Efficient Seamless Task Offloading based on Edge-Terminal Collaborative for AIoT Elastic Computing Services

Jing Wang, Yuhuai Peng, Xinyu Zhang, Lei Liu, *Member, IEEE,* Shahid Mumtaz, *Senior Member, IEEE,* Mohsen Guizani, *Fellow, IEEE,* Schahram Dustdar, *Fellow, IEEE*

***Abstract*—Artificial Intelligence of Things (AIoT) utilizes a combination of computing, storage, and networking resources to provide highly reliable and low-latency information services to the industrial production processes. However, with the increasing integration of numerous smart terminals into real-time sensing, autonomous decision-making, and precision manufacturing execution systems, the current task scheduling pattern appears to be insufficient to meet the latency requirements of computationally intensive tasks. To address the above challenge, this paper presents a collaborative edge-terminal task offloading scheme. First, the Task Backlog and Multi-slot Scheduling (TBMS) problem is converted from a long-term offloading problem to a single timeslot scheduling problem by Lyapunov optimization. Then, to simplify the problem, the single timeslot problem is decomposed into three subproblems: the local resource allocation problem, the server resource allocation problem, and the indicator weight selection problem. The two resource allocation problems are proved to be convex, which have been solved by using the Bisection method and the Karush-Kuhn-Tucker (KKT) method, respectively. For the indicator weight selection problem, we proposed the enhanced jumping spider optimization algorithm that integrates the elite opposition-based learning strategy. Extensive experiments show that the proposed algorithm can alleviate the computing pressure of the terminal device. Compared with the traditional methods, the offload system cost is effectively reduced by at least 58.8% and the average execution success rate is increased by at least 6%.**

***Index Terms*—artificial intelligence of things; edge-terminal collaboration; resource allocation; long-term task offloading; Lyapunov optimization.**

Corresponding author: Yuhuai Peng.

Jing Wang and Xinyu Zhang are with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: 2110651@stu.neu.edu.cn, zhangxinyu98a@163.com).

Yuhuai Peng is with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, and the Zhiyuan Research Institute, Hangzhou 310024, China (e-mail: pengyuhuai@mail.neu.edu.cn).

Lei Liu is with the Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China, and also with the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China (e-mail: tianjiaoliulei@163.com).

Shahid Mumtaz is with the Department of Applied Informatics, Silesian University of Technology, Akademicka, Gliwice 16 44-100, Poland, and also with the Departement of Engineering, Nottingham Trent University, Nottingham NG1 4FQ, UK (e-mail: shahid.mumtaz@ntu.ac.uk).

Mohsen Guizani is with the Department of Machine Learning, Mo hamed Bin Zayed University of Artificial Intelligence, Masdar, Abu Dhabi, United Arab Emirates (e-mail: mguizani@ieee.org).

Schahram Dustdar is head of the Distributed Systems Research Division of the TU Wien, 1040 Austria and part-time ICREA research professor at UPF, Barcelona (e-mail: dustdar@dsg.tuwien.ac.at).

I. Introduction

The Artificial Intelligence of Things (AIoT) has brought about transformational advances in industrial operational processes through the precise integration of Artificial Intelligence (AI), edge computing, and information and communication technology [1], [2]. By adopting the 'power + computing' network framework, AIoT is enhancing business collaboration, promoting safe and stable industrial operation, and optimizing intelligent management [3], [4]. With the further integration of massive smart terminals into real-time sensing, autonomous decision-making, and precise manufacturing execution systems, and the popularity of Computation Power Networks (CPNs), AIoT significantly enriches the information processing capabilities and intelligence of industry. Constrained by the limited computational capabilities and battery life, Terminal Devices (TDs) face the significant burden from the rapid expansion of demand for fault detection, load forecasting, intelligent inspection, remote control and other computationally intensive tasks [5]. Consequently, there is an urgent need to advance the precision management of computational resources. By leveraging the edge-terminal collaboration to dynamically and automatically coordinate computational, storage and communication resources, the capabilities of AIoT services will be significantly enhanced, leading to more efficient, reliable, and environmentally sustainable industrial production processes.

While edge-terminal collaborative computing technology facilitates the rapid execution of terminal tasks and promotes the development of AIoT, several challenges remain unresolved in industrial production process [6]. Resource allocation and offloading decisions are inherently dynamic and complex, requiring the comprehensive evaluation of task requirements, device computational capabilities, network conditions, and other influencing factors [7]. The edge computing environment, characterized by numerous heterogeneous devices and platforms, further complicates seamless offloading. In addition, most efficient task offloading algorithms have extremely high computational complexity and cannot be deployed on resource-constrained terminals [8]. The computational offloading algorithms executed on these terminals are inefficient, which compromises the success rates and real-time performance of time-sensitive tasks. Therefore, the complex interplay between network states, computational resources and device characteristics poses significant challenges for effectively

planning of long-term task scheduling and seamless offloading.

Various schemes for resource allocation and task offloading have been proposed [9], [10]. Resource allocation methods based on game theory simulate the offloading process by analyzing the resource competition between terminal devices and solving the optimal offloading decisions. The complexity of these methods is influenced by the task volume and the number of channels, making them more suitable for small-scale scenarios. With the development of Artificial Intelligence (AI), resource allocation based on Deep Learning (DL) has attracted considerable attention [11], [12]. These approaches rely on extensive training to achieve optimal offloading decisions, resulting in significant computational overhead due to their complexity. To achieve low-latency and energy-efficient computing services, task offloading methods based on swarm intelligence randomly or strategically alter individual behavior to achieve collective offloading optimization [13]. However, existing solutions have slow convergence rates, which cannot provide high-quality computing services.

To address the long-term task backlog problem in terminal device, we proposed a Lyapunov-based Collaborative Offloading Decision scheme for Edge-Terminal (CODET). The main contributions of this work are as follows:

1) The Task Backlog and Multi-slot Scheduling (TBMS) problem is defined. An upper bound on its drift function is derived, verifying the TBMS problem is transformable. The TBMS is converted from the long-term scheduling problem to the single timeslot offloading problem using Lyapunov optimization. To facilitate a fast solution, the timeslot problem is decomposed into three subproblems: local resource allocation, server resource allocation, and indicator weight optimization.
2) The CODET scheme based on task priority is proposed to obtain the optimal allocation solution. To reduce the complexity of the subproblem, it is proven that the two computational resource allocation subproblems are convex, and they are solved using the Bisection method and the Karush-Kuhn-Tucker (KKT) method, respectively. The enhanced jumping spider optimization algorithm is proposed to optimize the indicator weight.
3) Extensive simulations have shown that the CODET can effectively balance the computation delay and energy consumption, alleviate the offloading pressure on the local terminal. The average system cost is reduced by at least 58.8% and the average task execution success rate is increased by at least 6% compared to the traditional methods.

The remainder of this paper is organized as follows. Related work is reviewed in Section II. The network model is formulated in Section III. In Section IV, the design of the computation offloading scheme is presented in detail. Simulation results and analysis are discussed in Section V. Finally, conclusions are given in Section VI.

II. RELATED WORKS

Luo *et al.* [14] proposed the Game theory-based computation offloading methods [14], equilibrium solutions can be obtained to reduce computational complexity. To minimize the delay and energy consumption of peer-to-peer offloading of multiple satellites, Zhang *et al.* [15] introduced the online distributed computational offloading scheme based on Lyapunov to further improve the resource utilisation. Teymoori *et al.* [16] described the offloading decision process as a stochastic game model, using reinforcement learning to find Nash equilibria, thereby reducing mutual interference during channel access. Zhang *et al.* [17] proposed a mixed integer nonlinear programming (MINLP) based task offloading and resource allocation decision making method to minimize vehicle task latency. However, such approaches require significant computational resources and time. Computation offloading methods based on reinforcement learning or deep learning [18], with neural networks at their core, rely on extensive data training and autonomous learning to provide efficient resource allocation decisions and improve computation offloading efficiency. Qu *et al.* [19] combined multiple parallel deep neural networks with Q-learning to quickly obtain optimal offloading strategies from dynamic environments. Tang *et al.* [20] introduced a distributed algorithm combining Long Short-Term Memory networks, DQN, and DDQN to minimize task dropout rates and reduce average delay. Su *et al.* [21] proposed a reliable offloading re-decision algorithm based on deep reinforcement learning to maximise the comprehensive utility. However, these approaches are characterized by high complexity and slow learning speeds.

Computation offloading methods based on swarm intelligence [22], [23] are often used to address multi-objective optimization problems such as computation offloading and resource allocation. This approach significantly reduces energy consumption and delay during the computation offloading process. Bi *et al.* [24] formulated and solved a nonlinear constrained optimization problem, achieving joint optimization of computation offloading and resource allocation within data centers. Yuan *et al.* [25] established a fine-grained task offloading model, predicting tasks based on user preferences, and proposed an online offloading algorithm based on particle swarm optimization, extending the battery life of intelligent mobile devices. Zhang *et al.* [26] presented an energy-aware offloading scheme, combining iterative search algorithms to find optimal solutions, resulting in lower energy consumption and delay. These approaches effectively address energy consumption and delay optimization issues in computation offloading in complex environments but do not consider long-term queue backlog issues.

