot with enhanced network conditions for workload offloading, minimizing transmission

* Corresponding author.
E-mail addresses: awaken6758@gmail.com (L. Dai), jingmei1988@163.com (J. Mei), yangzb@ccsu.edu.cn (Z. Yang), tongzhao@hunnu.edu.cn (Z. Tong), 202220294009@hunnu.edu.cn (C. Zeng), lik@newpaltz.edu (K. Li).
[1] Member, IEEE.
[2] Fellow, IEEE.

https://doi.org/10.1016/j.sysarc.2024.103194
Received 26 December 2023; Received in revised form 19 April 2024; Accepted 28 May 2024
Available online 31 May 2024

energy use, the task queue length in the buffer might become extremely long as a result of this diversion, possibly even exceeding the worst-case latency. Secondly, in MEC systems with high demand, DSCI-type tasks could have lengthy queuing delays, which would break their latency constraints, due to the inability of the system to respond quickly to certain bursty computational demands [7]. At the same time, the unpredictability and dynamic nature of network quality and channel state during real-time wireless communication are significantly influenced by factors like device location and network congestion [8]. Additionally, the devices' task arrival rate is also difficult to obtain. Hence, to optimize energy consumption, it is essential to find an efficient task-offloading approach in stochastic systems.

To address the aforementioned challenges, we consider the system's dynamic nature and propose a method that combines Lyapunov optimization with DRL. Lyapunov optimization provides theoretical guarantees for system stability and long-term performance. However, Lyapunov methods typically require a precise model of the system, which is often unattainable in practical applications. Meanwhile, DRL is capable of learning optimal policies directly from data without explicit models, but it lacks guarantees for system stability. We use Lyapunov optimization to guide the DRL learning process, ensuring that the learned strategies not only optimize immediate rewards but also maintain system stability and long-term performance. By integrating these two approaches, we exploit a real-time offloading strategy for DSCI-type applications.

In this study, by restricting the device's task queue length threshold, we manifest the delay-sensitive nature of tasks and manage the task queue length to satisfy those characteristics. Moreover, most studies focus solely on optimizing the energy consumption of the terminal device, neglecting the energy usage of the MEC server. The purpose of this article is to reduce both devices and the MEC server's long-term energy usage. Here, we outline the primary contributions made in this article.

- We take into account the online task offloading issue for a single-MEC scenario with several IoT devices. In this scenario, the microprocessor is built into each device to process the tasks. Moreover, each device can dynamically adjust its computing ability and communication power consumption in response to the system state.
- We frame the online task offloading issue as a dynamic optimization problem to minimize energy consumption while upholding task queue stability in the long term. Concurrently, we set queue length thresholds for each device to enable efficient processing of DSCI-type tasks. Utilizing Lyapunov optimization theory, the original problem that is contingent on further uncertain information is reformulated into a new problem solely reliant on the current system state. To tackle this problem, we develop a novel Lyapunov-guided DRL for the delay-aware offloading algorithm (Ly2DOA).
- We validated the effectiveness of Ly2DOA through a series of experiments, observing the variations in queue lengths and system energy consumption when offloading tasks under different parameters.

The remainder of this paper is structured as follows. In Section 2, we summarize the work related to task offloading in MEC. In Section 3, we provide a thorough description of the system model and carefully frame the optimization problem. In Section 4, we use the Lyapunov optimization technique and design a novel Lyapunov-guided DRL for a delay-aware energy-efficient offloading algorithm called Ly2DOA to address this problem. In Section 5, we analyze Ly2DOA performance through a series of simulation experiments. In Section 6, we conclude this article.

2. Related work

In recent years, MEC, as a technology for deploying offloading nodes at the network edge, has received significant attention from the academic community. Islam et al. [9] summarized the MEC task offloading solutions proposed by various researchers, highlighting the deficiencies and challenges currently faced in the field of task offloading. Additionally, they identified future research directions. Wu et al. [10] proposed a delay-aware energy-efficient online offloading algorithm that adaptively offloads more tasks when network quality is favorable while ensuring that tasks do not violate their processing deadlines in scenarios of poor network quality. Chen et al. [11] investigated a Multi-User Caching enabled MEC environment that allows for the temporary storage and deferred execution of user-requested tasks on MEC servers. They designed a heuristic algorithm to optimize task caching locations and bandwidth, thereby reducing the overall energy consumption of the system. Cheng et al. [12] investigated the optimization problem in IoT fog computing scenarios and proposed an adaptive offloading scheme. This scheme determines offloading decisions in real time based on the computational demands of mobile terminals and dynamically allocates computing and communication resources to minimize system costs. The aforementioned studies achieve commendable performance in terms of energy consumption and computational capabilities. Tong et al. [13] proposed a task offloading algorithm framed within federated learning. In simulated scenarios, this algorithm reduces device energy consumption and task response times while safeguarding user privacy. However, most of these studies assume that task and channel states are known in advance or can be predicted based on historical information. In real-world scenarios, devices experience dynamic task arrival rates, and wireless channel quality is influenced by various factors, making it challenging to precisely predict their specific values.

To address the limitations of the aforementioned studies, recent research has employed various stochastic optimization strategies that allow for the online determination of offloading decisions without any prior information. Among these, the Lyapunov method is the widely applied stochastic optimization technique [14]. In the research literature on the application of Lyapunov optimization methods, the system first formulates a quadratic Lyapunov function based on the queue backlog vector. Next, establish an upper limit for the Lyapunov "*drift-plus-penalty*" function and minimize this limit to maintain overall system stability while optimizing the system objective function [15]. According to the perturbed Lyapunov optimization technique and game theory, Xia et al. [16] designed a real-time offloading scheme for EH-enabled MEC offloading systems. The scheme allocates on-demand resources and determines the offloading scheme for heterogeneous tasks online through an offloading pre-screening criterion. Li et al. [17] studied offloading strategies in heterogeneous wireless networks, and created a queueing model to depict the end-user load offloading issue. Besides, this paper employed the Lyapunov optimization technique to balance the system optimization objective and the buffer queue length. Using the Lyapunov optimization technique, Li et al. [18] considered the spatiotemporal optimality of long-term system cost minimization and decoupled a series of bidirectional offloading problems by using gap-preserving transformation. Gao et al. [19] exploited traffic prediction to formulate the stochastic resource scheduling issue in a multilayer fog computing system as a dynamic system optimization issue. Their goal was to optimize average power consumption while ensuring queue stability throughout the system. Zhang et al. [20] proposed a centralized resource allocation and distributed task offloading strategy, designed to optimize resource utilization and reduce energy consumption while meeting the latency requirements of real-time video applications.

Recently, Lyapunov optimization and artificial intelligence (AI), especially machine learning (ML), have been combined in various studies to address dynamic system optimization issues, with deep reinforcement learning (DRL) methods being particularly widely used in MEC. Du et al. [21] introduced and surveyed AI/ML and its application in

6G based. In a multiuser MEC scenario with time-varying channels, Bi et al. [22] transformed the problem of maximizing the device offloading rate into a mixed-integer nonlinear optimization problem (MINOP). A DRL approach was utilized to construct a real-time offloading scheme that determines the device's offloading policy in real time for each time slot. Different from most literature, this paper used the DRL method to solve the task offloading scheme without modeling it as a Markov decision process (MDP). Jia et al. [23] investigated a MEC-assisted Telematics system with constrained computational resources, to maximize the system's average data processing rate. A locally observed multi-intelligence proximal policy optimization algorithm was designed to reduce the collaboration overhead, but the strategy of this paper cannot efficiently control how quickly each device responds to tasks. Zhang et al. [24] proposed a collaborative deep neural network (DNN) inference framework that dynamically determines computational and network resource allocation schemes to jointly optimize latency and energy consumption. Gao et al. [25] proposed a hierarchical computational partitioning strategy for the DNN to determine the binary offloading problem in multi-mobile device MEC networks. The cost of the whole system is minimized by scheduling the task offloading strategy for each device, but The cost does not account for the MEC server's energy use. Fan et al. [26] considered a DNN model integrating task offloading, computation, and communication resource allotment, and designed a DRL algorithm based on deep deterministic policy gradient (DDPG) to solve the near-optimal solution to optimize the total system cost. Nevertheless, the majority of prior research on offloading strategies and task scheduling has considered only computationally intensive tasks, and few studies have considered both latency-sensitive and computationally intensive aspects.

