IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 November 2022, accepted 5 December 2022, date of publication 12 December 2022,
date of current version 19 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3228403

RESEARCH ARTICLE

A Trade-Off Task-Offloading Scheme in Multi-User Multi-Task Mobile Edge Computing

RUIXIA LI[1,2], CHIA SIEN LIM[2], MUHAMMAD EHSAN RANA[2], AND XIANCUN ZHOU[1]
[1]School of Electronic and information Engineering, West Anhui University, Lu'an 237012, China
[2]Graduate School of Technology, Asia Pacific University of Technology and Innovation, Kuala Lumpur 57000, Malaysia

Corresponding author: Ruixia Li (tp063070@mail.apu.edu.my)

This work was supported in part by the Key Research and Development Technology Project of Anhui Province under Grant 2022107020005, and in part by the Major Project of Natural Science Research in Anhui Universities under Grant KJ2021ZD0116.

ABSTRACT Mobile edge computing (MEC) is a novel technique that can reduce computational burden of local terminal devices by tasks offloading, which emerges as a promising paradigm to provide computing capabilities near mobile users. Furthermore, the computing power consumption and battery capacity in the terminal devices is generally limited in MEC, which directly affect the quality of service (QoS), which brings new challenge in achieving the stability of the system. The trade-off between delay and energy consumption of task offloading process under multi-user MEC scenario is in general a complicated problem that highly correlates to the computation offloading decisions of computation tasks, i.e., whether or not to offload a task for edge execution. To address the above issues, the task offloading decision to solve the queuing problem of tasks to be processed in the local terminals was being considered, with an application of Lyapunov theory to ensure the queue stability in this paper. Then, a trade-off model was formulated to minimize the delay and energy consumption to achieve a minimum execution cost. Moreover, an improved Dynamic Niche-based Self-organizing Learning Algorithm was presented to accelerate the speed of the search process to gain an optimal task offloading scheme. The simulation results provided supporting evidence that the proposed optimal scheme IDO could achieve a lower average energy consumption than LFO and a lower average execution delay than EFO under the scenarios of the waiting queue length varies within [10, 50], the number of users varies within [4, 20], and the number of tasks varies within [20, 100].

INDEX TERMS Lyapunov theory, mobile edge computing, task offloading, evolutionary algorithm.

I. INTRODUCTION

Recently, with the popularity of Internet and the growth of mobile users, mobile communication technology has developed rapidly. The functions of terminal devices are constantly improved, and many emerging interactive experience applications, such as augmented reality (AR), virtual reality (VR), video-on-demand, social networking services, motion sensing game and so on, are booming, leading to an explosion in the volume of mobile data. According to Cisco statistics, mobile data volume will reach 15% of global IP traffic by 2021 [1]. Mobile intelligent devices (mobile phones, tablet computers, etc.), the number of which is even staggering, have gradually become necessities in people's daily life. With the increase of data volume and the updating of applications, the performance and computing capability of terminal devices are required to be higher. Nevertheless, due to their limited size and battery capacity, there is not enough computing capability for certain service programs (such as AR and immersive experiences) and delay requirements cannot also be met.

Cloud computing has abundant storage space and powerful computing capability. Offloading tasks to remote cloud can provide users with rich resources and excellent

The associate editor coordinating the review of this manuscript and approving it for publication was Hosam El-Ocla.

FIGURE 1. An illustration of the MEC architecture.

experience [2], and also help alleviate the delay and energy consumption problems caused by lacking of computing capability in mobile devices. However, moving large amounts of data to the cloud will result in a huge cost in network transmission. It is very difficult to reduce network delay between remote cloud servers and user devices by leveraging existing infrastructure.

In 2014, the European Telecommunications Standards Institute (ETSI) proposed Mobile Edge Computing (MEC) to reduce network delay between remote cloud servers and user devices by leveraging on existing infrastructure. Based on MEC technology, MEC servers are deployed at the edge of mobile networks to sink cloud storage and computing resources to the edge of networks, which form the mobile edge computing networks providing computing capability and service environment for application developers and service providers. As shown in Fig. 1 adapted from [3], the MEC system model mainly includes mobile user devices, access points, base stations, and mobile edge servers. Mobile devices communicate with base stations through wireless networks. Network operators typically deploy MEC servers near to a base station. A MEC server is a small server similar to a remote cloud that has a range of functions such as computing and storage as those in a cloud computing center.

When the edge computing nodes are deployed near to user devices, environmental requirements are usually considered to ensure them a long-term stable operation. In practical applications, edge computing can be deployed independently and plays an important role in real-time, short-period data processing and analysis as well as local decision-making scenarios, such as driverless cars, intelligent factories and so on. Users offload tasks to MEC servers through wireless channels so as to reduce the processing delay, the energy consumption of mobile devices, and save the cost of task processing.

Some application scenarios, such as augmented reality, virtual reality, and self-driving car networking, put forward high requirements for network delay, bandwidth, computing and storage. In addition to that, the computing power consumption and battery capacity in the terminal devices is generally limited, which directly affect the quality of service (QoS). To guarantee the system stability, reduce the packet loss rate, and improve the users' satisfaction, it is important to compromise the delay and energy according to the practical needs of different applications in the research on task offloading decision in MEC.