For addressing long-term optimization problems in computation offloading within dynamic load, Gao *et al.* [27] introduced a prediction-based offloading and resource allocation scheme, constrained to maintain queue stability, significantly reducing average energy consumption. Dai *et al.* [28] tackled long-term migration cost issues, with the goal of minimizing offloading energy consumption. They used the Lyapunov method to transform the objective problem and employed a critic algorithm to find the optimal offloading strategy. Peng *et al.* [29] proposed an online resource coordination allocation scheme that allows computation offloading designs to adapt well to real-time dynamic networks. Guo *et al.* [30] presented

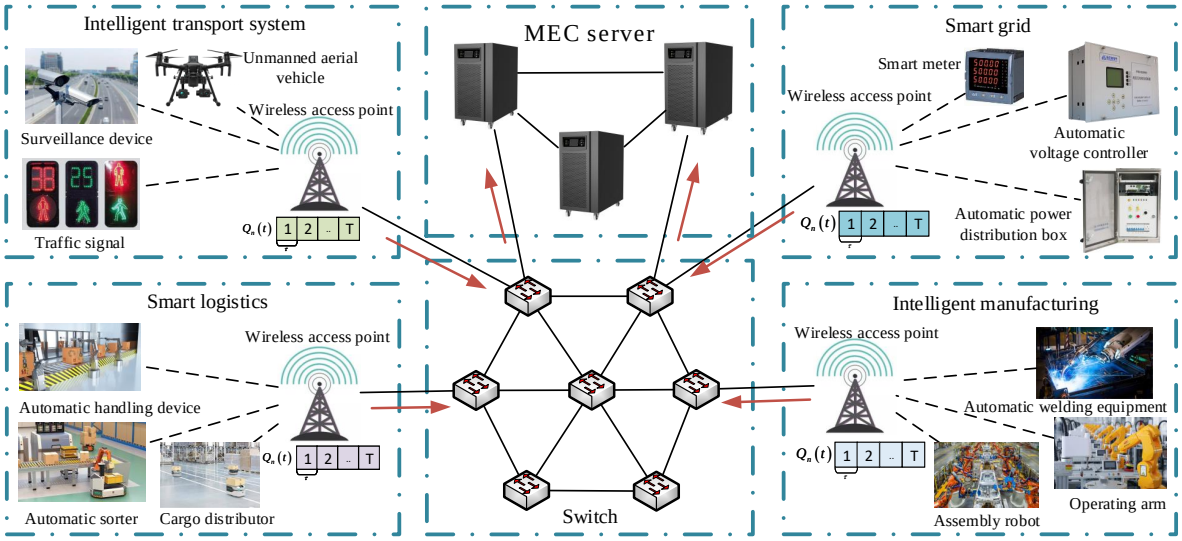


Fig. 1. Edge-Terminal Collaborative Computing Architecture

a Lyapunov optimization-based solution method, addressing the coupling of energy constraints and offloading ratio variables in multi-user partial computation offloading. Tong *et al.* [31] introduced a Lyapunov online energy optimization algorithm that effectively balances system queue backlog and energy consumption. Jiang *et al.* [32] constructed an energy-deficit queue and, combined with Lyapunov, solved the online joint offloading and resource allocation problem. Bi *et al.* [33] designed an online computation offloading algorithm to enhance network data processing capabilities and improve computational performance in computing networks. In cases where computational resources are abundant, computation offloading is relatively simple, as computational tasks are sent directly to servers for processing. However, when computational resources are limited and task backlog is significant, factors such as network connectivity among multiple servers and balancing computational resources and storage capacities among them need to be considered, making the computation offloading process more complex.

III. SYSTEM MODEL

A. Network Model

AIoT is widely used in intelligent transport systems, smart grids, intelligent manufacturing and various other industrial scenarios that require real-time control. Fig. 1 illustrates the collaborative computing network model between terminal device and MEC server in AIoT. On the terminal side, there are N industrial devices equipped with communication capabilities and limited computing capabilities, denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$. On the edge side, there are M MEC servers with communication capabilities and amounts of computing resources to assist with computational work, represented as $\mathcal{M} = \{0, 1, \ldots, M\}$. It generates task offloading decisions by collecting task and queue status information from all terminal devices in its coverage area.

Industrial device's tasks are uploaded to the MEC servers for collaborative computing through wireless communication. The information transmission process utilizes Orthogonal Frequency Division Multiple Access (OFDMA) strategy. The entire spectrum is divided into K orthogonal carriers of W bandwidth, where $\mathcal{K} = \{1, 2, ..., K\}$. At time t, the uplink transmission $r^i_{n,m}(t)$ for n^{th} industrial terminal offload task i to MEC server m can be expressed as follows:

$$r^i_{n,m}(t) = W \log_2 \left(1 + \frac{w^{i,k}_{n,m}(t) p_n(t) h^k_{n,m}(t)}{N_0}\right), \qquad (1)$$

where $w^{i,k}_{n,m}(t) \in \{0, 1\}$ is channel selection variable. $w^{i,k}_{n,m}(t) = 1$ denotes that n^{th} terminal offload task i to MEC server m using sub-carrier k for uplink transmission. $p_n(t)$ represents transmit of the n^{th} terminal at time t. $h^k_{n,m}(t)$ is channel gain for sub-carrier k. N_0 represents Additive White Gaussian Noise (AWGN).

B. Task Queuing Model

During system operation, the time domain is discretized into several time slots of equal length, which can be represented by the set $\mathcal{T} = \{1, 2, \ldots, T\}$. N industrial terminals generate I tasks in time t, which can be represented by the set $\mathcal{I} = \{1, 2, ..., I\}$, where the i^{th} task is defined as $A_i(t) \triangleq \left(S_i(t), C_i(t), T^{max}_i(t), E^{ave}_i(t)\right)$. Where $S_i(t)$ represents data size of task i. $C_i(t)$ represents computational resources required for the task. $T^{max}_i(t)$ and $E^{ave}_i(t)$ respectively represent maximum tolerable delay and normalized energy consumption value of task i. A task is classified as failed task if its processing delay exceeds maximum delay limit. Within each time slot of a discrete-time system, tasks are generated with randomness. It is assumed that the tasks arriving in each time slot of the industrial terminal obey independence and the amount of task data follows a Gamma distribution within a certain range [34].

We consider a multiple concurrency scenario where there are multiple industrial devices in the same Local Area Network (LAN). These devices have the potential to generate tasks within each time slot. To ensure that the task can complete the computation within the maximum delay constraint, the task is offloaded to the edge server for execution. The task volume $C_n^l(t)$ for locally executing tasks and the task volume $C_m^e(t)$ for edge server executing tasks at time slot t are as follows:

$$C_m^e(t) = \sum_{i=1}^{I} x_i(t) C_i(t), \quad x_i(t) = 1, \tag{2}$$

$$C_n^l(t) = \sum_{i=1}^{I} x_i(t) C_i(t), \quad x_i(t) = 0, \tag{3}$$

where $x_i(t) \in \{0, 1\}$ represents the decision of whether the i^{th} task generated by the n^{th} terminal at time slot t is to be offloaded to the terminal server. $x_i(t) = 0$ indicates local execution, while 1 indicates offloading to the edge server.

Due to the limitations of computational resources, each terminal devices performs some of its computational tasks on its local server and offloads some of its computational tasks to the associated edge server. The edge server has a greater computational capacity than is required by the tasks, but the smaller capacity of the TD causes the tasks to queue in the local task buffer. Assuming that the local buffer has sufficient capacity, the length of the task backlog in the local task buffer of TD $n \in \mathcal{N}$ at the beginning of time slot t is denoted as $Q_n(t)$. This value can be dynamically updated by the following equations:

$$Q_n(t+1) = Q_n(t) + C_n^l(t) - D_n(t), \tag{4}$$

$$D_n(t) = \begin{cases} Tw_{n,i}^l(t) \sum_{j=1}^{J} f_{n,j}^l(t-1) \\ + \left(\tau - Tw_{n,i}^l(t)\right) \sum_{i=1}^{I} f_{n,i}^l(t), & \tau \geq Tw_{n,i}^l(t) \\ \tau f_n^{l,max}(t), & \tau < Tw_{n,i}^l(t), \end{cases} \tag{5}$$

where $C_n^l(t)$ represents the computational resources of tasks that have been determined to be locally computed in time slot t. $D_n(t)$ represents the computational resources of tasks that have been processed by n^{th} TD. $f_{n,i}^l(t)$ is the local computational resource that has been allocated by n^{th} TD for task i in time slot t. $f_n^{l,max}(t)$ denotes the maximum computational capacity of n^{th} TD, which is expressed in terms of the number of CPU cycles per second. $Tw_{n,i}^l(t)$ represents the waiting latency for the task i to be locally computed, which will be discussed in detail later. In a time slot t, if there is a backlog in the task queue, the local task $C_n^l(t)$ must wait for the completion of the backlog task $Q_n(t)$ before it is executed. If there is no backlog in the task queue $Q_n(t) = 0$ and $Tw_{n,i}^l(t) = 0$, the local task can be executed immediately. In the event that the queuing delay of the local time slot exceeds the length of the local time slot, all computational resources within the local time slot are allocated to the processing of the backlogged tasks, resulting in $D_n(t) = \tau f_n^{l,max}(t)$.

C. Delay and Energy Consumption Model

1) Delay Model: The delay associated with local computation task i, denoted as $T_{n,i}^l(t)$, consists of both computation delay $Tc_{n,i}^l(t)$ and queue delay $Tw_{n,i}^l(t)$. The formula for calculating delay of tasks processed locally is as follows:

$$Tc_{n,i}^l(t) = \frac{C_i(t)}{f_{n,i}^l(t)}, \tag{6}$$

$$Tw_{n,i}^l(t) = \begin{cases} \frac{Q_n(t)}{f_n^{l,max}}, & Q_n(t) \geq \tau f_n^{l,max} \\ 0, & Q_n(t) < \tau f_n^{l,max}, \end{cases} \tag{7}$$

$$T_{n,i}^l(t) = Tc_{n,i}^l(t) + Tw_{n,i}^l(t). \tag{8}$$

Task computation results typically have a smaller size than data uploaded, resulting in negligible downlink transmission delay of task results. The processing delay $T_{m,i}^e(t)$ for task i computed on the MEC server includes the uplink transmission delay $Tt_{m,i}^e(t)$ and the computation delay $Tc_{m,i}^e(t)$.

$$Tc_{m,i}^e(t) = \frac{C_i(t)}{f_{m,i}^e(t)}, \tag{9}$$

$$Tt_{m,i}^e(t) = \frac{S_i(t)}{r_{n,m}^i(\mathrm{t})}, \tag{10}$$

$$T_{m,i}^e(t) = Tt_{m,i}^e(t) + Tc_{m,i}^e(t), \tag{11}$$

where $f_{m,i}^e(t)$ is the computational resource allocated by server m at time t for task i. The total processing delay $T_i(t)$ for the task can be represented as

$$T_i(t) = x_i(t) T_{m,i}^e(t) + (1 - x_i(t)) T_{n,i}^l(t). \tag{12}$$

2) Energy Consumption Model: When tasks are processed locally, the energy consumption associated with task processing is primarily associated with the compute process.