This study reflects the delay-sensitive nature of tasks by constraining the queue backlog thresholds and managing the task queue length to accommodate these characteristics. The distinctions between this research and our previous study [27] are as follows:

- This study incorporates the energy consumption of MEC servers into the optimization model, dynamically adjusting the operational frequency of MEC servers according to varying workloads. This improvement helps reduce the power consumption of MEC servers, save operational costs, and enhance the sustainability of MEC deployments. Specifically, this paper utilizes a DDPG algorithm to dynamically adjust task offloading decisions and the operational states of MEC servers. Following the training of the DNN, real-time decisions are made based on the current server load and task demands, aiming to minimize the energy consumption of the entire system. This dynamic adjustment allows for the reduction of server operations without impacting task processing latency and reliability, thereby decreasing unnecessary energy expenditure during periods of low demand.
- Compared to Ref. [27], this study employs DRL methods to simultaneously reduce system energy consumption, maintain queue stability, and dynamically determine task offloading decisions and operational states of MEC servers. In contrast, the knapsack problem typically focuses on optimizing a single objective and is not well-suited to address the complex optimization issues presented in this manuscript. Moreover, the knapsack problem in Ref. [27] is structurally simple and fixed, whereas DRL leverages neural networks to learn from and adapt to highly complex and nonlinear environmental characteristics.

3. System model and problem formulation

We consider an MEC scenario consisting of N IoT devices in a set $\mathcal{N} = \{1, 2, \ldots, N\}$ and one MEC server in this paper, which is depicted in Fig. 1. The IoT devices in the scenario communicate with the MEC server in the vicinity by the wireless network. Given the system's dynamic nature, the entire system operates in discrete time, with time slots represented as $\mathcal{T} = 0, 1, \ldots, T$ [28]. Assuming each time slot, designated as t, is of equal duration τ, we can approximate the system as static during each time slot [29].

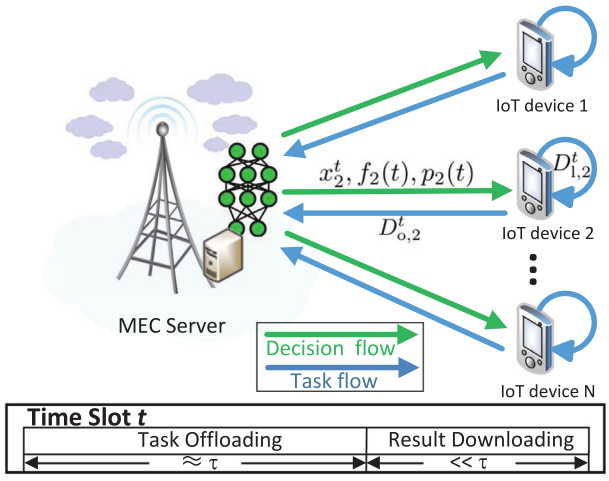


Fig. 1. The system architecture.

3.1. Task model

At the start of each time slot t, every device within the MEC system generates computational tasks A_i^t (in Mbits). The tasks are not macro-level DAG-based multi-stage computations, but a single DNN offloading. As mentioned in [30], the tasks generated by the devices have bit-wise independence, allowing for the application of partial offloading schemes on each device. The generated task A_i^t is initially placed in the task buffer queue $q_i(t)$, waiting to be processed. Additionally, the task buffer queue follows the first-come-first-served (FCFS) principle.

3.2. Computation and energy models

3.2.1. Local computing

In this scenario, individual IoT devices can independently execute computational tasks locally. For the ith IoT device, its local CPU computation frequency is denoted by $f_i(t)$ (in MHz unit), with an upper bound of f_i^{max}, and the number of CPU cycles required for computing one bit is ϕ. Thus, the tasks processed locally in slot t is

$$D_{l,i}^t = \frac{f_i(t) \times \tau}{\phi}. \tag{1}$$

The energy expended in local computing is

$$E_{l,i}^t = \xi f_i^3(t) \times \tau + \omega_0, \tag{2}$$

where ω_0 and ξ denote the constant circuit energy consumption and effective capacitance coefficient of CPU, respectively.

3.2.2. Task offloading

Assuming that each device transmits tasks to the MEC server using Orthogonal Frequency Division Multiple Access (OFDMA) transmission. We utilize the binary variable x_i^t, where $x_i^t = 1$ or 0, to indicate whether device i participates in task offloading during the specific time slot t. Assuming that the devices participating in task offloading in time slot t have equal bandwidth, denoted as $B/\sum_{i=1}^{N} x_i^t$, where B represents the total system bandwidth. Let $p_i(t)$ represent the transmit power, subject to the constraint $p_i(t) \le p_i^{max}$, where p_i^{max} denotes the maximum power. The channel power gain is indicated by h_i^t. Because of the channel's stochastic nature, its power gain fluctuates dynamically. Consequently, we specify R_i^t (in Mbits/s) as the attainable data transmission rate for tasks, which is denoted as

$$R_i^t = \frac{x_i^t B}{\sum_{i=1}^{N} x_i^t} \log_2 \left(1 + \frac{h_i^t p_i(t)}{N_0 B / \sum_{i=1}^{N} x_i^t + \chi} \right),$$

where N_0 and χ are the noise power spectral density and average co-channel interference power [26].

Due to the ability of the MEC server to increase the transmission power to reduce downlink transmission delay, and considering that the size of computational results is typically small, we neglect the latency associated with edge computing and result downloading. Thus, the number of offloaded tasks during slot t is represented as

$$D_{\mathrm{o},i}^t = R_i^t \times \tau x_i^t. \tag{3}$$

The energy consumed by task offloading for the ith IoT device is

$$E_{\mathrm{o},i}^t = p_i(t) \times \tau x_i^t. \tag{4}$$

3.2.3. MEC server processing

For generality, in this study, we use $\sum_{i=1}^{N} D_{\mathrm{o},i}^t$ (in Mbits) to represent the tasks arriving at the MEC server during time slot t. The MEC's CPU frequency, denoted as $f_{\mathrm{m}}(t)$ (in MHz), is subject to an upper bound of f_{m}^{max}, and ψ (in cycles/bit) represents the CPU cycles needed for bit task processing. For the MEC server, we assume that all tasks arriving in time slot t must be processed, i.e.,

$$A_{\mathrm{M}}^t = \frac{f_{\mathrm{m}}(t) \times \tau}{\psi} = \sum_{i=1}^{N} D_{\mathrm{o},i}^t. \tag{5}$$

The energy consumed by MEC server processing is

$$E_{\mathrm{M}}^t = \zeta f_{\mathrm{m}}^3(t) \times \tau, \tag{6}$$

where the constant parameter ζ varies with chip architecture [30].