One of the current challenges in MEC is how to guarantee system stability and lower packet loss rate in the process of offloading tasks while achieving the trade-off between delay and energy under the multi-user multi-task scenario.

The main contributions in this paper are listed as follows.

a) Considering the trade-off between delay and energy consumption of task offloading process under multi-user MEC scenario, we propose a task offloading optimization scheme to effectively utilize the computing resources of mobile edge servers, reduce delay and energy consumption of users' tasks optimally, and improve users' satisfaction degree, which is close to the actual application scenarios and makes performance analysis exactly substantially.

b) Justification of Lyapunov queue stability theory is provided in scheduling task queue and reducing users'

packet loss rate when users have massive amounts of tasks exceeding terminal loads to offload.

c) A trade-off optimization between average response time and average power is realized mathematically. Moreover, we also consider queuing delay caused by guaranteeing system stability which can reduce the packet loss rate significantly and thus provide an excellent user experience.

d) Since the above problem falls into the NP-hard scope [4], an improved Dynamic Niche-based Self-organizing Learning Algorithm (DNSLA) is proposed to search for a trade-off solution between delay and energy consumption of task offloading. It is demonstrated at section VII that our improved DNSLA based offloading scheme (IDO) has better comprehensive performance when compared to ''local first'' offloading scheme (LFO) and ''edge first'' offloading scheme (EFO).

The rest of the paper is organized as follows. Section II introduces the recent related work of computing offloading in detail. In section III, the mathematical model of computing offloading problem is built. Section IV describes the problem in detail and transforms it into the optimal problem with the minimum value. Section V carries out queue stability via Lyapunov optimization theory. Section VI introduces our proposed the improved DNSLA based task offloading scheme IDO. Section VII evaluates our proposed scheme using simulation experiments and analyses the computational complexity. Finally, conclusions are drawn in Section VIII.

II. RELATED WORK

The main goal of computing offloading lies in offloading the computing tasks in terminal devices to appropriate destinations according to task offloading algorithms and thus achieving a minimum cost of the whole system. Based on the review below, the research work of MEC mainly focused on the optimization of delay and energy consumption caused by data transmission and task processing during offloading process of computing tasks. [5] proposed a GA-based multi-edge and cloud collaborative computing offloading model. This computing offloading solution combines local edge and remote edge to perform task offloading; it used a GA to achieve the minimum system cost considering both delay and energy consumption at the same time; the GA improved the resource utilization of the edge node, minimizes the data flow between the edge and the cloud and relieves the pressure of core networks between the edge and the cloud. [6] presented a resource allocation and offloading decision algorithm based on the divided time slot to improve the resource utilization of edges and relieve the potential congestion of core networks by reducing the cloud-edge communication traffic. However, these work did not take the stability of task queues into account and also ignored the tradeoff between energy consumption and delay. [7] formulated the task offloading problem as minimizing the long-term response delay of the MEC system under the constraint of long-term average leasing cost and proposed the single-slot service chain caching and task offloading algorithm which integrates computation and utility-based caching algorithm, cross-server caching algorithm and relative-distance-based task offloading algorithm to reduce the long-term average response delay of the MEC system while keeping the long-term average leasing cost at a relative low level from the perspective of application service providers. [8] investigated the energy-delay tradeoff for dynamic offloading in an MEC system with energy harvesting devices and proposed an online dynamic Lyapunov optimization based offloading algorithm to make decisions for tasks assignment to solve a problem of minimizing energy consumption with the buffer queue and battery energy level stable. [9] considered a general mobile cloud computing system consisting of multiple users and one remote cloud server, where each user has multiple independent tasks. To minimize a weighted total cost of energy, computation, and the delay of all users, the proposed multi-user multi-task offloading algorithm uses separable semidefinite relaxation and binary recovery to jointly compute the offloading decision and communication resource allocation.

To address the problem of multi-user dependent task offloading, [10] established a user dependent task model based on the comprehensive consideration of delay and energy consumption and proposed a multi-user task offloading strategy based on delay acceptance to solve the problem of minimizing energy consumption under delay constraints, which can solve the problem of multi-user task offloading using a two-step non-dominated single user optimal offloading strategy and adjustment strategy to solve resource competition. [11] considered a MEC enabled multi-cell wireless network where each base station (BS) was equipped with a MEC server that assists mobile users in executing computation-intensive tasks via task offloading and studied the problem of joint task offloading and resource allocation in order to maximize the users' task offloading gains, which was formulated as a mixed integer nonlinear program that involved jointly optimizing the task offloading decision, uplink transmission power of mobile users, and computing resource allocation at the MEC servers. [12] developed a unified MEC design framework with joint energy beamforming, offloading, and computing optimization in emerging wireless powered multiuser MEC systems and proposed an efficient wireless powered multiuser MEC design by considering the latency-constrained computation, for which the access point minimizes the total energy consumption subject to the users' individual computation latency constrains. Finally, the optimal solution is obtained in a semi-closed form by adopting the Lagrange duality method. [13] investigated the optimal computation offloading and time allocation for multi-access MEC, in which the mobile user (MU) uses non-orthogonal multiple access (NOMA) to offload the computation-workloads to a group of edge-servers