CPU power consists of dynamic power, short-circuit power and leakage current, of which dynamic power is dominant [35]. Therefore, we only consider the dynamic power of the mobile execution. In CMOS circuits, the energy consumption per operation is proportional to the supply voltage of the chip.

$$E_{n,i}^l(t) = \kappa_n C_i(t) f_{n,i}^l(t)^2, \tag{13}$$

where κ_n is usually taken as 10^{-27} and represents the effective switching capacitance determined by the chip architecture of the n^{th} TD [36]. The focus of our study is on the analysis of the offloading system cost from the perspective of the TD. Therefore, the energy consumption $E_{m,i}^e(t)$ when a task is offloaded to the edge server includes the uplink transmission energy consumption and excludes the consumption of the edge server when processing the task. The formula for calculating the energy consumption $E_{m,i}^e(t)$ when tasks are offloaded to the server is

$$E_{m,i}^e(t) = p_n(t) Tt_{m,i}^e(t). \tag{14}$$

The total energy consumption $E_i(t)$ for task computation can be expressed as

$$E_i(t) = x_i(t) E_{m,i}^e(t) + (1 - x_i(t)) E_{n,i}^l(t). \tag{15}$$

D. Formulation of the Objective Problem

Task offload aims to balance both task reactivity and long-term endurance of the TD. Therefore, the offload system cost function for TD task i is defined as the indicator weighting sum of delay and energy consumption, represented by $v_i(t)$.

$$v_i(t) = \alpha_i(t)\frac{T_i(t)}{T_i^{max}(t)} + (1-\alpha_i(t))\frac{E_i(t)}{E_i^{ave}(t)}, \tag{16}$$

where $\alpha_i(t)$ takes values in the range [0, 1] and represents the weighted of task i for delay and energy consumption. If $\alpha_i(t)$ is closer to 0, it means that offloading tends to reduce the processing energy consumption, and vice versa, it means that it is more important to optimise the processing delay. The offload system cost for all TD is

$$v(t) = \sum_{i=1}^{I} v_i(t). \tag{17}$$

The optimization objective of our work is to reduce the long-term average system cost of the task, denoted as $v(t)$, as mentioned in Task Backlog and Multi-slot Scheduling (TBMS) problem $\boldsymbol{P1}$.

$$\begin{aligned}
\boldsymbol{P1}\ \min_{x,\alpha,f^l,f^e}\ & \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{v(t)\} \\
\text{s.t. } & C1: \alpha_i(t)\in[0,1] \\
& C2: T_i(t) \le T_i^{max}(t) \\
& C3: 0 < \sum_{i=1}^{I} f_{n,i}^{l}(t) \le f_n^{l,max} \\
& C4: 0 < \sum_{i=1}^{I} f_{m,i}^{e}(t) \le f_m^{e,max} \\
& C5: x_i(t)\in\{0,1\} \\
& C6: w_{n,m}^{i,k}(t)\in\{0,1\}, \sum_{k=1}^{K} w_{n,m}^{i,k}(t) \le 1 \\
& C7: \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{n=1}^{N}\mathbb{E}\{Q_n(t)\} < \infty,
\end{aligned} \tag{18}$$

where $C1$ represent the weighting of delay and energy consumption, which have values in the range [0,1]. $C2$ represents the maximum allowable completion delay and energy consumption for tasks. $C3$ represent the computational resource constraints allocated to TD locally and to the MEC server.$C4$ represents that the total computational resources used by all tasks offloaded to server m cannot exceed $f_m^{e,max}$. $C5$ and $C6$ indicate that offload association and channel selection are binary variables. During the decision-making process, each TD can only select one server and one channel. $C7$ represents that task backlog for all TDs is stable.

IV. SCHEME DESIGN

A. Question Conversion

Aiming at the issue of variable coupling in long-term edge collaborative offloading, the proposal of a objective problem transformation scheme based on Lyapunov theory has been put forward. Therefore, this approach can convert the long-term optimization problem into a time slot optimization problem. The backlog vector at time slot t is defined as $\theta(t) = \{Q_1(t), \ldots, Q_n(t), \ldots, Q_N(t)\}$, and the Lyapunov function is defined as:

$$L(\theta(t)) = \frac{1}{2}\sum_{n=1}^{N} Q_n(t)^2, \tag{19}$$

where the function $L(\theta(t))$ is an optimization function controlling the queue state. Clearly, the value of $L(\theta(t))$ will increase significantly when the queue experiences congestion. Therefore, the Lyapunov drift function is defined as:

$$\Delta L(\theta(t)) = \mathbb{E}\{L(\theta(t+1)) - L(\theta(t))\}. \tag{20}$$

Associating target problem $\boldsymbol{P1}$ with queue state optimization for joint optimization, resulting in problem $\boldsymbol{P2}$.

$$\begin{aligned}
\boldsymbol{P2}\ & \min_{x,\alpha,f^l,f^e} \Delta L(\theta(t)) + V\mathbb{E}\{v(t)\} \\
& s.t. C1 - C7,
\end{aligned} \tag{21}$$

where V is a non-negative weight constant.

Theorem 1: For any indicator weighting sum V, task states, and offloading decisions, there exists an upper bound on the sum of drift and penalty functions.

$$\begin{aligned}
&\Delta L(\theta(t)) + V\mathbb{E}\{v(t)\} \le \\
&\quad B - \sum_{n=1}^{N} Q_n(t)\mathbb{E}\left\{D_n(t) - C_n(t)\right\} + V\mathbb{E}\{v(t)\}, \\
&\quad B = 0.5\sum_{n=1}^{N}\left[D_n^{\max}(t)^2 + [\sum_{i=1}^{I} C_i(t)]^2\right],
\end{aligned} \tag{22}$$

where B is a constant.

Proof: The following can be derived according to Eq. (4).

$$\begin{aligned}
Q_n(t+1)^2 &= \left[Q_n(t) - D_n(t) + C_n^l(t)\right]^2 \\
&= Q_n(t)^2 + D_n(t)^2 + C_n^l(t)^2 + 2Q_n(t)C_n^l(t) \\
&\quad - 2Q_n(t)D_n^l(t) - 2C_n(t)D_n(t) \\
&\le Q_n(t)^2 + D_n(t)^2 + C_n^l(t)^2 \\
&\quad - 2Q_n(t)\left[D_n(t) - C_n^l(t)\right] \\
&\le Q_n(t)^2 + D_n^{max}(t)^2 + \left(\sum_{i=1}^{I} C_i(t)\right)^2 \\
&\quad -2Q_n(t)\left[D_n(t) - C_n^l(t)\right].
\end{aligned} \tag{23}$$

So,

$$\begin{aligned}
L(\theta(t+1)) - L(\theta(t)) \le \frac{1}{2}\sum_{n=1}^{N}\left[D_n^{max}(t)^2 + \left(\sum_{i=1}^{I} C_i(t)\right]^2\right) \\
- \sum_{n=1}^{N} Q_n(t)\left[D_n(t) - C_n^l(t)\right],
\end{aligned} \tag{24}$$

$$\begin{aligned}\Delta L(\theta(t)) \leq & \frac{1}{2}\sum_{n=1}^{N}\left[D_n^{max}(t)^2+\left(\sum_{i=1}^{I}C_i(t)\right)^2\right] \\ & -\sum_{n=1}^{N}Q_n(t)\mathbb{E}\left[D_n(t)-C_n^l(t)\right] \\ = & B-\sum_{n=1}^{N}Q_n(t)\mathbb{E}\left[D_n(t)-C_n^l(t)\right].\end{aligned} \tag{25}$$

Neglecting the upper bound constant term B, the objective problem $\boldsymbol{P2}$ is transformed into $\boldsymbol{P3}$.

$$\begin{aligned}\boldsymbol{P3}\ \min_{x,\alpha,f^l,f^e}\ & Vv(t)-\sum_{n=1}^{N}Q_n(t)\left[D_n(t)-C_n^l(t)\right] \\ \text{s.t. } & C1-C6.\end{aligned} \tag{26}$$

B. Resource Allocation of Terminal Devices

Building upon the $\boldsymbol{P3}$ problem, the correlation variable x are set as constants, with computational resource allocation being the sole consideration. When only considering the $C4$ constraint, the objective problem $\boldsymbol{P3}$ is transformed into the TD computational resource allocation problem $\boldsymbol{P4}$.

$$\begin{aligned}\boldsymbol{P4}\quad \min_{f^l}\ & \sum_{i=1}^{I}V\left[\alpha_i(t)\frac{C_i(t)}{T_i^{max}(t)f_{n,i}^l(t)}\right. \\ & \left.+(1-\alpha_i(t))\frac{\kappa_n C_i(t)f_{n,i}^l(t)^2}{E_i^{ave}(t)}\right] \\ & -\sum_{n=1}^{N}Q_n(t)\left[D_n(t)-C_n^l(t)\right] \\ \text{s.t. } & C3.\end{aligned} \tag{27}$$

To ensure that tasks can be completed within their maximum tolerable delay, the TD computational resource allocation needs to satisfy

$$C_i(t)/f_{n,i}^l(t) \leq \left[T_i^{max}(t)-Tw_{n,i}^l(t)\right]. \tag{28}$$

The lower bound of local computational resources is defined as

$$f_{n,i}^{l,min}(t)=C_i(t)/\left[T_i^{max}(t)-Tw_{n,i}^l(t)\right]. \tag{29}$$

The prior weighted $\alpha_i(t)$ of delay and energy consumption is taken as the lower bound for the new weighted and is fed back into the subsequent iterations of the indicator weighted optimization algorithm. The TD decisions are independent of each other. By optimizing the computational resource allocation decision for each TD, the solution to the objective problem $\boldsymbol{P4}$ can be achieved. Local task backlogs impact computational resource allocation decisions. Therefore, we discuss the TD computational resource allocation separately under $Q_n(t)=0$ and $Q_n(t)\neq 0$.