3.3. Queue framework evolution model

For the ith IoT device, $q_i(t)$ represents the queue length in time slot t, with $q_i(0)$ initialized to 0. Subsequently, $q_i(t)$ is updated to

$$q_i(t+1) = \max\left\{q_i(t) + A_i^t - D_i^t, 0\right\}, \tag{7}$$

where $D_i^t = D_{\mathrm{l},i}^t + D_{\mathrm{o},i}^t$.

In this study, our goal is to sustain the stability of task queues among IoT devices, ensuring that each device's queue backlog consistently stays beneath a predetermined threshold for an extended duration. As tasks arrive and are queued before processing, the device's queue length is strongly linked to task waiting times [10]. By rigorously constraining the queue length, we ensure that tasks are offloaded within the specified timeframe. Thus, there is a maximum average queue backlog constraint that applies across multiple time slots in this scenario, i.e.,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left\{q_i(t)\right\} \le \rho_i, \ \forall i \in \mathcal{N}; \tag{8}$$

where ρ_i represents the queue length threshold for device i.

3.4. Problem formulation

Our scheme aims to both maintain the queue length of each device below a threshold and minimize the overall energy consumption, encompassing both IoT devices and MEC server. Consequently, in time slot t, the energy consumption of the whole system is expressed by

$$\begin{aligned} e(t) &= \sum_{i=1}^{N} \left(E_{\mathrm{l},i}^t + E_{\mathrm{o},i}^t\right) + E_{\mathrm{M}}^t \\ &= \sum_{i=1}^{N} \left(\xi f_i^3(t) + x_i^t p_i(t)\right)\tau + \zeta f_{\mathrm{m}}^3(t)\tau + N\omega_0. \end{aligned} \tag{9}$$

Based on the aforementioned model description, one can notice that wireless channel quality and task arrivals at devices dynamically change with time. As a result, the overall energy consumption of the system also experiences fluctuations. Hence, we counteract the system's dynamism by estimating the anticipated power usage over an extended time horizon. Since the energy consumption of the entire system is contingent on device offloading choices and the allocation of computation and communication resources, the optimization objective can be accomplished through jointly optimizing task offloading decisions and resource allocation. Let $\boldsymbol{\pi}^t = \{\boldsymbol{x}^t, \boldsymbol{f}^t, \boldsymbol{p}^t\}$, where $\boldsymbol{x}^t = \{x_i^t\}_{i=1}^N$, $\boldsymbol{f}^t = \{f_i(t)\}_{i=1}^N$, and $\boldsymbol{p}^t = \{p_i(t)\}_{i=1}^N$. Then, we can formulate our optimization problem as

$$\begin{aligned} \mathbf{P1}: &\min_{\boldsymbol{\pi}^t} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left\{e(t)\right\} \\ \text{s.t. } &\mathrm{C1}: x_i^t \in \{0,1\}; \\ &\mathrm{C2}: D_{\mathrm{l},i}^t + D_{\mathrm{o},i}^t \le q_i(t) + A_i^t; \\ &\mathrm{C3}: A_{\mathrm{M}}^t \le \frac{f_{\mathrm{m}}(t) \times \tau}{\psi}; \\ &\mathrm{C4}: f_i^{\min} \le f_i(t) \le f_i^{\max}; \\ &\mathrm{C5}: p_i^{\min} \le p_i(t) \le p_i^{\max}; \\ &\mathrm{C6}: \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left\{q_i(t)\right\} \le \rho_i; \\ &\mathrm{C1}-\mathrm{C6 \ satisfy} \ i \in \mathcal{N} \ \text{and} \ t \in \mathcal{T}. \end{aligned} \tag{10}$$

Here, C1 indicates whether the device has the capability for remote offloading; C2 guarantees that the current processing capacity of the device does not surpass its existing workload; C3 specifies that MEC's processing capacity matches the combined workload offloaded by all devices; C4 represents the range of local computational resources available for the device; C5 represents the range of transmission power available for the device; C6 ensures the threshold constraint of all task queues.

4. Problem conversion and algorithm exploit

4.1. Problem conversion via Lyapunov optimization

Addressing **P1** poses a challenge as it necessitates the acquisition of parameters for every time slot indefinitely in advance. To tackle it, we employ Lyapunov optimization [15], a stochastic optimization technique that allows for the decoupling of spatiotemporally coupled variables and enables the attainment of asymptotically optimal solutions. The core idea involves transforming **P1** into a deterministic optimization problem, to minimize the upper bound of the Lyapunov *drift-plus-penalty* function for each time slot.

Before employing Lyapunov optimization techniques, it is necessary to design a virtual queue Q under the constraint of queue backlog [31], that is

$$Q_i(t+1) = \max\{Q_i(t) + q_i(t+1) - \rho_i, 0\}, \tag{11}$$

(11) shows the deviation of $q_i(t)$ from ρ_i at the conclusion of time slot t. According to [31], we assume that $Q_i(0) = 0$ for all devices. The queue backlog of $Q_i(t)$ grows when the length of $q_i(t)$ surpasses the threshold ρ_i at time slot t, and vice versa. Consequently, (8) representing the long-term queue length constraint can be translated into a stability requirement for the virtual queue backlog $Q_i(t)$.

Next, we utilize Lyapunov optimization method [15] for problem transformation. Denoting $\Theta(t) \triangleq \{Q_1(t), \ldots, Q_N(t)\}$, we employ a quadratic Lyapunov function to represent the fulfillment of C6, which defined as

$$\boldsymbol{Z}(\Theta(t)) \triangleq \frac{1}{2} \sum_{i=1}^{N} Q_i^2(t).$$

Using the quadratic Lyapunov functions from successive time slots, we establish a one-slot Lyapunov drift function to capture queue variations between them, which formulated as

$$\Delta(\Theta(t)) \triangleq \mathbb{E}\left[\boldsymbol{Z}(\Theta(t+1)) - \boldsymbol{Z}(\Theta(t)) | \Theta(t)\right].$$

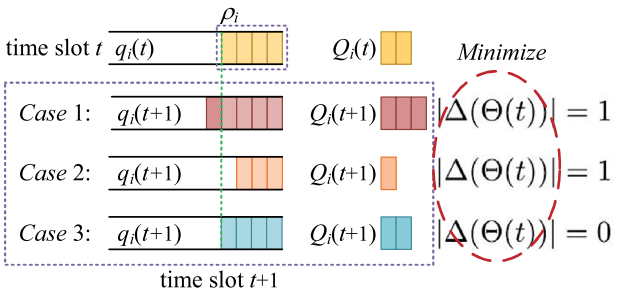


Fig. 2. Conversion of virtual queue Q.

By leveraging the virtual queue Q, this study aims to design an efficient online offloading approach that maintains the stability of the virtual queue, which is shown in Fig. 2. For this purpose, we define the Lyapunov *drift plus penalty* function as $\Delta(\Theta(t))+Ve(t)$ by merging virtual queue stability and energy consumption [32]. This setup implies that the task offloading decision for slot t continues to rely on information from the subsequent slot t+1. Consequently, we introduce Theorem 1 to address this, offering an upper bound for the *drift-plus-penalty* function.