simultaneously, aiming at minimizing the overall-delay for the MU to complete its computation requirement, by jointly optimizing the MU's offloaded workloads to the edge-servers and the NOMA transmission-time. The authors proposed an efficient algorithm to compute the MU's optimal offloading solution for the single MU's offloading and the distributed algorithm to find the MUs' optimal offloading solution as well as the MUs' NOMA transmission time for the multi-MUs' offloading. [14] evaluated whether advanced transmission techniques proposed for 5G can improve the performance of mobile edge computing, and investigated the possible gain of the joint allocation of radio and computational resources to minimize the total transmission energy consumption for computational offloading under individual delay constraint by considering the allocation of three different resources, including resource blocks, transmission power and computational units.

[15] proposed a distributed dynamic heterogeneous task offloading methodology algorithm by exploiting game theory and Lyapunov optimization, which could achieve heterogeneous control and allocation of computation resources by dynamic quote price mechanism. In order to solve the problems of small coverage density and hotspot overload of the central node in the current MEC network, [16] designed a distributed edge computing architecture for ultra-dense networks and proposed a multi-base station game offloading algorithm to minimize the system overhead. In the proposed algorithm, the Lagrange multiplier method was used to solve the problem of computing resource allocation, and then the matching game theory was exploited to obtain the optimal offloading strategy, so that the mutual benefits of both users and MEC servers could be maximized. [17] studied the multi-user computation offloading problem in MEC from a behavioral perspective and applied the framework of prospect theory to model mobile device users' realistic behavior when making the computation offloading decision. The authors cast the users' decision making of whether to offload or not as a prospect theory-based non-cooperative game and proposed a distributed computation offloading algorithm to achieve the Nash equilibrium. [18] investigated the use of cooperative communications in computation offloading for a wireless power transfer (WPT) MEC system, in which an access point (AP) acts as an energy source via WPT and serves as an MEC server to assist two near-far mobile devices to complete their computation-intensive latency-critical tasks. Joint power and time allocation for cooperative computation offloading was considered based on a block-based harvest-then-offload protocol, with the aim to minimize the transmit energy of the AP for completing the computation tasks of the two users. [19] studied the joint computation offloading and transmission scheduling for delay-sensitive applications in mobile edge computing and a queueing model to characterize the dynamic management of the system with potential network uncertainties. By considering tradeoffs between local and edge computing, wireless features and non-cooperative game behaviors of smart mobile users, the authors proposed a multi-user computation offloading and transmission scheduling mechanism to jointly determine the computation offloading scheme, the transmission scheduling discipline and the pricing rule.

To address the tradeoff issue within the systems executing all applications in fog nodes in terms of average response time and average cost, [20] presented an online algorithm called unit-slot optimization, which is a quantified near-optimal online solution based on the technique of Lyapunov optimization to balance the three-way tradeoff among average response time, average cost and average number of application loss. [21] discussed multi-user multi-task offloading scheduling schemes in a renewable mobile edge cloud system. The authors proposed the centralized and distributed greedy maximal scheduling algorithms to determine the energy harvesting strategy, a set of offloading requests and a sub-set of wireless devices to compute the workload for the admitted offloading request to maximize the overall system utility. To exploit the computing capacity at the green mobile edge cloud thoroughly, the scheduling algorithms match the offloading energy consumption at the mobile edge cloud to its harvestable energy. [22] studied the optimal joint energy and task allocation problem for a single-user wireless powered MEC system with dynamic task arrivals over time to minimize the transmission energy consumption at the energy transmitter subject to the energy/task causality and task completion constraints at the user within a finite horizon of multiple slots. Leveraging the convex optimization techniques, the authors obtained the well-structured optimal offline solutions with non-causal channel/task state information known a-priori, in the scenarios with static and time-varying channels, respectively, and further proposed heuristic online joint energy and task allocation designs with only causal channel/task state information available. [23] studied multi-server multi-user multi-task computation offloading for MEC networks, with the aim to guarantee the network's quality of service and to minimize wireless devices' energy consumption. By formulating different real-time task offloading decisions as static optimization problems, the authors investigated a linear programing relaxation-based algorithm to approximate the optimum and further investigated the heterogeneous distributed deep learning-based offloading algorithm for MEC networks by taking advantage of deep reinforcement learning.

To sum up, there are still many problems to be solved under the scenario of multi-user and multi-task offloading over mobile edge networks. Our work focus on how to realize efficient computing offloading and resource allocation while ensuring system performance and guaranteeing end users' quality of service (QoS). In the paper, system performance includes availability, reliability, throughput, response time, task blocking probability and other metrics. QoS parameters considered in our manuscript mainly refer to delay and packet loss rate.