1) $Q_n(t)=0$: When there are no backlogged tasks in the local queue, the computational resource allocation objective

Algorithm 1 Pseudo code of *RALD*

Input: $C_i(t)$, $Q_n(t)$, $\alpha_i(t)$, κ_n, ε_2, previous time slot task-related information, , $x_i(t)$;
Output: $f_{n,i}^{l,*}(t)$, $x_i(t)$;

```
Calculate Tw^l_{n,i}(t) and f^{l,min}_{n,i}(t) based on Eq.(7) and Eq.(29);
left = f^{l,min}_{n,i}(t), right = f^{l,max}_n;
if f^{l,min}_{n,i}(t) > f^{l,max}_n then
    The task i is offloaded to the MEC server;
    f^{l,*}_{n,i} = 0, x_i(t) = 1;
else
    if Q_n(t) = 0 then
        Calculate f^{l,*}_{n,i}(t) according to Eq.(32) and Eq.(33);
    else
        while right − left > ε_2 do
            mid = 0.5(left + right);
            if F_2(mid) ≤ F_2(right) then
                right = mid;
            else
                left = mid;
            end if
        end while
        f^{l,*}_{n,i} = mid;
    end if
end if
```

function $F_1\left(f_{n,i}^l(t)\right)$ and its first derivative $F_1'\left(f_{n,i}^l(t)\right)$ for each TD are written as follows.

$$\begin{aligned}F_1\left(f_{n,i}^l(t)\right)=&\mathrm{V}\left[\alpha_i(t)\frac{C_i(t)}{T_i^{max}(t)f_{n,i}^l(t)}\right. \\ & \left.+(1-\alpha_i(t))\frac{\kappa_n C_i(t)f_{n,i}^l(t)^2}{E_i^{ave}(t)}\right],\end{aligned} \tag{30}$$

$$\begin{aligned}F_1'\left(f_{n,i}^l(t)\right)=&-\mathrm{V}\left[\alpha_i(t)\frac{C_i(t)}{T_i^{max}(t)f_{n,i}^l(t)^2}\right. \\ & \left.+2(1-\alpha_i(t))\frac{\kappa_n C_i(t)f_{n,i}^l(t)}{E_i^{ave}(t)}\right].\end{aligned} \tag{31}$$

The zero points of the first-order derivative function are as follows.

$$\widehat{f_{n,i}^l}(t)=\sqrt[3]{\frac{\alpha_i(t)E_i^{ave}(t)}{2(1-\alpha_i(t))\kappa_n T_i^{max}(t)}}. \tag{32}$$

Proof: When $f_{n,i}^l(t) > \widehat{f_{n,i}^l}(t)$, the function is monotonically increasing. When $f_{n,i}^l(t) < \widehat{f_{n,i}^l}(t)$, the function is monotonically decreasing. Thus, $\widehat{f_{n,i}^l}(t)$ is the local minimum point of $F_1\left(f_{n,i}^l(t)\right)$. The optimal allocation decision for local computing resources is denoted as $f_{n,i}^{l,*}(t)$.

$$f_{n,i}^{l,*}(t)=\begin{cases}f_{n,i}^{l,min}(t) & \widehat{f_{n,i}^l}(t)\leq f_{n,i}^{l,min}(t) \\ f_{n,i}^l(t) & f_{n,i}^{l,min}(t)<\widehat{f_{n,i}^l}(t)<f_n^{l,max} \\ f_{n,i}^{l,max} & f_n^{l,max}\leq\widehat{f_{n,i}^l}(t).\end{cases} \tag{33}$$

2) $Q_n(t) \neq 0$: When there is a task backlog in the local queue, combining with Eq.(5), the computational resource allocation objective function $F_2\left(f_{n,i}^l(t)\right)$ and its second derivative $F_2''\left(f_{n,i}^l(t)\right)$ are interpreted for each TD as follows:

$$\begin{aligned} F_2\left(f_{n,i}^l(t)\right) =& \mathrm{V}\left[\alpha_i(t)\frac{C_i(t)}{T_i^{max}(t)f_{n,i}^l(t)}\right. \\ &\left.+(1-\alpha_i(t))\,\frac{\kappa_n C_i(t) f_{n,i}^l(t)^2}{E_i^{ave}(t)}\right] \\ &-Q_n(t)\left[D_n(t)-C_n^l(t)\right], \end{aligned} \tag{34}$$

$$\begin{aligned} F_2''\left(f_{n,i}^l(t)\right) =& V\left[\alpha_i(t)\frac{2C_i(t)}{T_i^{max}(t)f_{n,i}^l(t)^3}\right. \\ &\left.+2\,(1-\alpha_i(t))\,\frac{\kappa_n C_i(t)}{E_i^{ave}(t)}\right]. \end{aligned} \tag{35}$$

Because function $F_2\left(f_{n,i}^l(t)\right)$ is first-order differentiable, and its second-order derivative is greater than zero, it is a convex function. The function must have a minimum point within the range of $\left[f_{n,i}^{l,min}(t), f_n^{l,max}\right]$, and the Bisection method is employed to find the optimal resource allocation decision within the interval. The pseudo code of the Resource Allocation of Local Device (RALD) is described in **Algorithm 1**.

C. Resource Allocation of MEC Servers

When considering only constraints $C5$ and $C6$, the objective problem ***P3*** is transformed into the MEC server computational resource allocation problem ***P5***.

$$\begin{aligned} \boldsymbol{P5} \quad & \min_{f^e} \sum_{i=1}^{I} V\alpha_i(t)\frac{x_i(t)C_i(t)}{T_i^{max}(t)f_{m,i}^e(t)} \\ & \text{s.t. } C4. \end{aligned} \tag{36}$$

The computational resource allocation function of server m for the i^{th} task within it is explained as Eq. (37). The total number of tasks that are offloaded to the MEC server is S ($S \leq I$), and the set is $S_m = \{1, \ldots, i, \ldots, S\}$.

$$F_3\left(f_{m,i}^e(t)\right) = V\alpha_i(t)\frac{C_i(t)}{T_i^{max}(t)f_{m,i}^e(t)}. \tag{37}$$

Theorem 2: $F_3\left(f_{m,i}^e(t)\right)$ is a strictly convex function.

Proof: The Hessian matrix of $F_3\left(f_{m,i}^e(t)\right)$ can be stated as

$$H = \begin{bmatrix} \frac{\partial^2 F\left(f_{m,1}^e\right)}{\partial\left(f_{m,1}^e\right)^2} & \cdots & \frac{\partial^2 F\left(f_{m,1}^e\right)}{\partial\left(f_{m,1}^e\right)\partial\left(f_{m,S}^e\right)} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F\left(f_{m,S}^e\right)}{\partial\left(f_{m,1}^e\right)\partial\left(f_{m,S}^e\right)} & \cdots & \frac{\partial^2 F\left(f_{m,S}^e\right)}{\partial\left(f_{m,S}^e\right)^2} \end{bmatrix}. \tag{38}$$

When satisfying $\forall S \in S_m, i \neq S$, the second-order derivatives and mixed second-order partial derivatives of $F_3\left(f_{m,i}^e\right)$ are expressed as

$$\begin{aligned} &\partial^2 F_3\left(f_{m,i}^e(t)\right)/\partial\left(f_{m,i}^e(t)\right)^2 = \\ &\quad 2V\alpha_i(t)C_i(t)/T_i^{max}(t)\left(f_{m,i}^e(t)\right)^3 > 0, \end{aligned} \tag{39}$$

$$\partial^2 F_3\left(f_{m,i}^e(t)\right)/\partial(f_{m,i}^e(t))\partial\left(f_{m,s}^e(t)\right) = 0, \tag{40}$$

So, $F_3\left(f_{m,i}^e(t)\right)$ is a strictly convex function because of the eigenvalues of matrix H are all greater than zero.

The KKT conditions are employed to solve problem ***P5***. The Lagrangian function can be represented using Eq.(41).

$$F\left(f_{m,i}^e, \lambda\right) = \frac{C_i}{f_{m,i}^e} + \lambda\left(\sum_{i=1}^{S} f_{m,i}^e - f_m^{e,max}\right), \tag{41}$$

where λ is non-negative Lagrange multiplier. For $\forall i \in S_m$, the KKT conditions can be expressed as

$$-\frac{C_i}{\left(f_{m,i}^e\right)^2} + \lambda = 0, \tag{42}$$

$$\lambda\left(\sum_{i=1}^{S} f_{m,i}^e - f_m^{e,max}\right) = 0, \tag{43}$$

$$\lambda \geq 0, \tag{44}$$

$$\sum_{i=1}^{S} f_{m,i}^e - f_m^{e,max} \leq 0. \tag{45}$$

The update formulas for λ is represented as

$$\lambda(t+1) = \max\left\{\varepsilon_1, \lambda(t) + \delta_1(t)\left(\sum_{i=1}^{S} f_{m,i}^e - f_m^{e,max}\right)\right\}. \tag{46}$$

The pseudo code of the Resource Allocation of MEC Server (RAES) is described in **Algorithm 2**.

Algorithm 2 : Pseudo code of *RAES*

Input: $f_m^{e,max}$, λ, $\delta_1(t)$, ε_1, S_m;
Output: $\{f_{m,i}^e(t) | \forall i \in S_m\}$;

```
while f_m^{e,max} − Σ_{i=1}^{S} f_{m,i}^e(t) > ε_1 do
    Calculate the allocated computational resources f_{m,i}^e(t)
    for task i by Eq.(42);
    Update λ according to Eq.(46);
end while
```

D. Optimization Problem of Indicator Weights

Tasks partial are offloaded from the TD can reduce TD task backlog. Assuming computational resource allocation and offloading decisions are known, the original problem is then transformed into the weighted optimization problem of delay and energy consumption. An Enhanced Jumping Spider Optimization (EJSO) algorithm is proposed to adjust the indicator weights in each time slot, enabling the efficient utilization of global resources.

The jumping spider optimization algorithm consists of predation strategy, searching strategy, and pheromone update strategy. It simulates the jumping, pursuing, and foraging behaviors of jumping spiders during their hunting process to conduct optimization, while using pheromone to evaluate the current optimization credibility.