Theorem 1. *If A_i^t is limited to $A_i^{\max}$ across time slots, the drift-plus-penalty value satisfies*

$$\Delta(\Theta(t)) + Ve(t) \le B_1 + B_2 + Ve(t) + Y(t), \tag{12}$$

for any task offloading algorithm. Here,

$$Y(t) = \sum_{i=1}^{N}\left(\frac{\left(D_i^t\right)^2}{2} + \left(\rho_i - q_i(t) - A_i^t - Q_i(t)\right) D_i^t\right),$$

$$B_1 = \frac{1}{2}\sum_{i=1}^{N}\left(\left(A_i^{\max}\right)^2 + \left(\rho_i\right)^2\right),$$

and

$$B_2 = \sum_{i=1}^{N}\left(\frac{1}{2}q_i^2(t) + A_i^{\max}q_i(t)\right) + \sum_{i=1}^{N} Q_i(t)\left(q_i(t) + A_i^{\max} - \rho_i\right).$$

Proof. Taking the square of (11), we have

$$\begin{aligned} &Q_i(t+1)^2 - Q_i(t)^2 \\ &\le (Q_i(t) + q_i(t+1) - \rho_i)^2 - Q_i(t)^2 \\ &= (q_i(t+1) - \rho_i)^2 + 2Q_i(t)(q_i(t+1) - \rho_i) \\ &= q_i(t+1)^2 - 2q_i(t+1)\rho_i + 2Q_i(t)q_i(t+1) \\ &\quad + (\rho_i)^2 - 2Q_i(t)\rho_i. \end{aligned} \tag{13}$$

Taking square on (7), we have

$$\begin{aligned} &q_i(t+1)^2 \le (q_i(t) - D_i^t + A_i^t)^2 \\ &= \left(q_i(t) - D_i^t\right)^2 + \left(A_i^t\right)^2 + 2A_i^t(q_i(t) - D_i^t) \\ &= q_i(t)^2 - 2q_i(t)D_i^t + \left(D_i^t\right)^2 + \left(A_i^t\right)^2 \\ &\quad + 2A_i^t q_i(t) - 2A_i^t D_i^t. \end{aligned} \tag{14}$$

In addition, because $D_i^t \le q_i(t) + A_i^t$, we can also obtain

$$-2q_i(t+1)\rho_i = 2D_i^t\rho_i - 2\rho_i(q_i(t) + A_i^t), \tag{15}$$

and

$$2Q_i(t)q_i(t+1) = 2Q_i(t)(q_i(t) + A_i^t) - 2Q_i(t)D_i^t. \tag{16}$$

Substituting (14), (15), (16) to (13), we have

$$\begin{aligned} &Q_i(t+1)^2 - Q_i(t)^2 \\ &\le \left(D_i^t\right)^2 + 2\left(\rho_i - q_i(t) - A_i^t - Q_i(t)\right) D_i^t \\ &\quad + q_i(t)^2 + \left(A_i^t\right)^2 + 2A_i(t)q_i(t) - 2\rho_i\left(q_i(t) + A_i^t\right) \\ &\quad + 2Q_i(t)\left(q_i(t) + A_i(t) - \rho_i\right) + (\rho_i)^2, \\ &\le \left(D_i^t\right)^2 + 2\left(\rho_i - q_i(t) - A_i(t) - Q_i(t)\right) D_i^t \\ &\quad + q_i(t)^2 + (A_i^{\max})^2 + 2q_i(t)A_i^{\max} + (\rho_i)^2 \\ &\quad + 2Q_i(t)(q_i(t) + A_i^{\max} - \rho_i). \end{aligned}$$

Let

$$B_1 = \frac{1}{2}\sum_{i=1}^{N}\left((A_i^{\max})^2 + (\rho_i)^2\right),$$

and

$$B_2 = \sum_{i=1}^{N}\left(\frac{1}{2}q_i^2(t) + A_i^{\max}q_i(t)\right) + \sum_{i=1}^{N} Q_i(t)\left(q_i(t) + A_i^{\max} - \rho_i\right).$$

The objective function (12) can be upper bounded as

$$\begin{aligned} &\Delta(\Theta(t)) + Ve(t) \le B_1 + B_2 + Ve(t) \\ &\quad + \sum_{i=1}^{N}\left(\frac{\left(D_i^t\right)^2}{2} + \left(\rho_i - q_i(t) - A_i^t - Q_i(t)\right) D_i^t\right). \end{aligned}$$

The theorem is proven. ■

Now, in each time slot, **P1** is transformed into **P2** by eliminating the constant components (B_1 and B_2).

$$\begin{aligned} &\mathbf{P2}: \min_{\boldsymbol{\pi}^t} \; Ve(t) + Y(t) \\ &\text{s.t. } C1 - C5 \end{aligned} \tag{17}$$

The optimal solution to **P2** exhibits asymptotically optimum properties according to the Lyapunov optimization theory [15].

4.2. DDPG-based algorithm design

Clearly, **P2** resembles a MINOP as previously introduced in Ref. [22]. Given the non-convex nature of the optimization objective function and the mixed continuous-discrete variables in the proposed problem, solving **P2** directly is an exceedingly challenging task. Furthermore, utilizing iterative numerical methods to solve **P2** will result in excessively high computational complexity. Accordingly, we develop an online offloading algorithm that leverages a DRL approach, specifically the DDPG technique [33], to achieve the asymptotically optimal solution for the given problem.

4.2.1. Markov decision process of **P2**

In each time slot, **P2** may also be characterized as a Markov Decision Process (MDP) with the following definitions for the states, actions, and reward function [34].

- State space $\mathcal{S}$: The state set includes the task arrival rates $\boldsymbol{A}^t = \{A_i^t\}_{i=1}^N$, the task queue lengths $\boldsymbol{q}^t = \{q_i^t\}_{i=1}^N$, the queue thresholds $\boldsymbol{q}^c = \{\rho_i\}_{i=1}^N$, the virtual queue lengths $\boldsymbol{Q}^t = \{Q_i^t\}_{i=1}^N$, and the channel conditions $\boldsymbol{h}^t = \{h_i^t\}_{i=1}^N$. We express it as $s_t = \left\{\boldsymbol{A}^t, \boldsymbol{q}^t, \boldsymbol{q}^c, \boldsymbol{Q}^t, \boldsymbol{h}^t\right\}$, where $s_t \in \mathcal{S}$.
- Action space $\mathcal{A}$: The action set includes the remote offloading decision, local computing resource and communication power allocation, and MEC computing resource distribution, i.e., $a_t = \left\{\boldsymbol{x}^t, \boldsymbol{f}^t, \boldsymbol{p}^t, f_{\mathrm{m}}^t\right\}$, where $a_t \in \mathcal{A}$. is action space.
- The reward function corresponds to the negative of **P2**'s objective, i.e., $r_t = -Ve(t) - Y(t)$.

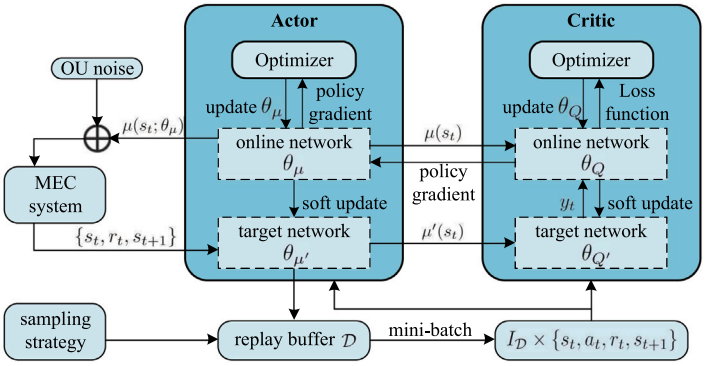


Fig. 3. The DDPG framework.

4.2.2. DDPG-based algorithm

The algorithm we propose is an extension of the widely recognized DDPG technique and operates on the MEC. At the start of each time slot, MEC gathers device state information and subsequently transmits the offloading policy to the devices. As the DDPG algorithm follows an actor-critic architecture, our algorithm similarly comprises an actor-network and a critic-network. The actor-network decides the real-time action policy based on the present state information. Considering the current state, the critic-network evaluates the action policies provided by the actor-network, i.e., Q-values, which determines whether the action strategy is good or bad. The pseudo-code for the proposed algorithm is displayed in Algorithm 1, operating in a time-slotted manner and involving four deep neural networks.