The main advantages of our proposed task-offloading scheme IDO in this paper are listed as follows.

a) Formalization of the trade-off optimization between average response time and average power consumption during the multi-task offloading process under multi-user MEC scenario. The result is expressed through Equation (34).
b) Leverage Lyapunov queue stability theory by controlling the Lyapunov drift and adjusting the value of Lyapunov function in the scheduling task queue to guarantee system stability. The result is expressed through Equations (57) and (58).
c) An improved Dynamic Niche-based Self-organizing Learning Algorithm carries out a self-organizing learning process consisting of global learning, neighborhood learning and self-learning where each population in the ecosystem studies and searches synchronously, and exchanges individual dynamically one another to accelerate the speed of searching process to gain the optimal solution. The algorithm is presented in Algorithm 2; the comparative results with other existing scheme are depicted in Figure 3, 5 and 6 will provide evidence that the proposed Algorithm outperforms against the selected schemes.

III. COMPUTING OFFLOADING MODEL

We consider a scenario of providing computing offloading services by a two-layer system consisting of MEC edge servers and the local devices in their coverage areas. We solve the problem whether the tasks generated by local devices need to be offloaded to the edge servers under the condition of guaranteeing system performance. The local terminal node can offload tasks to edge nodes within its direct communication range to expand its computing capability and reduce energy consumption. The data transmission between the terminal devices and the MEC base stations is carried out by wireless communication links. Different wireless links are transmitted by orthogonal channels and will not interfere with each other.

Running process of the system is divided into many time slots in the time domain. The set of time slot is represented by $t = \{1, 2, \ldots, T\}$, and the length of each time slot is l. The terminal layer composed of various terminal devices generates task data to be processed in each timeslot. The data are generated in timeslot t-1, and then the resource allocation module decides the offloading strategy in timeslot t. Therefore, we assume that the data will begin execution at time slot t. The offloading strategy decides that whether executing the tasks from terminal devices locally or offloading them to edge nodes for processing. Based on the definition of discrete time system [24], the task processing process of terminal device can be modelled into a discrete time system, which has the following characteristics:

- In each time slot, the tasks from each terminal device are randomly generated and assumed to be independent of each other;
- The number of tasks from the terminal devices in each time slot is subject to Independent and Identical Distribution (I.I.D);
- The number of CPU cycles and data size required by task execution also obey I.I.D.

In particular, different tasks require different execution timing, that is, some tasks may not be completed in one timeslot and will continue to be processed in the next timeslot. Therefore, the queuing delay must be counted into computing task delay if a task in the last time slot has not been completed. The queuing model of the system is displayed in Fig.2.

A. MATHEMATICAL FORMULATION OF TASK OFFLOADING MODEL

The task $W_{(i,j)}^{(t)}$ denotes the j-th task from terminal node M_i in time slot t. The $W_{(i,j)}^{(t)}$ can further be described as $\{D_{(i,j)}^{(t)}, C_{(i,j)}^{(t)}\}$ where $D_{(i,j)}^{(t)}$ represents packet size of task $W_{(i,j)}^{(t)}$ and $C_{(i,j)}^{(t)}$ represents the number of CPU cycles required for processing task $W_{(i,j)}^{(t)}$. We assume that an edge server covers M mobile terminal nodes and arrival process of tasks into the terminal device obeys the Poisson distribution. Let $\lambda_i(t)$ denote the arrival rate of tasks into terminal device M_i at time slot t. In each time slot, the resource allocation module provides offloading strategies that determine which tasks are processed in terminal devices locally or offloaded to the edge servers. Let $X_i^{local}(t)$ and $X_i^{mec}(t)$ denote the task sets processed in the local and edge node in time slot t respectively; $N_i^{local}(t)$ and $N_i^{mec}(t)$ denote the number of tasks in $X_i^{local}(t)$ and $X_i^{mec}(t)$ respectively. Based on the above expressions, we conclude the relationship in (1).

$$N_i^{local}(t) + N_i^{mec}(t) = \lambda_i(t) \tag{1}$$

In addition to the computing delay of tasks in devices, the queuing delay may arise when the task $W_{(i,j)}^{(t)}$ is executed locally. Furthermore, the transmission delay and energy may also occur when the tasks are offloaded from local nodes to the edge nodes. Therefore, the delay $T_{(i,j)}(t)$ and energy consumption $E_{(i,j)}(t)$ of task $W_{(i,j)}^{(t)}$ can be expressed as follows.

$$T_{(i,j)}(t) = I_{(i,j)}^{l}(t)T_{(i,j)}^{l}(t) + I_{(i,j)}^{m}(t)T_{(i,j)}^{m}(t) \tag{2}$$

$$E_{(i,j)}(t) = I_{(i,j)}^{l}(t)E_{(i,j)}^{l}(t) + I_{(i,j)}^{m}(t)E_{(i,j)}^{m}(t) \tag{3}$$

where $I_{(i,j)}^{l}(t) + I_{(i,j)}^{m}(t) = 1$ and they indicate the target devices of the task offloading in the system. When the task $W_{(i,j)}^{(t)}$ is processed locally, $I_{(i,j)}^{l}(t) = 1$; otherwise $I_{(i,j)}^{l}(t) = 0$. When task $W_{(i,j)}^{(t)}$ is processed in the edge server, $I_{(i,j)}^{m}(t) = 1$; otherwise $I_{(i,j)}^{m}(t) = 0$. Table 1 below lists the main notations and their meanings in this paper.