Assuming the population consists of I jumping spiders (task), each of the jumping spiders searches for optimal positions in the D-dimensional space. The initial position set of jumping spider population is denoted as $P^0(t)$.

$$P^0(t) = \begin{bmatrix} P^0_{1,1} & P^0_{1,2} & \cdots & P^0_{1,D} \\ P^0_{2,1} & P^0_{2,2} & \cdots & P^0_{2,D} \\ \vdots & \vdots & P^0_{i,d} & \vdots \\ P^0_{I,1} & P^0_{I,2} & \cdots & P^0_{I,D} \end{bmatrix}. \tag{47}$$

The elite opposition-based learning strategy is introduced to generate the optimal solutions for the algorithm. The fundamental idea behind the backward strategy is to introduce a reverse learning mechanism on top of the current solutions to generate reverse points. This strategy optimizes the search region of the traditional algorithm's solutions and avoids densely distributed randomly generated solutions. For any jumping spider $i \in \mathcal{I}$ in population, its reverse solution is defined as R_i.

$$R_i = l_i + u_i - P_i, \tag{48}$$

where l_i and u_i are respectively the lower and upper bounds of the indicator weights for task i in time slot t.

The generation of reverse solutions is advantageous for increasing the diversity of the solutions within the population. However, the generation of reverse solutions can be somewhat blind and may not effectively explore the search space of the current solutions. To address this issue, elite individuals are identified based on their fitness values using reverse solutions.

Furthermore, the objective fitness value for the EJSO algorithm is set as Eq.(26). The fitness values of inverse solution is represented as $F(R_i)$. When

$$F(R_i) \leq F(P_i), \tag{49}$$

the reverse solution is recognized as elite and retained. $F(P_i)$ is the fitness value of current optimal solution for task i.

The pseudocode of the EJSO algorithm is described in **Algorithm 3**. Firstly, generates initial position of jumping spider population. Secondly, jumping spiders choose predation, leaping, or searching behavior strategies based on probabilities and update their positions accordingly. Thirdly, jumping spiders use a pheromone formula to evaluate position selection, and if the pheromone is too low, they perform position updates. Next, each individual calculates its current iteration's position utility value and retains the best utility value and its corresponding individual position among all individuals. Finally, the reverse solution is generated by the elite opposition-based learning strategy and the fitness values of the reverse solution and the current optimal solution are compared. The algorithm continues to perform the above process until the end of iterations, resulting in the optimal indicator weights $\alpha^*_i(t)$.

Algorithm 3 : Pseudo code of *EJSO*

Input: $P^0(t)$, μ^{max};
Output: $P^*_i(\mu)$, $v^*(\mu)$;
Randomly generating initial positions;
Record the lower and upper bounds of the position l_i and u_i;
Calculate the fitness value set $\{F(P^0_{i,d})\}$ of all initial positions by Eq.(26) and recording the best and worst fitness value along with their corresponding positions $P^b_i(\mu)$, F_{min}, $P^w_i(\mu)$, and F_{max};
for $\mu \leftarrow 1$ to μ^{max} **do**
  if $random < 0.5$ **then**
    if $random < 0.5$ **then**
      $P_i(\mu+1) = \frac{1}{2}[P_i(\mu) - P_r(\mu)]$;
    else
      $P_i(\mu+1) = P_i(\mu)\tan(\gamma) - \frac{gP_i^2(\mu)}{2[v_0\cos(\gamma)]^2}$;
    end if
  else
    if $random < 0.5$ **then**
      $P_i(\mu+1) = P^b_i(\mu) + walk(0.5-\varepsilon)$;
    else
      $P_i(\mu+1) = P^b_i(\mu) + \varepsilon[P^b_i(\mu) - P^w_i(\mu)]$;
    end if
  end if
  $pher(i) = \frac{F_{\max} - F(P_i(\mu+1))}{F_{\max} - F_{\min}}$;
  if $pher(i) < 0.3$ **then**
    $P_i(\mu+1) = P^b_i(\mu) + \frac{1}{2}[P_{r_1}(\mu) - (-1)^{\omega}P_{r_2}(\mu)]$;
  end if
  Calculate the fitness value $F(P_i(\mu+1))$;
  if $F(P_i(\mu+1)) < F_{min}$ **then**
    $P^*_i(\mu) = P_i(\mu+1)$, $F_{min} = F(P_i(\mu+1))$;
  end if
  The fitness value of R_i is generated by by Eq.(48) ;
  if $F(R_i) < F_{min}$ **then**
    $P^*_i(\mu) = R_i$, $F_{min} = F(R_i)$;
  end if
  Update $P^b_i(\mu)$, F_{min}, $P^w_i(\mu)$, and F_{max};
end for
$v^*(\mu) = F_{min}$; Update $P^*_i(\mu)$, $v^*(\mu)$;

E. Integrated Offloading Decision Scheme

Assuming constant computational resource allocation indicator weights, the original problem is transformed into a time-slot-based problem of selecting the appropriate MEC server for TD computational tasks.

A method for integrated offloading decision based on task requirement ranking is proposed. This method obtains approximate optimal offloading association and channel selection decisions based on task urgency. The task requirement of n^{th} TD in timeslot t is defined as $req_i(t)$.

$$req_i(t) = \frac{C_i(t)}{T^{max}_i(t) - Tw^l_{n,i}(t)}. \tag{50}$$

The longer the task queuing delay, the smaller the remaining processing delay $T^{max}_i(t) - Tw^l_{n,i}(t)$ indicating a more urgent demand for task completion. The task information of the

devices and the status of many devices and edge servers are transmitted to the base station, and the base station server collects the task information of all TDs in the current time slot and classifies them according to the task requirements.

When the MEC server m is assigned a task, the calculated pseudo remaining computational resource $\widehat{f_m^e}$ is stated as

$$\widehat{f_m^e} = f_m^{e,max} - \sum_{i=1}^{I} x_i(t) req_i(t). \tag{51}$$

In the subsequent task selection process, the TD task selects the MEC server with the highest value of $\widehat{f_m}$ from the current set of all MEC servers' pseudo remaining computational resources $\left\{\widehat{f_m^e} \mid m \in \mathcal{M}\right\}$. If multiple MEC server have the same $\widehat{f_m^e}$ value, the MEC server selects the closest edge server for task computation to reduce transmission delay. During the channel selection process for TD, the interference on the shared channel k is represented as γ^k.

$$\gamma^k = \sum_{i=1}^{I} w_{n,m}^{i,k}(t) p_n(t) h_{n,m}^k(t). \tag{52}$$

The higher the task requirement, the higher the priority for TD to select channels. TD sequentially choose channels from low to high based on the current channel interference level $\left\{\gamma^k \mid k \in \mathcal{K}\right\}$ to ensure better transmission quality. The pseudo code of the CODET is described in **Algorithm 4**.

Complexity analysis: **Algorithm 4** comprises three sub-algorithms: **Algorithm 1**, **Algorithm 2**, and **Algorithm 3**. Specifically, **Algorithm 1** employs a bisection method with computational complexity of $O(\log \frac{f_n^{l,max} - f_{n,i}^{l,min}}{\varepsilon_2})$. **Algorithm 2** contains a single nested loop with complexity $O(\frac{f_m^{e,max}}{\varepsilon_1} \cdot S)$. **Algorithm 3** contains one nested loop with complexity of $O(\mu^{max})$. Considering that each device executes these algorithms independently for N devices and I tasks per device, the total computational complexity of **Algorithm 4** can be expressed as

$$O(I \cdot (\log \frac{f_n^{l,max} - f_{n,i}^{l,min}}{\varepsilon_2} + \frac{f_m^{e,max}}{\varepsilon_1} \cdot S) + \mu^{max}).$$

It is worth noting that $f_n^{l,max}$, $f_m^{e,max}$, ε_1, and ε_2 are constants independent of the algorithm input size, then their values do not change as the problem size grows, $f_n^{l,min} = 0$. Furthermore, the total number of computational tasks I differs between time slots, as does the number of tasks S be allocated to the server, $S \leq I$. The complexity of **Algorithm 4** reduces to

$$O(I \cdot (\log \frac{f_n^{l,max}}{\varepsilon_2} + \frac{f_m^{e,max}}{\varepsilon_1} \cdot S) + \mu^{max}).$$

Within the time slot, the logarithmic term $\log \frac{f_n^{l,max}}{\varepsilon_2}$ and $\frac{f_m^{e,max}}{\varepsilon_1}$ are constants. The complexity of **Algorithm 4** reduces to $O(I^2)$. **Algorithm 4** has moderate complexity, does not involve a large number of computationally intensive operations and can be used to perform online resource allocation tasks on resource-constrained terminal devices.

Algorithm 4 : Pseudo code of CODET

Input: $C_i(t)$, $A^0(t)$, $f_m^{e,max}$, λ, $\delta_1(t)$, ε_1, I, S_m, $Q_n(t)$, κ_n, ε_2, μ^{max};

Output: $\{x_i(t)\}$, $\{\alpha_i(t)\}$, $f_{n,i}^{l,*}(t)$, $f_{m,i}^e(t)$, $v(t)$;

```
Collects the task information of all TDs and the status of all devices and edge servers in the current time slot t.
Generated {x_i(t), x_i(t) = {0, 2}} randomly, v(t) = inf;
for i ← 1 to I do
    Obtain f_{n,i}^{l,*}(t) and x_i(t) according to Algorithm 1;
    Calculate the task requirement according to Eq.(50);
    if x_i(t) ≠ 1 then
        x_i(t) = {0, 2} randomly;
    end if
    if x_i(t) = 1 or x_i(t) = 2 then
        Calculate the remaining computational resource f̂_m^e by Eq.(51) and sort them;
        Select the channel according to Eq.(52);
        Execute Algorithm 2 to obtain f_{m,i}^e(t);
    end if
end for
Execute Algorithm 3 to obtain P_i^*(μ) and v^*(μ);
if v^*(μ) < v(t) then
    v(t) = v^*(μ);
end if
Record the corresponding {x_i(t)} and {P_i^*(μ)}, {α_i(t)} = {P_i^*(μ)};
```