The actor-network makes actions on remote offloading decisions, local computing resources and communication power allocation, and MEC computing resource allocation based on s_t, and adds Ornstein–Uhlenbeck (OU) noise. Then, the system executes $\mathcal{A}_t$, engaging with the environment to acquire r_t and the subsequent state s_{t+1}. Finally, the set $\{s_t, a_t, r_t, s_{t+1}\}$ is formed into a tensor, which is used to update the network parameters stored in the experience replay memory $\mathcal{D}$. Fig. 3 depicted the detailed operational structure.

To cut off the relevance of experience during the network learning process, LyD2OA randomly selects I training samples $\{s_t, a_t, r_t, s_{t+1}\}$ from $\mathcal{D}$. Denote $\mu(s;\theta_\mu)$ and $Q(s,a;\theta_Q)$ as the current actor and critic networks with weights θ_μ and θ_Q, respectively. Likewise, $\mu'(s;\theta_{\mu'})$ and $Q'(s,a;\theta_{Q'})$ correspond to the target actor and critic networks with weights $\theta_{\mu'}$ and $\theta_{Q'}$, respectively. Hence, the target Q-value is defined as

$$y_t = r_t + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1};\theta_{\mu'}\right);\theta_{Q'}\right), \tag{18}$$

where $\gamma \in [0,1]$ is the discount factor. Thus, the loss function is defined as

$$\mathcal{L}\left(\theta_Q\right) = \mathbb{E}\left[y_t - Q\left(s_t, a_t;\theta_Q\right)\right]^2. \tag{19}$$

Notice that $\mathcal{L}\left(\theta_Q\right)$ is the loss function of critic's current network. Instead, the actor's current network strives to achieve a higher Q-value wherever feasible. Consequently, the policy objective function of the actor's current network can be expressed by

$$J\left(\theta_\mu\right) = \mathbb{E}\left[Q\left(s_t, \mu(s_t;\theta_\mu);\theta_Q\right)\right]. \tag{20}$$

Thus, we employ the gradient descent method to modify the weights θ_μ and θ_Q of the current actor and networks. Furthermore, to enhance the learning process's stability, we employ soft target updating to ensure gradual updates of the network weight parameters, which is,

$$\theta_{Q'} = \alpha\theta_Q + (1-\alpha)\theta_{Q'}, \tag{21a}$$

$$\theta_{\mu'} = \alpha\theta_\mu + (1-\alpha)\theta_{\mu'}. \tag{21b}$$

Algorithm 1: The LyD2OA Algorithm

Input: System environment parameters;
Output: Optimal offloading strategy $\boldsymbol{\pi}^*$;
Randomly initialize the weights θ_Q and θ_μ of the current network;
Initialize experience replay memory $\mathcal{D}$ and the target network's weights $\theta_{Q'} \leftarrow \theta_Q, \theta_{\mu'} \leftarrow \theta_\mu$;
for ***each*** *episode* **do**
  Reset the system environment, and initialize the task and virtual queue $q_i(t)$ and $Q_i(t)$ according to (7) and (11) for each device i;
  for $t = 1$ *to* T **do**
    Obtain $a_t = \{\boldsymbol{x}^t, \boldsymbol{f}^t, \boldsymbol{p}^t, f_{\mathrm{m}}^t\}$ according to the current actor network and OU noise;
    Send $\{\boldsymbol{x}^t, \boldsymbol{f}^t, \boldsymbol{p}^t\}$ to each device, and run it;
    Run f_{m}^t on the MEC;
    Observe the reward r_t and the next time slot's state s_{t+1};
    Store the tensor s_t, a_t, r_t, s_{t+1} in $\mathcal{D}$;
    Sample a random minibatch of I tensors from $\mathcal{D}$;
    Update the current networks of the critic and actor based on (19) and (20) with the learning rate l_c and l_a, respectively;
    Update the target networks of the critic and actor according to (21a) and (21b);
  end
end

4.3. LyD2OA computational complexity analysis

Given that the actor and critic networks comprise fully connected layers, the computational complexity of LyD2OA in time slot t is solely determined by the network's neuron count. Let the actor network have Φ fully connected layers and the critic network have Ψ fully connected layers. Therefore, The computational complexity of the algorithm we propose is similar to [25], denoting a complexity of $O(\sum_{i=0}^{\Phi} fcl_i^a \cdot fcl_{i+1}^a + \sum_{j=0}^{\Psi} fcl_j^c \cdot fcl_{j+1}^c)$. Here, the number of neurons in the actor network's ith layer is represented by the constant fcl_i^a $(i > 0)$; the number of neurons in the critic network's j th layer is represented by the constant fcl_j^c $(j > 0)$. fcl_0^a and fcl_0^c is the input size of two networks. Regarding performance analysis for LyD20A, an in-depth proof methodology has been provided in [27], and we will refrain from duplicating it here.

5. Results analysis for Ly2DOA

This section presents a sequence of experiments designed to assess the performance of LyD2OA. We evaluate the system's efficiency by

Table 1
Experimental simulation parameters.

Parameters	Value	Parameters	Value
N_0	−174 dBm/Hz	episode	100
λ	[0.8, 1.5] Mbit/s	T	500
$f_m(t)$	$[0, 2\times10^4]$ MHz	γ	0.99
$f_i(t)$	[0, 500] MHz	α	0.001
$p_i(t)$	[0, 500] mW	l_a	10^{-4}
χ	10^{-10} mW	l_c	10^{-3}
ρ_i	32 Mbit	D	64
B	1 MHz	I	8
ω_0	10 μW	N	10
ξ	10^{-26}	ϕ	1000
ζ	10^{-28}	ψ	1000

Table 2
Task processing delay (s).

	$\rho=30$	$\rho=35$	$\rho=40$	$\rho=45$	$\rho=50$
Device 1	29.2668	38.3471	43.4299	44.9710	47.4126
Device 2	30.6943	40.6263	41.1090	49.9282	52.4134
Device 3	31.9883	37.4893	43.9698	46.1077	47.0206
Device 4	29.1149	35.3380	40.7044	42.8673	48.5008
Device 5	32.1484	34.5065	39.1554	48.7956	50.0020
Device 6	30.9905	34.1802	44.5535	46.0622	57.7057
Device 7	31.0879	34.4146	45.0259	46.9513	54.3541
Device 8	28.9552	35.0359	39.8948	45.7691	50.3947
Device 9	28.0375	36.2060	42.4484	46.8214	50.5976
Device 10	34.4007	35.9845	39.1025	47.1505	56.3855
Average	30.6685	36.2129	41.9394	46.5424	51.4787

adjusting various parameters, including task arrival rates, which signify the computational load and are usually quantified by the data volume within each task. This reflects real-world manufacturing scenarios where varying amounts of data are collected. Additionally, we examine the system's robustness by setting different task queue thresholds.

5.1. Experiment and parameter settings

In LyD2OA, the actor network is deployed as a five-layer Fully Connected Neural Network (FCNN) with $[5N, 800, 400, 200, 3N + 1]$ neurons, and the critic network is deployed as a five-layer FCNN with $[801 + 8N, 800, 300, 10, 1]$ neurons. The actor network employs the *tanh* activation function in its output layer, while the *Relu* activation function is used in all other layers of the actor network as well as in all layers of the critic network. The Adam optimizer is responsible for updating all model parameters.