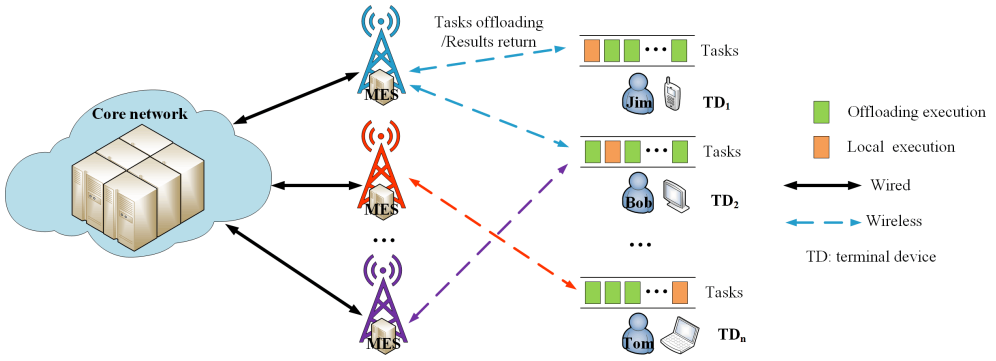


FIGURE 2. The multi-user and multi-task offloading model.

B. EXECUTING TASKS AT LOCAL NODES

1) LOCAL DELAY

Computing delay $Te^{l}_{(i,j)}(t)$ of executing the task $W^{(t)}_{(i,j)}$ locally is calculated in (4). Since the local processing ability is limited, we formulate the task processing in each terminal node into a real-time task queuing model based on [25]. Task queue length $Q_i(t+1)$ in terminal node M_i can be calculated in (5) where $Q_i(0) = 0$. Waiting delay $Tw^{l}_{(i,j)}(t)$ of the task $W^{(t)}_{(i,j)}$ processed locally is calculated in (6). Accordingly, the total delay $T^{l}_{(i,j)}(t)$ of the task $W^{(t)}_{(i,j)}$ processed locally is derived in (7).

$$Te^{l}_{(i,j)}(t) = C^{(t)}_{(i,j)}/f_{i,l} \tag{4}$$

$$Q_i(t+1) = \max[Q_i(t) - \mu_i(t), 0] + N^{local}_i(t) \tag{5}$$

$$Tw^{l}_{(i,j)}(t) = \sum_{j=1}^{Q_i(t)} \left(C^{(t)}_{(i,j)}/f_{i,l} \right) \tag{6}$$

$$T^{l}_{(i,j)}(t) = Te^{l}_{(i,j)}(t) + Tw^{l}_{(i,j)}(t) \tag{7}$$

2) LOCAL ENERGY CONSUMPTION

The power consumption $P_{i,0}$ of local node i is a superlinear function of execution frequency $f_{i,l}$, namely $P_{i,0} = \alpha(f_{i,l})^{\gamma}$, usually $\alpha = 10^{-11}$, $\gamma = 2$ according to [26]. When tasks are processed locally, power consumptions is mainly derived from CPU operations. Therefore, the energy consumption $E^{l}_{(i,j)}(t)$ of executing task $W^{(t)}_{(i,j)}$ locally at the time slot t can be calculated in (8).

$$E^{l}_{(i,j)}(t) = P_{i,0} Te^{l}_{(i,j)}(t) = \alpha(f_{i,l})^{\gamma} Te^{l}_{(i,j)}(t) \tag{8}$$

C. EXECUTING TASKS AT EDGE SERVERS

A typical process of offloading tasks to edge nodes consists of three steps: (1) The local nodes upload the tasks to the edge nodes; (2) The edge nodes allocates computing resources to the offloaded tasks; (3) The edge nodes return the computing results to the local nodes. Without loss of generality, we ignore the cost of the last step, so the cost of offloading tasks to edge nodes is composed of the following two parts: (1) The transmission time and energy consumption of offloading tasks to edge nodes; (2) Processing time of the offloaded tasks on edge nodes.

TABLE 1. Notations definition list.

Notation	Meaning
M	The number of local nodes
$W^{(t)}_{(i,j)}$	the task j in local node i at time slot t
$D^{(t)}_{(i,j)}$	the size of data packages in $W^{(t)}_{(i,j)}$
$C^{(t)}_{(i,j)}$	the number of CPU cycles processing $W^{(t)}_{(i,j)}$
$\lambda_i(t)$	arrival rate of tasks in local node i at time slot t
$X^{local}_i(t)$	the set of tasks assigned to local node i at time slot t
$X^{mes}_i(t)$	the set of tasks assigned to edge node i at time slot t
$N^{local}_i(t)$	the number of tasks in $X^{local}_i(t)$
$N^{mes}_i(t)$	the number of tasks in $X^{mes}_i(t)$
$I_{(i,j)}(t)$	identifier of task scheduling location
$Q_i(t)$	the number of tasks queuing in local node i at time slot t
$f_{i,l}$	the number of CPU cycles in unit time for local node i
$f_{i,m}$	the number of CPU cycles in unit time for edge node i
$\mu_i(t)$	the number of tasks processed locally in local node i at time slot t
$P_{i,0}$	power consumption of local node i
W	user bandwidth
N_0	noise power
p^{up}_i	transmission power consumption
h_i	information gain
l	unit time slot length