V. SIMULATION EXPERIMENTS AND PERFORMANCE ANALYSIS

A. Simulation Parameter Setting

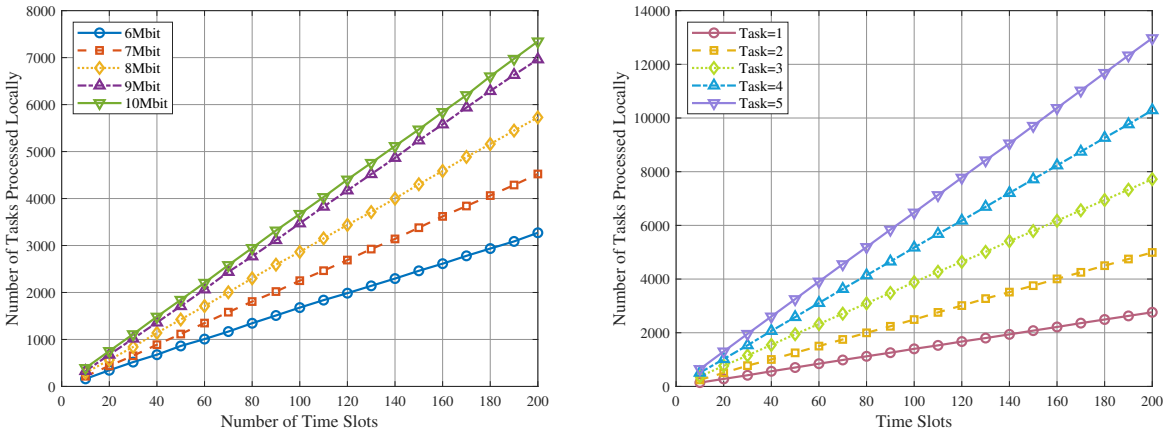
MATLAB 2021 is validated for stability and effectiveness of our solutions. It is assumed that there are $M = 3$ edge servers and $N = 20$ devices within a cellular network coverage area of 3000×3000m. The amount of task data per time slot of the TDs obeys a Gamma Distribution (mean υ, variance σ^2) [37]. There are $K = 10$ subchannels for uplink task transmission with a bandwidth of 5 MHz per sub-channel. The channel gain is $128.1 + 37.6 \log_{10} d_{n,m}$, where $d_{n,m}$ is the distance (in kilometers) from n^{th} device to server m. Other parameter settings can be found in TABLE I [38].

TABLE I
SIMULATION EXPERIMENT PARAMETER SETTINGS

Parameter Name	Parameter Value
Servers Computational Capability $f_m^{e,max}$	30×10^9 cycles /s
Device Computational Capability $f_n^{l,max}$	$[0.5, 2] \times 10^9$ cycles /s
Maximum Delay of Tasks $T_{n,max}(t)$	$[1, 1.5]s$
Normalized Energy Consumption $E_{n,ave}(t)$	2J
Channel Noise N_0	10^{-7}w
Device Transmit Power $p_n(t)$	0.5w
Device Effective Switching Capacitance κ_n	10^{-27}
Lyapunov Optimization Weights V	10^6
Total Number of Time Slot T	200τ

The CODET is compared with four different mechanisms.

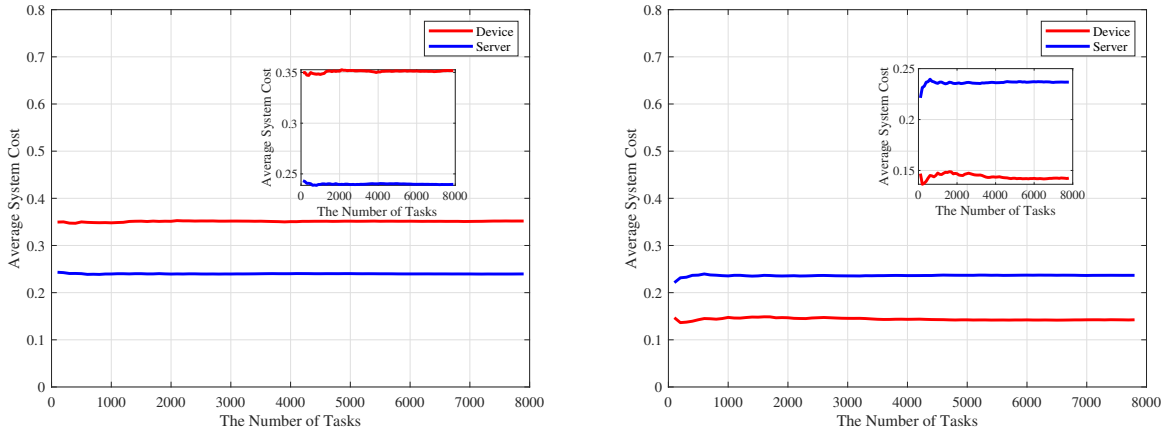
Deep Reinforcement learning-based Online Offloading (*DROO*) [39] and Lyapunov-guided Deep Reinforcement learning (DRL)-based Online Offloading *(LyDROO)* [40]: The



(a) The impact of the amount of the task data

(b) The impact of the tasks number of per device

Fig. 2. The impact of different factors on the task backlog



(a) None

(b) EJSO

Fig. 3. The impact of EJSO on the average system cost

two schemes optimally adapt task offloading decisions and resource allocations to the time-varying channel conditions.

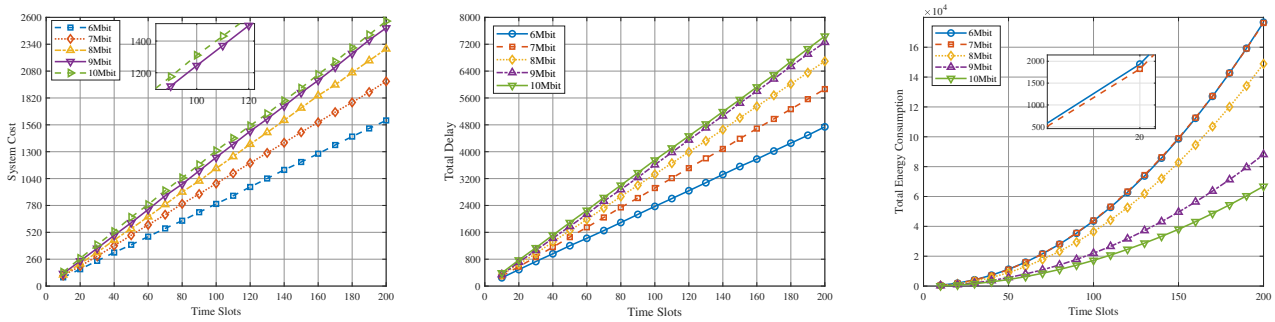
Only Energy Consumption (*OE*): This scheme aims to minimize energy consumption and adheres to our mechanisms for offloading correlation, channel selection, and computational resource allocation.

Only Delay (*OD*): This scheme aims to minimize delay and adheres to our mechanisms for offloading correlation, channel selection, and computational resource allocation.

B. Simulation Results Analysis

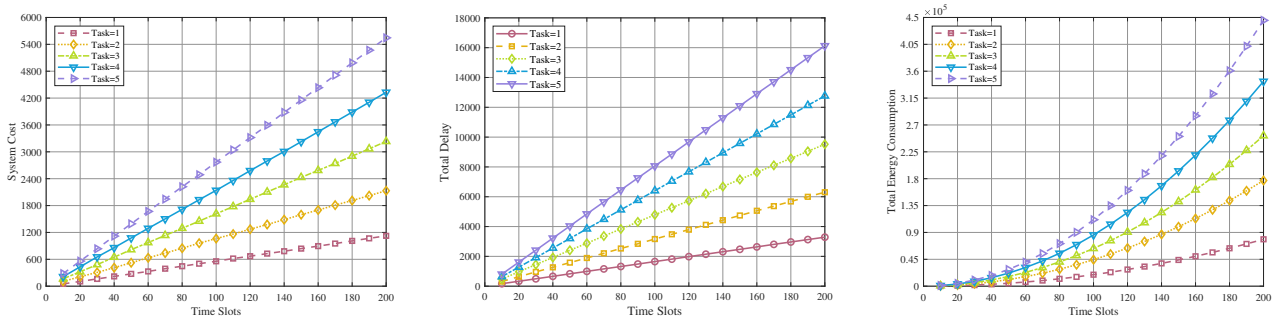
1) Task Backlog Analysis: Fig. 2 shows the impact of various factors on the task backlog. As time progresses, the number of tasks in the backlog increases across time slots. Furthermore, the backlog becomes more significant as the number of tasks and data volumes increase. The two algorithms of always processing tasks locally or always offloading tasks to the MEC server lead to a significant increase in queuing delay or transmission delay, which results in a significant accumulation of tasks, especially under high workload conditions. In turn, this causes a significant increase in both latency and energy consumption, often exceeding acceptable thresholds by orders of magnitude. Therefore, there is a strong demand for a dynamic task offloading approach for computationally resource-constrained terminal devices to effectively alleviate the task backlog problem.

2) The Impact of EJSO on The Average System Cost: With the number of tasks randomly distributed in $[1, 3]$ and the amount of task data follows a Gamma distribution with a mean υ = 6Mbit and a variance σ^2 = 1, Fig. 3 shows the impact of EJSO on the average system cost. It is obvious that the average system cost of both local and server processing tasks decreases after improving indicator weights α with the EJSO. By optimizing the initial solution and behavioral strategy, EJSO can better balance delay and energy consumption to reduce system cost.