In terms of the environmental parameter configuration, the task arrival rate of device i in each time slot follows an exponential distribution, i.e., $A_i^t \sim \mathrm{E}(\lambda)$ Mbit. h_i^t refers the channel power gain model in [22], and time slot length $\tau = 1$. The simulation experiments make use of the default parameter values, as detailed in Table 1, with the majority of these values sourced from [26,30].

5.2. Performance evaluation and comparative schemes

5.2.1. Convergence performance and energy-delay tradeoff

In this group of experiments, given the tasks arrival rate $\lambda = 1$ Mbit/s, we plot the results of four trainings to illustrate the convergence performance of LyD2OA, and the average virtual queue length and the system energy consumption under different values of ν. The queue threshold ρ_i is set to 32. Here, $\nu = V / \max_{\pi^t} Y(t)$, and the reward function is converted to

$$r_i' = e(t) + \frac{Y(t)}{\nu \times \max_{\pi^t} Y(t)}.$$

Therefore, the value range of the reward function is scaled to a reasonable order of magnitude, which is conducive to updating the weight of DNN [33].

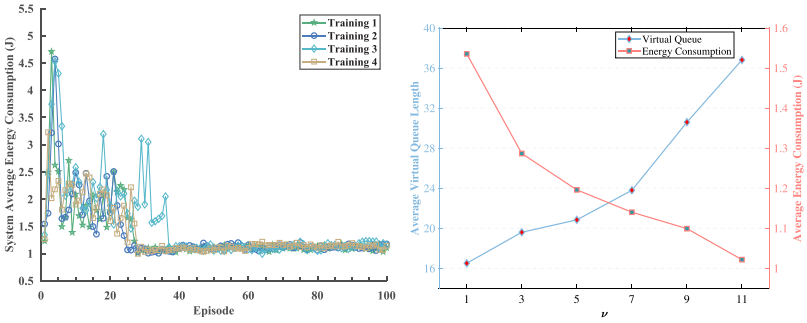
The four curves in Fig. 4(a) represent the average energy consumption per episode during the training phase. From Fig. 4(a), It is evident that during the initial phases of training, there is a significant energy consumption fluctuation in each episode. This is primarily due to the agent's actor and critic network parameters being initialized with random values at the outset. As training advances, the parameters are iteratively fine-tuned to minimize the system's energy consumption. During the latter phases of training, the energy consumption per episode gradually reaches a stable state, signifying that after about 40 episodes, the Q-value obtained by the agent also stabilizes. Fig. 4(b) illustrates a trend where energy consumption experiences an initial rapid decrease, followed by a gradual stabilization, while the queue length grows linearly with ν.

5.2.2. Effect of the queue threshold ρ

In this set of experiments, we plot the system queue lengths and energy consumption under different queue thresholds. Additionally, we calculated the task processing delays for each device after the queue stabilized. As shown in Fig. 5(a), the LyD2OA algorithm effectively controls the system queue length. After 100 time slots, LyD2OA ensures that the task queue remains stable under all conditions. Under various queue thresholds, the system queue length rapidly converges to near the threshold values. Table 2 presents the task processing delays for devices after the queue stabilization, with the last row indicating the average processing delay. From Table 2, it is evident that there is a direct correlation between the device queue length and the task processing delay. Longer device queue lengths result in longer waiting times for task processing. By strictly controlling the task queue length, the average waiting time for tasks can be reduced. Fig. 5(b) shows that under different queue thresholds, the system's energy consumption fluctuates within specific intervals. It is noteworthy that in the first 100 time slots, the system energy consumption appears to rise briefly and then decreases as the slots progress. This is due to the DNN's inability to establish an optimal offloading strategy based on the initial environment. As the agent continues to interact with the environment, the DNN gradually finds the optimal offloading strategy and stabilizes the queue near the threshold.

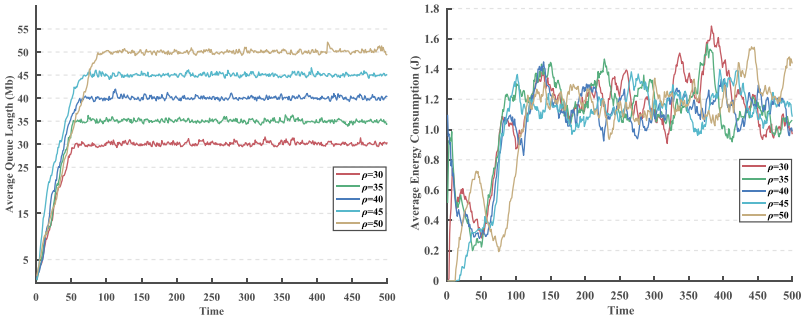
5.2.3. Effect of the task arrival rate λ

Under different task arrival rates, we depict the dynamic trends of task queues and virtual queues. Fig. 6(a) gives the average length of the queue, presented as a bar chart. From Fig. 6(a), we can observe that with the growth in the task arrival rate, both the task queue and virtual queue length of the system expand. When $\lambda = 0.8$, the average queue length of the system is less than ρ. Therefore, the virtual queue length is very small. Besides, with the increase in task arrival rate, the queue length gradually surpasses the threshold constraint. This is because the number of arriving tasks per unit of time slot gradually increases and is temporarily stored in the tail of the queue, while the system cannot process the tasks in the head of the queue in time. Thus the queue length starts to increase slowly, leading to a slow increase in the size of the virtual queue. Fig. 6(b) gives the trend of the queue under different task arrival rates, presented as a graph. From Fig. 6(b), we know that the task queue length gradually starts to be unstable as the task arrival rate increases. The queue length cannot be effectively constrained around the threshold value. When $\lambda = 1.5$, the task queue length fluctuates with large ups and downs and cannot be well constrained. This is because the number of arriving tasks is approaching the critical value of the system's processing capacity, and the system cannot handle the upcoming tasks within the designated time frame.



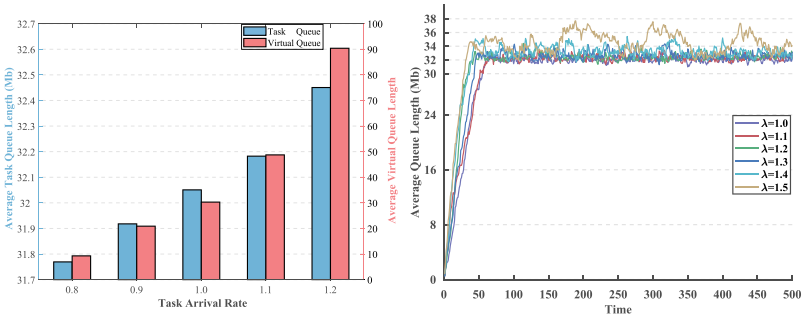
(a) Energy consumption per episode in the training phase (b) Impact of the different parameter ν

Fig. 4. Convergence performance and energy-delay tradeoff.



(a) Queue length with different ρ (b) Energy consumption with different ρ

Fig. 5. Impact of the different parameter ρ.



(a) Task and virtual queue length with different λ ($\lambda = 0.8 \sim 1.2$) (b) Task and virtual queue length with different λ ($\lambda = 1.0 \sim 1.5$)

Fig. 6. Impact of the different parameter λ.

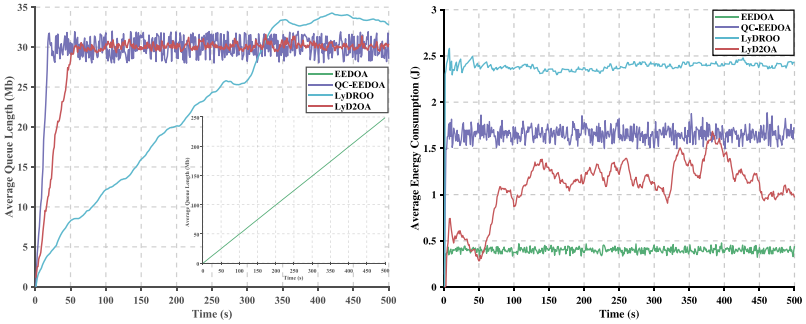
5.2.4. Comparison of different offloading schemes

We compare LyD2OA with two other algorithms to assess the practical effectiveness of the algorithm.