1) DELAY IN TRANSMISSION OF TASKS

Data transmission rate $W^{(t)}_{(i,j)}$ of offloading tasks to edge nodes is calculated in (9), where $h_i = d_n^{-s}$ and usually $s = 4$, $d_n = [0, 50]$ according to [27]. Thus, the delay $Tt^{m}_{(i,j)}(t)$ of transmitting tasks from local nodes to edge nodes can be calculated in (10). Computing delay $Te^{m}_{(i,j)}(t)$ of executing the

task $W_{(i,j)}^{(t)}$ at the edge nodes is calculated in (11). Accordingly, the total delay $T_{(i,j)}^{m}(t)$ of offloading the task $W_{(i,j)}^{(t)}$ to execute at edge nodes is calculated in (12).

$$R_i = W \log_2(1 + p_i^{up} h_i / N_0) \tag{9}$$

$$Tt_{(i,j)}^{m}(t) = D_{(i,j)}^{(t)} / R_i \tag{10}$$

$$Te_{(i,j)}^{m}(t) = C_{(i,j)}^{(t)} / f_{i,m} \tag{11}$$

$$T_{(i,j)}^{m}(t) = Te_{(i,j)}^{m}(t) + Tt_{(i,j)}^{m}(t) \tag{12}$$

2) ENERGY CONSUMPTION

Our work focuses on analyzing the energy consumption and delay cost from the users' perspective. Therefore, if the task is offloaded to edge processing, the energy consumption will not be included. So energy consumption $E_{(i,j)}^{m}(t)$ of offloading tasks from local nodes to edge nodes can be calculated in (13), where we only consider the energy consumption on uploading tasks and ignore the energy consumption on executing tasks that is saved by the offloading strategy.

$$E_{(i,j)}^{m}(t) = P_i^{up} Tt_{(i,j)}^{m}(t) \tag{13}$$

D. AVERAGE DEALY AND ENERGY CONSUMPTION

According to (2)-(13), the average delay and average energy consumption of offloading tasks to edge nodes at the time slot t are respectively calculated as follows.

$$T_{avg}(t) = \sum_{i=1}^{M} \sum_{j=1}^{\lambda_i(t)} T_{(i,j)}(t) / \sum_{i=1}^{M} \lambda_i(t) \tag{14}$$

$$E_{avg}(t) = \sum_{i=1}^{M} \sum_{j=1}^{\lambda_i(t)} E_{(i,j)}(t) / \sum_{i=1}^{M} \lambda_i(t) \tag{15}$$

Further, long-term average expectation of the delay and energy consumption of offloading tasks to the edge node can be expressed as follows.

$$\overline{T_{avg}} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[T_{avg}(t)] \tag{16}$$

$$\overline{E_{avg}} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[E_{avg}(t)] \tag{17}$$

IV. PROBLEM DESCRIPTION AND TRANSFORMATION

A. PROBLEM DESCRIPTION

Minimizing the energy consumption at user's side, namely $\overline{E_{avg}}$ in (17), is our optimization objective. Simultaneously, in order to avoid heavy task loads in local nodes resulting in the serious queuing delay, the average waiting time of the task queues in local nodes meets the following condition.

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[T_{avg}(t)] < \infty \tag{18}$$

Lemma 1: if the queue $Q_i(t) = \{Q_1(t), Q_2(t), \ldots\ldots Q_M(t)\}$ remain stable, then the condition (18) holds.

Proof: Assume the maximum completion time of tasks at time slot t is D, according to the definition of system stability [28]:

$$Q_i(t+1) = \max[Q_i(t) - D, 0] + T_{avg}(t) \tag{19}$$

Thus the following relationship exists:

$$Q_i(t+1) \geq Q_i(t) - D + T_{avg}(t) \tag{20}$$

Take expectations of both sides in (20):

$$E[Q_i(t+1)] - E[Q_i(t)] \geq -D + E[T_{avg}(t)] \tag{21}$$

Expression (22) can be derived by accumulating the values at each time slot.

$$E[Q_i(T)] - E[Q_i(0)] \geq -TD + \sum_{t=0}^{T-1} E[T_{avg}(t)] \tag{22}$$

Dividing both sides of (22) by T:

$$\frac{E[Q_i(T)] - E[Q_i(0)]}{T} \geq -D + \frac{1}{T} \sum_{t=0}^{T-1} E[T_{avg}(t)] \tag{23}$$

Since $Q_i(0) = 0$, we arrive at

$$\frac{E[Q_i(T)]}{T} \geq -D + \frac{1}{T} \sum_{t=0}^{T-1} E[T_{avg}(t)] \tag{24}$$

Let $T \to \infty$, then

$$\lim_{T\to\infty} \frac{E[Q_i(T)]}{T} \geq -D + \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[T_{avg}(t)] \tag{25}$$

If the average rate of $Q_i(T)$ is stable, then $\lim_{T\to\infty} sup\,(E[Q_i(T)]/T) = 0$. So we can get

$$0 \geq -D + \lim_{T\to\infty} sup \frac{1}{T} \sum_{t=0}^{T-1} E[T_{avg}(t)] \tag{26}$$

Namely,

$$\lim_{T\to\infty} sup \frac{1}{T} \sum_{t=0}^{T-1} E[T_{avg}(t)] \leq D \tag{27}$$

Leveraging the task waiting queues, we transform the constraint condition (17) into a queue stability problem. Therefore, if we can ensure that the queue is stable, then the condition (18) holds and Lemma 1 is proved.