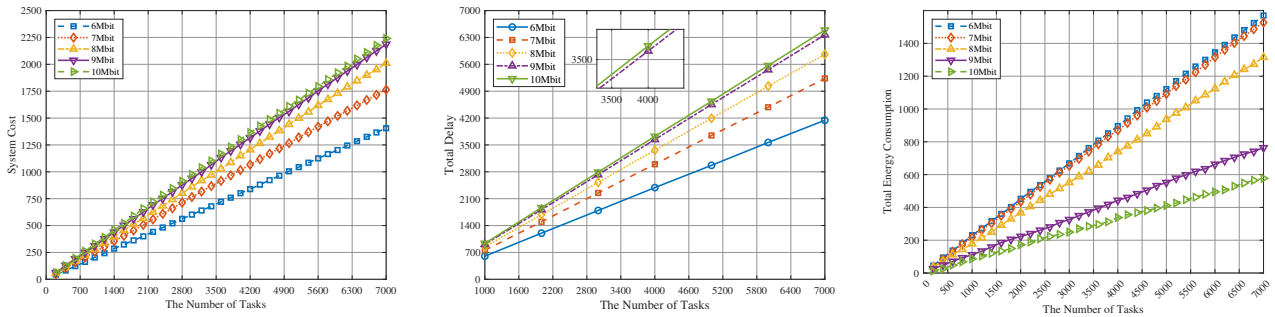
(a) System cost (b) Delay (c) Energy consumption

Fig. 4. The impact of time slots on system cost



(a) System cost (b) Delay (c) Energy consumption

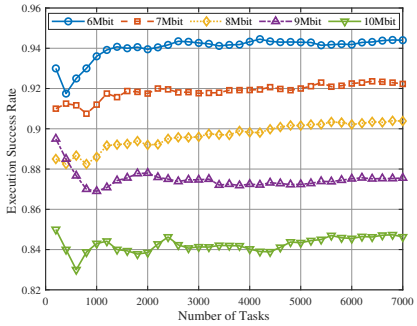
Fig. 5. The impact of the tasks number of per device on system cost



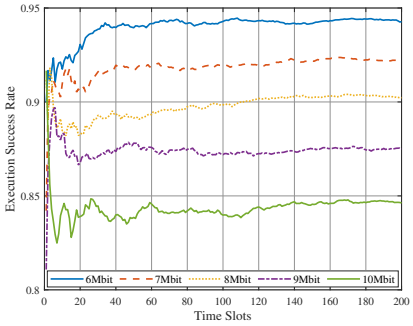
(a) System cost (b) Delay (c) Energy consumption

Fig. 6. The impact of the total number of tasks on system cost

3) System Cost Analysis: Fig. 4, Fig. 5 and Fig. 6 show the impact of different factors on system cost. The system cost increases with time slots, the amount of task data, the tasks number of per device and the total number of tasks. It is obvious that the system cost increases with the amount of task data for a certain number of tasks. This is because the computing resources that need to be shared by the terminal increase with the number of tasks. The delays and energy consumption of transmission and computation for each task are increased, resulting in an increase in system cost. As shown in Fig. 4 (c) and Fig. 6 (c), when the number of tasks is certain, the amount of data increases while the energy consumption decreases. This is because our system cost considers local delay and energy consumption. For realism, we design a long transmission distance, the transmission delay is much greater than local computation delay. For a certain bandwidth, the transmission time and transmission energy consumption of task offloaded increase with the amount of task data, so the data-heavy tasks tend to be computed locally. The computational energy consumption is much less than the transmission energy consumption, leading to a slight reduction in the total energy consumption. However, the computation delay increases significantly with the average queue length, leading to an increase in the system cost with the amount of



(a) The impact of the total number of tasks



(b) The impact of the amount of the data

Fig. 7. The execution success rate

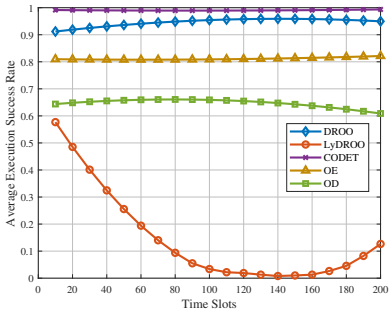


Fig. 8. Average Execution Success Rate

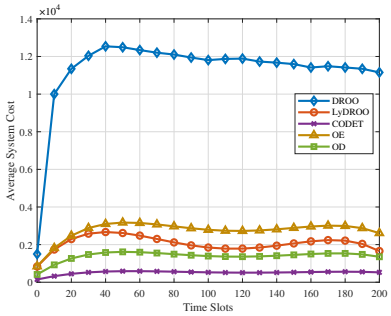


Fig. 9. System cost

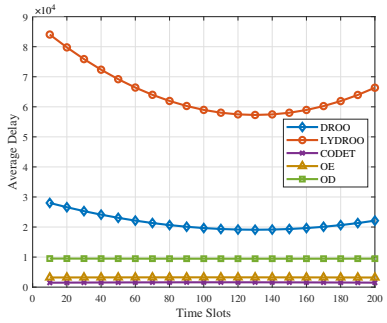


Fig. 10. Delay

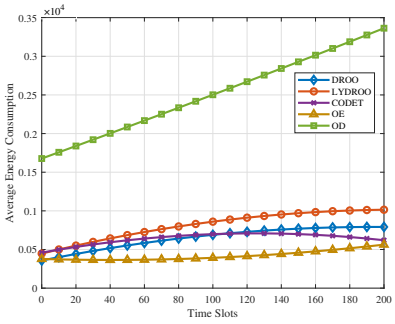


Fig. 11. Energy consumption

task data.

4) Execution Success Rate Analysis: Fig. 7 shows the impact of different factors on execution success rate of each terminal device when the number of tasks generated per time slot is randomly distributed as $[2, 3]$. It is clear that as the total number of tasks and the time slots increase, execution success rate tend to stabilize. But execution success rate decreases with the amount of data increased. The average task size is under mean υ = 7Mbit, and the task execution success rate can reach more than 90%. This is because the increase in the number of tasks and the amount of data leads to strong competition for channel and computational resources, which reduces the success rate of the tasks.

5) Comparative Experiment: With the number of tasks randomly distributed in $[1, 3]$ and the amount of task data follows a Gamma distribution with a mean υ = 6Mbit and a variance σ^2 = 1, Fig.8 shows the average execution success rate for five offloading schemes. It is clear that the average execution success rate of each scheme tends to stabilize with time slots. The average execution success rate of CODET is the highest, being 4.51% higher than that of the suboptimal scheme. Compared to Fig. 2, the proposed algorithm can effectively reduce the terminal device's task backlog and improve the rate at which tasks are successfully executed. Because it takes relatively long to transmit data to the edge server, tasks are more likely to be processed locally than offloaded in order to meet latency constraints. Compared to Fig. 7, the task execution success rate of the proposed algorithm remains above 99%. This indicates that the system may not be able to process or offload all tasks within the deadline when there are too many tasks on the local device per time slot. Although our algorithm dynamically adjusts offloading decisions based on task urgency and resource availability, the number of tasks generated by each device per time slot should be randomly distributed within the range $[1, 3]$. Beyond this range, the system is unable to maintain a 99% task completion rate.

Fig. 9 presents the average system cost for five offloading schemes. It is clear that the average system cost of each scheme tends to stabilize with time slots. The system cost of CODET is the lowest, being at least 57.13% lower than the system cost of the suboptimal scheme. The reason is that the unreasonable indicator weights decisions and offloading association decisions of the other schemes increased latency and reduced task success rate. While CODET uses EJSO to optimize the indicator weights, which effectively balances delay and energy consumption. This approach reduces system cost and ensures task success rates.

As shown in Fig. 8-11, the CODET algorithm is superior in all performance metrics. It's worth mentioning that although better in terms of latency or energy consumption, the OE and OD schemes have a much lower task execution success rate than the proposed scheme, with a significantly higher average system cost. This is because the CODET approach considers task delay and energy consumption, while also making full use of the computational resources of the edge servers to assist with local task completion. The approach has

low computational complexity and can arrive at an allocation scheme quickly.

Of course, for time-sensitive tasks, the transmission delay alone may make offloading infeasible, thus limiting the effectiveness of any offloading-based strategy. When the MEC server becomes a bottleneck due to limited computing resources or high contention from multiple users, the performance gain of our algorithm diminishes compared to ideal scenarios. These analyses help us to identify the practical scope of our method. It performs best in environments with moderate to high resources and heterogeneous task requirements. This is particularly true when local computation capacity is limited, but the MEC server offers sufficient processing power.

Extensive experiments have further validated the limited energy and time overhead of the proposed algorithm, makes it potentially suitable for deployment on energy-constrained devices and base stations.

VI. CONCLUSION

To improve the long-term task execution success rate of AIoT, the multi-slot edge-terminal collaborative computing offloading scheme is proposed. In our scheme, the long-term offloading optimization problem is simplified to the single-slot partial offloading optimization problem by using the Lyapunov optimization. Through numerous simulation experiments, the system cost increases with the time slots, the amount of data, the tasks number of per device and the total number of tasks. We found that by indicator weights $\alpha_i(t)$, CODET can effectively balance the computation delay and energy consumption, reduce the system offload costs, and improve the AIoT service quality. When the total number of tasks and the time slots increase, the execution success rate tends to stabilize. Numerical statistics show that CODET performs well in terms of the terminal task queue, average task execution success rate, and average system offloading cost. The complexity analysis and extensive experiments are provided to show that the algorithm has low time and energy overhead, making it suitable for implementation on devices with limited resources.

ACKNOWLEDGMENTS

This work was supported in part by the Joint fund of Equipment Pre-Research and Ministry of Education under Grant 8091B032131, in part by the Aeronautical Science Foundation of China under Grant 2020Z066050001, and in part by the Fundamental Research Funds for the Central Universities under Grant N25GFZ015.

REFERENCES

[1] S. Zhang, N. Yi, and Y. Ma, "A survey of computation offloading with task types," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 8313–8333, 2024.

[2] B. Lin, J. Weng, X. Chen, Y. Ma, and C.-H. Hsu, "A game-based computation offloading with imperfect information in multi-edge environments," *IEEE Transactions on Services Computing*, vol. 18, no. 1, pp. 1–14, 2025.

[3] C. Sun, X. Li, C. Wang, Q. He, X. Wang, and V. C. M. Leung, "Hierarchical deep reinforcement learning for joint service caching and computation offloading in mobile edge-cloud computing," *IEEE Transactions on Services Computing*, vol. 17, no. 4, pp. 1548–1564, 2024.

[4] R. Tang, R. Zhang, Y. Xu, and C. Yuen, "Deep reinforcement learning-based resource allocation for multi-uav-assisted full-duplex wireless-powered iot networks," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2024.

[5] L. Liu, Z. Zhang, H. Zhang, Y. Zhang, and Y. Xu, "Two-timescale dynamic resource management in smart-grid powered heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 23, no. 4, pp. 2681–2695, 2024.

[6] H. Liao, Z. Zhou, Z. Jia, Y. Shu, M. Tariq, J. Rodriguez, and V. Frascolla, "Ultra-low aoi digital twin-assisted resource allocation for multi-mode power iot in distribution grid energy management," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3122–3132, 2023.

[7] W. Fan, "Blockchain-secured task offloading and resource allocation for cloud-edge-end cooperative networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 8, pp. 8092–8110, 2024.