- *Lyapunov-guided DRL for Online Computation Offloading* (LyDROO) [22]
  In each time slot, all tasks follow a binary offloading principle. Utilizing the Lyapunov optimization method, task offloading decisions are determined in real-time under the condition of maintaining stability of both the task and energy queues, thereby maximizing the computation rate of tasks.
- *Energy Efficient Dynamic Offloading Algorithm with Queue Length Constraint* (QC-EEDOA) [27]
  In each slot, all tasks adhere to the principle of partial offloading. By employing the Lyapunov optimization method to establish virtual queues, the constraint of task backlog thresholds is transformed into a stability constraint for the virtual queue. The knapsack principle is utilized to dynamically determine task offloading decisions while maintaining the stability of the virtual queue.
  Energy Efficient Dynamic Offloading Algorithm (EEDOA) [35]
  In each time slot, all tasks are offloaded to MEC. By applying Lyapunov optimization, the offloading decision is generated in each time slot to maintain system stability.

In this experiment, the parameters for each algorithm were kept consistent with those of LyD2OA, with the experimental data for LyD2OA corresponding to $\rho = 30$ in Fig. 5.

Fig. 7(a) presents the queue length of the four algorithms. From Fig. 7(a), it is evident that the LyD2OA algorithm's queue length quickly converges to the vicinity of the threshold, whereas the EEDOA



(a) Task queue length with different algorithms (b) Energy consumption with different algorithms

Fig. 7. The comparison of queue length and energy consumption of different algorithms.

algorithm fails to effectively control queue length in this scenario due to its sole reliance on remote offloading without considering local processing. For LyDROO, the queue length gradually increases, as this approach only aims to stabilize the task queue length without imposing strict constraints on it. For QC-EEDOA, although the queue length is effectively maintained near the threshold, the fluctuations are considerable. This is attributed to the QC-EEDOA algorithm's use of the knapsack problem for solving, which cannot learn from and adapt to changes in the environment, thereby failing to efficiently control the variability of the queue. In contrast, DRL methods can learn optimal decisions amidst uncertainty by exploring different strategies, thereby further constraining the fluctuation of task queues within the threshold limits.

Fig. 7(b) presents the energy consumption of the four algorithms. It is observed that EEDOA consumes the least energy because it focuses solely on remote offloading, neglecting local offloading. LyDROO exhibits the highest energy consumption since it is a binary offloading algorithm optimized for maximizing the offloading rate. When channel conditions are favorable, more tasks are offloaded to the MEC server, inevitably increasing the server's power demand and thus elevating the average energy consumption of the system. Under poor channel conditions, the algorithm attempts to process more tasks to maximize the offloading rate, which results in increased energy usage. For QC-EEDOA, while the average system energy consumption remains relatively stable, this algorithm does not manage the operational power of the MEC servers. In poor channel conditions, it fails to reduce server operations and minimize unnecessary energy expenditure. In contrast, LyD2OA dynamically generates offloading policies by thoroughly considering channel conditions, queue sizes, and system energy consumption. Regardless of the task arrival rate, LyD2OA ensures that the queue size remains close to the threshold while optimizing the overall system energy consumption. Based on observations from Fig. 7(a) and (b), we can infer that LyD2OA performs well in optimizing system energy consumption and constraining queue lengths.

6. Conclusion

In this paper, we propose a novel Lyapunov-guided DRL for the delay-aware offloading algorithm (LyD2OA) by considering the dynamic characteristics of IoT systems. This algorithm can make real-time decisions for system offloading, adapting dynamically to offload tasks with better network quality. Meanwhile, it ensures that deadlines are not violated when offloading tasks in poor communication environments. Relevant simulation experiments indicate the algorithm's capacity to control long-term energy consumption and average task queue length within the system through adjustments of parameter V. Comparative analysis against alternative offloading schemes reveals that LyD2OA effectively minimizes long-term energy consumption while maintaining low latency, making it a suitable choice for addressing DSCI-type tasks.

In future work, we will explore MEC systems with multiple edge servers, addressing challenges related to limited computational resources and devising an appropriate resource allocation scheme. In other words, offloading will be delayed only when the communication conditions are poor to minimize the transmission delay of the tasks and avoid violating the deadlines for task processing.

CRediT authorship contribution statement

Longbao Dai: Writing – review & editing, Writing – original draft, Resources, Methodology. **Jing Mei:** Writing – review & editing, Methodology, Investigation, Funding acquisition, Formal analysis. **Zhibang Yang:** Methodology, Funding acquisition, Formal analysis, Conceptualization. **Zhao Tong:** Writing – review & editing, Funding acquisition, Formal analysis, Conceptualization. **Cuibin Zeng:** Writing – review & editing, Investigation, Conceptualization. **Keqin Li:** Writing – review & editing, Conceptualization.

Declaration of competing interest

We wish to draw the attention of the Editor to the following facts which may be considered as potential conflicts of interest and to significant financial contributions to this work. [OR]

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

We further confirm that any aspect of the work covered in this manuscript that has involved either experimental animals or human patients has been conducted with the ethical approval of all relevant bodies and that such approvals are acknowledged within the manuscript.

We understand that the Corresponding Author is the sole contact for the Editorial process (including Editorial Manager and direct communications with the office). He/she is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs.

Data availability

The authors are unable or have chosen not to specify which data has been used.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the Program of National Natural Science Foundation of China (grant No. 62072174, 61502165), Provincial Natural Science Foundation of Hunan, China (grant No. 2020JJ5370, 2022JJ40278, 2023JJ30083), Scientific Research Fund of Hunan Provincial Education Department, China (Grant No. 22A0026, 22A0592).