Based on (2), the vectors of scheduling position strategies for $\lambda_i(t)$ tasks at time slot t are shown as follows, where $\pi_{(i,j)}(t) = [I_{(i,j)}^{l}(t), I_{(i,j)}^{m}(t)]$ denotes the vector of scheduling position strategy for tasks $W_{(i,j_1)}^{(t)}, W_{(i,j_2)}^{(t)}, \cdots, W_{(i,j_n)}^{(t)}$.

$$\pi^{i}(t) = [\pi_{(i,1)}(t), \pi_{(i,2)}(t), \ldots, \pi_{(i,\lambda_i(t))}(t)] \tag{28}$$

For all the local nodes, the vectors of task scheduling position strategies can be represented as follows.

$$\pi(t) = [\pi^{(1)}(t), \ldots, \pi^{(i)}(t), \ldots, \pi^{(M)}(t)] \tag{29}$$

The $\overline{E_{avg}}$ in (17) depends on $\pi(t)$, so $E[E_{avg}(t)]$ can further be denoted as $E[E_{avg}(\pi(t))]$ and the task offloading problem aiming to minimize the energy consumption can be described as follows.

$$\begin{aligned}\min\ \overline{E_{avg}} &= \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E[E_{avg}(\pi(t))] \\ &= \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\frac{\sum_{i=1}^{M}\sum_{j=1}^{\lambda_i(t)}E_{(i,j)}(t)}{\sum_{i=1}^{M}\lambda_i(t)}, \\ &\forall t \in \{0,1,2,\ldots,\infty\}, \end{aligned} \tag{30}$$

$$S.t.\ (1), (18)$$

$$N_i^{local}(t) = \sum\nolimits_{i\in X_i^{local}(t)} I_{(i,j)}^{l}(t) \tag{31}$$

$$N_i^{mec}(t) = \sum\nolimits_{i\in X_i^{mec}(t)} I_{(i,j)}^{m}(t) \tag{32}$$

where (30) can be determined by (17) constraint (1) describes the service model of system, constraint (18) ensures system stability, constraints (31) and (32) are the task offloading strategy following $\pi(t)$.

The problem can further be equivalent to how to gain the optimal offloading strategy $\pi^*(t)$ under time series of $t = 0, 1, \ldots, \infty$ and minimize the expectation of energy consumption in (30), where $\pi^*(t) = [\pi^{(1)*}(t), \ldots \pi^{(i)*}(t), \ldots \pi^{(M)*}(t)]$.

B. PROBLEM TRANSFORMATION

The optimization problem of (30) can be converted to:

$$\begin{aligned} &min\ E[E_{avg}(\pi(t))], \quad \forall t \in \{0,1,2,\ldots,\infty\} \\ &S.t.\ (1), (18), (31), (32). \end{aligned} \tag{33}$$

Based on Lemma 1, the optimization problem in this paper can be transformed into:

$$\begin{aligned} &min\ E[E_{avg}(\pi(t))] = \sum_{i=1}^{M}(\sum_{j=1}^{\lambda_i(t)}E_{(i,j)}(t))/\sum_{i=1}^{M}\lambda_i(t), \\ &\forall t \in \{0,1,2,\ldots,\infty\} \\ &S.t.\ \overline{T_{avg}} \le D, \quad \forall i \in \{1,2,\ldots M\}, (1), (18), (31), (32). \end{aligned} \tag{34}$$

V. LYAPUNOV THEORY BASED STABILITY ANALYSIS

A. BOUNDING UNIT-SLOT LYAPUNOV DRIFT

In order to avoid a large number of tasks queuing at terminal nodes, the stability of queues should be guaranteed first by Lemma 1. Lyapunov optimization is a method to optimize system control. As in [29], we define a Lyapunov function, $L(Q_i(t)) = Q_i^2(t)/2$ as the measure of the overstocking degree in a task queue. Moreover, to indicate the change rate of Lyapunov function $L(Q_i(t))$, from a time slot to the next time slot, we define the conditional unit-slot Lyapunov drift as follows.

$$\Delta Q_i(t) = E[(L(Q_i(t+1)) - L(Q_i(t)))|Q_i(t)] \tag{35}$$

If all the task queues are relatively short, then the value of $L(Q_i(t))$ is relatively small. If a certain queue in the system is congested, then the value of $L(Q_i(t))$ can increase rapidly. The value of Lyapunov function can be maintained at a relatively low level by controlling the Lyapunov drift $\Delta Q_i(t)$.