[8] C. Ling, W. Zhang, H. He, R. Yadav, J. Wang, and D. Wang, "Qos and fairness oriented dynamic computation offloading in the internet of vehicles based on estimate time of arrival," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 7, pp. 10554–10571, 2024.

[9] M. Chen, X. Gong, and Y. Cao, "Delay-optimal distributed edge computation offloading with correlated computation and communication workloads," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5846–5857, 2023.

[10] H. Maleki, M. Başaran, and L. Durak-Ata, "Handover-enabled dynamic computation offloading for vehicular edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 9394–9405, 2023.

[11] X. Sui, Z. Jiang, Y. Lyu, R. Fan, H. Hu, and Z. Liu, "Integrating convex optimization and deep learning for downlink resource allocation in leo satellites networks," *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[12] J. Zheng, Y. Pan, S. Jiang, Z. Chen, and F. Yan, "A federated learning and deep q-network based cooperative resource allocation algorithm for multi-level services in mobile edge computing networks," *IEEE Transactions on Cognitive Communications and Networking*, 2023.

[13] H. Liao, Z. Zhou, N. Liu, Y. Zhang, G. Xu, Z. Wang, and S. Mumtaz, "Cloud-edge-device collaborative reliable and communication-efficient digital twin for low-carbon electrical equipment management," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1715–1724, 2023.

[14] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for internet of vehicles: Model and algorithm," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 5913–5925, 2021.

[15] X. Zhang, J. Liu, R. Zhang, Y. Huang, J. Tong, N. Xin, L. Liu, and Z. Xiong, "Energy-efficient computation peer offloading in satellite edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 3077–3091, 2024.

[16] P. Teymoori and A. Boukerche, "Dynamic multi-user computation offloading for mobile edge computing using game theory and deep reinforcement learning," in *ICC 2022-IEEE International Conference on Communications*, pp. 1930–1935, IEEE, 2022.

[17] Y. Zhang, C. Chen, H. Zhu, Y. Pan, and J. Wang, "Latency minimization for mec-v2x assisted autonomous vehicles task offloading," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 3, pp. 4917–4932, 2025.

[18] F. Tang, H. Hofner, N. Kato, K. Kaneko, Y. Yamashita, and M. Hangai, "A deep reinforcement learning-based dynamic traffic offloading in space-air-ground integrated networks (sagin)," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 276–289, 2021.

[19] G. Qu, H. Wu, R. Li, and P. Jiao, "Dmro: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.

[20] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985–1997, 2020.

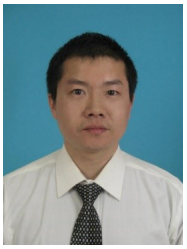
[21] S. Su, P. Yuan, and Y. Dai, "Reliable computation offloading of dag applications in internet of vehicles based on deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 2, pp. 2116–2128, 2025.

[22] A. Hazra, P. K. Donta, T. Amgoth, and S. Dustdar, "Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial iot applications," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 3944–3953, 2022.

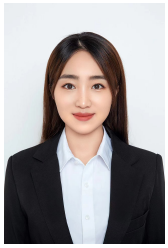
[23] S. Dong, Y. Xia, and J. Kamruzzaman, "Quantum particle swarm optimization for task offloading in mobile edge computing," *IEEE Transactions on Industrial Informatics*, 2022.
[24] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, 2020.
[25] J. Yuan, Y. Xiang, Y. Deng, Y. Zhou, and G. Min, "Upoa: A user preference based latency and energy aware intelligent offloading approach for cloud-edge systems," *IEEE Transactions on Cloud Computing*, 2022.
[26] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2017.
[27] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, "Pora: Predictive offloading and resource allocation in dynamic fog computing systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 72–87, 2019.
[28] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4968–4977, 2020.
[29] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2d-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for mec," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 4858–4873, 2021.
[30] M. Guo, W. Wang, X. Huang, Y. Chen, L. Zhang, and L. Chen, "Lyapunov-based partial computation offloading for multiple mobile devices enabled by harvested energy in mec," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 9025–9035, 2021.
[31] Z. Tong, J. Cai, J. Mei, K. Li, and K. Li, "Dynamic energy-saving offloading strategy guided by lyapunov optimization for iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19903–19915, 2022.
[32] H. Jiang, X. Dai, Z. Xiao, and A. K. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Transactions on Mobile Computing*, 2022.
[33] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7519–7537, 2021.
[34] S. Suman, F. Chiariotti, Č. Stefanovic, S. Došen, and P. Popovski, "Statistical characterization of closed-loop latency at the mobile edge," *IEEE Transactions on Communications*, vol. 71, no. 7, pp. 4391–4405, 2023.
[35] W. Yuan and K. Nahrstedt, "Energy-efficient cpu scheduling for multimedia applications," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 3, pp. 292–331, 2006.
[36] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.
[37] D. Han, W. Chen, B. Bai, and Y. Fang, "Offloading optimization and bottleneck analysis for mobile cloud computing," *IEEE Transactions on Communications*, vol. 67, no. 9, pp. 6153–6167, 2019.
[38] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2d-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for mec," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 4858–4873, 2021.
[39] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2019.
[40] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7519–7537, 2021.

VII. Biography Section

Jing Wang received the M.E. degree in Electronic Science and Technology from Yanshan University, Qinhuangdao, China, in 2020. She is currently working toward the Ph.D. degree in the School of Computer Science and Engineering at Northeastern University, Shenyang, China. Her research interests include industrial Internet of Things (IoT), edge computing, digital twin and health monitoring. (E-mail: 2110651@stu.neu.edu.cn).



Yuhuai Peng received the Ph.D. degree in communication and information systems from Northeastern University, Shenyang, China, in 2013. He is currently a professor with the Department of Communications and Electronic Information, Northeastern University. His research interests include Internet of Things (IoT), Cyber-Physical Systems (CPS), intelligent information networks, industrial communication networks, edge computing, and Prognostics and Health Management (PHM). (E-mail: pengyuhuai@mail.neu.edu.cn).



Xinyu Zhang received the B.S. degree in Communication Engineering from Changchun University of Science and Technology, Changchun, China, in 2020. She is currently working toward the M.E. degree in the School of Computer Science and Engineering at Northeastern University, Shenyang, China. Her research interests Resource Allocation and IoT. (E-mail: zhangxinyu98a@163.com).

Lei Liu (Member, IEEE) received the B.Eng. degree in electronic information engineering from Zhengzhou University, Zhengzhou, China, in 2010, and the M.Sc. and Ph.D degrees in communication and information systems from Xidian University, Xian, China, in 2013 and 2019, respectively. From 2013 to 2015, he was employed by a subsidiary of China Electronics Corporation. From 2018 to 2019, he was supported by China Scholarship Council to be a visiting Ph.D. student with the University of Oslo, Oslo, Norway. He is currently an Associate Professor with the Guangzhou Institute of Technology, Xidian University. His research interests include vehicular ad hoc networks, intelligent transportation, mobile-edge computing, and Internet of Things. (E-mail: tianjiaoliulei@163.com).

Shahid Mumtaz (Senior Member, IEEE) is an IET Fellow, the IEEE ComSoc Lecturer, and an ACM Distinguished Speaker. He has authored 4 technical books, 12 book chapters, and more than 300 technical articles (more than 200 IEEE Journals/ Transactions, more than 100 conferences, and two IEEE best paper awards) in mobile communications. Dr. Mumtaz is the recipient of the NSFC Researcher Fund for Young Scientist in 2017 from China and the IEEE ComSoC Young Researcher Award in 2020. He was awarded an Alain Bensoussan Fellowship in 2012. He is the Founder and EiC of IET Journal of Quantum Communication, the Vice Chair of Europe/Africa Region—IEEE ComSoc: Green Communications & Computing society and the IEEE standard on P1932.1: Standard for Licensed/Unlicensed Spectrum Interoperability in Wireless Mobile Networks. (E-mail: shahid.mumtaz@ntu.ac.uk)

Mohsen Guizani (M'89–SM'99–F'09) received the BS (with distinction), MS and PhD degrees in Electrical and Computer engineering from Syracuse University, Syracuse, NY, USA in 1985, 1987 and 1990, respectively. He is currently a Professor of Machine Learning and the Associate Provost at Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, UAE. Previously, he worked in different institutions in the USA. His research interests include applied machine learning and artificial intelligence, Internet of Things (IoT), intelligent autonomous systems, smart city, and cybersecurity. He was elevated to the IEEE Fellow in 2009 and was listed as a Clarivate Analytics Highly Cited Researcher in Computer Science in 2019, 2020 and 2021. Dr. Guizani has won several research awards including the "2015 IEEE Communications Society Best Survey Paper Award", the Best ComSoc Journal Paper Award in 2021 as well five Best Paper Awards from ICC and Globecom Conferences. He is the author of ten books and more than 800 publications. He is also the recipient of the 2017 IEEE Communications Society Wireless Technical Committee (WTC) Recognition Award, the 2018 AdHoc Technical Committee Recognition Award, and the 2019 IEEE Communications and Information Security Technical Recognition (CISTC) Award. He served as the Editorin-Chief of IEEE Network and is currently serving on the Editorial Boards of many IEEE Transactions and Magazines. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer. (E-mail: mguizani@ieee.org)

Schahram Dustdar (Fellow, IEEE) is a Full Professor of Computer science heading the Research Division of Distributed Systems at the TU Wien, Austria. He is also part-time ICREA research professor at UPF Barcelona. He was the founding Co-Editor-in-Chief of ACM Transactions on Internet of Things (ACM TIoT) and is Editor-in-Chief of Computing (Springer). He is an Associate Editor of IEEE Transactions on Services Computing, IEEE Transactions on Cloud Computing, ACM Transactions on the Web, and ACM Transactions on Internet Technology, as well as on the editorial board of IEEE Internet Computing and IEEE Computer. Dustdar is a Recipient of the ACM Distinguished Scientist Award (2009), the ACM Distinguished Speaker Ward (2021), the IBM Faculty Award (2012), and an Elected Member of the Academia Europaea: The Academy of Europe, where he was Chairman of the Informatics Section for many years, as well as an IEEE Fellow. In 2021 Dustdar is elected President of the International Artificial Intelligence Industry Alliance (AIIA). (E-mail: dustdar@dsg.tuwien.ac.at)