References

[1] H. Duan, Y. Zheng, C. Wang, X. Yuan, Treasure collection on foggy islands: Building secure network archives for internet of things, IEEE Internet Things J. 6 (2) (2019) 2637–2650.
[2] H.R. Chi, M.F. Domingues, A. Radwan, Complex network analysis for ultra-large-scale mec small-cell based peer-offloading, in: 2021 IEEE Global Communications Conference, GLOBECOM, 2021, pp. 1–6.
[3] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, Proc. IEEE 107 (8) (2019) 1738–1762.
[4] H. Zhao, S. Deng, C. Zhang, W. Du, Q. He, J. Yin, A mobility-aware cross-edge computation offloading framework for partitionable applications, in: 2019 IEEE International Conference on Web Services, ICWS, 2019, pp. 193–200.
[5] X. Xu, B. Shen, X. Yin, M.R. Khosravi, H. Wu, L. Qi, S. Wan, Edge server quantification and placement for offloading social media services in industrial cognitive iov, IEEE Trans. Ind. Inform. 17 (4) (2021) 2910–2918.
[6] G. Qu, N. Cui, H. Wu, R. Li, Y. Ding, Chainfl: A simulation platform for joint federated learning and blockchain in edge/cloud computing environments, IEEE Trans. Ind. Inform. 18 (5) (2022) 3572–3581.
[7] H. Tang, H. Wu, Y. Zhao, R. Li, Joint computation offloading and resource allocation under task-overflowed situations in mobile-edge computing, IEEE Trans. Netw. Serv. Manag. 19 (2) (2022) 1539–1553.
[8] H. Wu, Y. Sun, K. Wolter, Energy-efficient decision making for mobile cloud offloading, IEEE Trans. Cloud Comput. 8 (2) (2020) 570–584.
[9] A. Islam, A. Debnath, M. Ghose, S. Chakraborty, A survey on task offloading in multi-access edge computing, J. Syst. Archit. 118 (2021) 102225.
[10] H. Wu, J. Chen, T.N. Nguyen, H. Tang, Lyapunov-guided delay-aware energy efficient offloading in iiot-mec systems, IEEE Trans. Ind. Inform. 19 (2) (2023) 2117–2128.
[11] J. Chen, H. Xing, X. Lin, S. Bi, Joint cache placement and bandwidth allocation for fdma-based mobile edge computing systems, in: ICC 2020-2020 IEEE International Conference on Communications, ICC, IEEE, 2020, pp. 1–7.
[12] Z. Chang, L. Liu, X. Guo, Q. Sheng, Dynamic resource allocation and computation offloading for iot fog computing system, IEEE Trans. Ind. Inform. 17 (5) (2020) 3348–3357.
[13] Z. Tong, J. Wang, J. Mei, K. Li, K. Li, Fedto: Mobile-aware task offloading in multi-base station collaborative mec, IEEE Trans. Veh. Technol. (2023).
[14] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, L. Guo, Computation offloading in mobile edge computing networks: A survey, J. Netw. Comput. Appl. 202 (2022) 103366.
[15] M. Neely, Stochastic Network Optimization with Application to Communication and Queueing Systems, Morgan & Claypool Publishers, 2010.
[16] S. Xia, Z. Yao, Y. Li, S. Mao, Online distributed offloading and computing resource management with energy harvesting for heterogeneous mec-enabled iot, IEEE Trans. Wireless Commun. 20 (10) (2021) 6743–6757.
[17] Y. Li, S. Xia, M. Zheng, B. Cao, Q. Liu, Lyapunov optimization-based trade-off policy for mobile cloud offloading in heterogeneous wireless networks, IEEE Trans. Cloud Comput. 10 (1) (2022) 491–505.
[18] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, X. Wang, Learning-aided computation offloading for trusted collaborative mobile edge computing, IEEE Trans. Mob. Comput. 19 (12) (2020) 2833–2849.
[19] X. Gao, X. Huang, S. Bian, Z. Shao, Y. Yang, Pora: Predictive offloading and resource allocation in dynamic fog computing systems, IEEE Internet Things J. 7 (1) (2020) 72–87.
[20] X. Zhang, A. Pal, S. Debroy, Effect: Energy-efficient fog computing framework for real-time video processing, in: 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing, CCGrid, IEEE, 2021, pp. 493–503.
[21] J. Du, C. Jiang, J. Wang, Y. Ren, M. Debbah, Machine learning for 6 g wireless networks: Carrying forward enhanced bandwidth, massive access, and ultrareliable/low-latency service, IEEE Veh. Technol. Mag. 15 (4) (2020) 122–134.
[22] S. Bi, L. Huang, H. Wang, Y.-J.A. Zhang, Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks, IEEE Trans. Wireless Commun. 20 (11) (2021) 7519–7537.
[23] Y. Jia, C. Zhang, Y. Huang, W. Zhang, Lyapunov optimization based mobile edge computing for internet of vehicles systems, IEEE Trans. Commun. 70 (11) (2022) 7418–7433.
[24] X. Zhang, M. Mounesan, S. Debroy, Effect-dnn: Energy-efficient edge framework for real-time dnn inference, in: 2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM, IEEE, 2023, pp. 10–20.
[25] M. Gao, R. Shen, L. Shi, W. Qi, J. Li, Y. Li, Task partitioning and offloading in dnn-task enabled mobile edge computing networks, IEEE Trans. Mob. Comput. 22 (4) (2023) 2435–2445.
[26] W. Fan, Z. Chen, Z. Hao, Y. Su, F. Wu, B. Tang, Y. Liu, Dnn deployment, task offloading, and resource allocation for joint task inference in iiot, IEEE Trans. Ind. Inform. 19 (2) (2023) 1634–1646.
[27] J. Mei, L. Dai, Z. Tong, L. Zhang, K. Li, Lyapunov optimized energy-efficient dynamic offloading with queue length constraints, J. Syst. Archit. 143 (2023) 102979.
[28] J. Mei, L. Dai, Z. Tong, X. Deng, K. Li, Throughput-aware dynamic task offloading under resource constant for mec with energy harvesting devices, IEEE Trans. Netw. Serv. Manag. 20 (3) (2023) 3460–3473.
[29] S. Xia, Z. Yao, G. Wu, Y. Li, Distributed offloading for cooperative intelligent transportation under heterogeneous networks, IEEE Trans. Intell. Transp. Syst. 23 (9) (2022) 16701–16714.
[30] Y. Ye, L. Shi, X. Chu, R.Q. Hu, G. Lu, Resource allocation in backscatter-assisted wireless powered mec networks with limited mec computation capacity, IEEE Trans. Wireless Commun. 21 (12) (2022) 10678–10694.
[31] C. Qiu, Y. Hu, Y. Chen, Lyapunov optimized cooperative communications with stochastic energy harvesting relay, IEEE Internet Things J. 5 (2) (2018) 1323–1333.
[32] M.J. Neely, et al., Stability and probability 1 convergence for queueing networks via lyapunov optimization, J. Appl. Math. 2012 (2012).
[33] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv: 1509.02971.
[34] S.P. Meyn, R.L. Tweedie, Markov Chains and Stochastic Stability, Springer Science & Business Media, 2012.
[35] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, X. Shen, Energy efficient dynamic offloading in mobile edge computing for internet of things, IEEE Trans. Cloud Comput. 9 (3) (2019) 1050–1060.

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

Jing Mei received the Ph.D. degree in computer science from Hunan University, China, in 2015. She is currently an associate professor in the College of Information Science and Engineering in Hunan Normal University. Her research interests include cloud computing, fog computing and mobile edge computing, high performance computing, task scheduling and resource management, etc. She has published more than 30 research articles in international conference and journals, such as *IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed System, IEEE Transactions on Service Computing, Cluster Computing, Journal of Grid Computing, Journal of Supercomputing.*

Zhibang Yang received the Ph.D. degree in computer science from Hunan University, China. His major research contains data management, parallel computing and network security. He is currently a professor at Changsha University.

Zhao Tong received the Ph.D. degree in computer science from Hunan University, Changsha, China in 2014. He was a visiting scholar at the Georgia State University from 2017 to 2018. He is currently an associate professor at the College of Information Science and Engineering of Hunan Normal University, the young backbone teacher of Hunan Province, China. His research interests include parallel and distributed computing systems, resource management, big data and machine learning algorithm. He has published more than 25 research papers in international conferences and journals, such as IEEE-TPDS, Information Sciences, FGCS, NCA, and JPDC, PDCAT, etc. He is a senior member of the China Computer Federation (CCF) and a Member of the IEEE.

Cuibin Zeng received the B.S. degree in computer science and technology from Jishou University, Jishou, China, in 2022. He is currently pursuing the M.S. degree at the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research focuses on resource scheduling and price allocation in mobile edge computing.

Keqin Li is a SUNY Distinguished Professor of Computer Science with the State University of New York. He is also a National Distinguished Professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has authored or co-authored over 850 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top 5 most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an associate editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Services Computing*, and the *IEEE Transactions on Sustainable Computing*. He is an IEEE Fellow and an AAIA Fellow. He is also a Member of Academia Europaea (Academician of the Academy of Europe).