Lemma 2: We can obtain an upper bound of the Lyapunov drift $\Delta Q_i(t)$ as shown in (36) according to the above dynamic relationship of the queues in the system.

$$\Delta Q_i(t) \le H + Q_i(t)E[-D + T_{avg}(t)|Q_i(t)] \tag{36}$$

$$H = [\max E[T_{avg}(t)^2] + D^2]/2 \tag{37}$$

Proof: From (35), we derive

$$\begin{aligned}\Delta Q_i(t) &= E[L(Q_i(t+1)) - L(Q_i(t))|Q_i(t)] \\ &= \frac{1}{2}E[Q_i^2(t+1) - Q_i^2(t)|Q_i(t)]\end{aligned} \tag{38}$$

Applying the definition of (19), (37) can be further expressed as below:

$$\begin{aligned}\Delta Q_i(t) = \frac{1}{2}E[[\max[Q_i(t) - D, 0] + T_{avg}(t)]^2 \\ - Q_i^2(t)|Q_i(t)]\end{aligned} \tag{39}$$

For any $Q_i(t) \ge 0, D \ge 0, T_{avg}(t) \ge 0$

$$\begin{aligned}&[\max[Q_i(t) - D, 0] + T_{avg}(t)]^2 \\ &\le Q_i^2(t) + T_{avg}(t)^2 + D^2 + 2Q_i(t)(T_{avg}(t) - D)\end{aligned} \tag{40}$$

So we can further get (41).

$$\begin{aligned}\Delta Q_i(t) &\le \frac{1}{2}E[T_{avg}(t)^2 + D^2 + 2Q_i(t)(T_{avg}(t) - D)|Q_i(t)] \\ &\le E[\frac{T_{avg}(t)^2 + D^2}{2}|Q_i(t)] \\ &\quad + Q_i(t)E[-D + T_{avg}(t)|Q_i(t)] \\ &\le \frac{1}{2}[\max E[T_{avg}(t)^2] + D^2] \\ &\quad + Q_i(t)E[-D + T_{avg}(t)|Q_i(t)] \\ &= H + Q_i(t)E[-D + T_{avg}(t)|Q_i(t)]\end{aligned} \tag{41}$$

Thus (36) is proved.

B. MINIMIZING DRIFT-PLUS-PENALTY PERFORMANCE

While minimizing the bound on $\Delta Q_i(t)$ every time slot would stabilize the system, the resulting cost might be unnecessarily large. Instead, we minimize a bound on the following drift-plus-penalty expression (42), where the coefficient V can be set to any nonnegative

value that represents the weight on minimizing energy consumption.

$$\Delta Q_i(t) + VE[E_{avg}(t)|Q_i(t)] \tag{42}$$

Next, we use (42) instead of $\Delta Q_i(t)$ in (36) and get a bound on the drift-plus-penalty as shown in (43).

$$\begin{aligned}&\Delta Q_i(t) + VE[E_{avg}(t)|Q_i(t)]\\ &\quad\le H + E[Q_i(t)T_{avg}(t)|Q_i(t)] - DQ_i(t)\\ &\qquad + VE[E_{avg}(t)|Q_i(t)]\end{aligned} \tag{43}$$

At each time slot, we minimize the following expression (44) instead of (34).

$$\min_{\forall j\in J} E[Q_i(t)T_{avg}(t)|Q_i(t)] + VE[E_{avg}(t)|Q_i(t)] \tag{44}$$

Finally, we define the One-time Slot Optimization Problem (OSOP) as in (45)–(50), shown at the bottom of the next page.

The optimization goal (34) can be achieved by minimizing the above one-time slot optimization problem at each step. We can achieve a quantified near optimal solution by minimizing (45) at each time slot, which is proved at the section below.

C. OPTIMALITY ANALYSIS

As in [30], let †denote any S-only offloading policy, and $T^{\dagger}_{avg}(t)$ and $E^{\dagger}_{avg}(t)$ denote the average delay and average energy consumption based on the policy †at time slot t. Then, expression (43) can be written as below

$$\begin{aligned}&\Delta Q_i(t) + VE[E_{avg}(t)|Q_i(t)]\\ &\quad\le H + E[Q_i(t)T_{avg}(t)|Q_i(t)] - D|Q_i(t)\\ &\qquad + VE[E_{avg}(t)|Q_i(t)]\\ &\quad\le H + Q_i(t)E[T^{\dagger}_{avg}(t) - D|Q_i(t)] + VE[E^{\dagger}_{avg}(t)|Q_i(t)]\\ &\quad= H + Q_i(t)E[T^{\dagger}_{avg}(t) - D] + VE[E^{\dagger}_{avg}(t)]\end{aligned} \tag{51}$$

If $\delta > 0$, there exists an S-only policy achieving $E[T^{\dagger}_{avg}(t)] \le D - \delta$ [31], and among all feasible S-only policies, $E^{*}_{avg}(\delta)$ is the optimal average energy consumption. Then, (51) can be transformed into (52).

$$\Delta Q_i(t) + VE[E_{avg}(t)|Q_i(t)] \le H - Q_i(t)\delta + VE^{*}_{avg}(\delta) \tag{52}$$

Taking expectations of (52), we get the following (53).

$$\begin{aligned}&\sum_{t=0}^{T-1} E[\Delta(Q_i(t))] + V\sum_{t=0}^{T-1} E[E[E_{avg}(t)|Q_i(t)]]\\ &\qquad\le TH - \sum_{t=0}^{T-1} E[Q_i(t)]\delta + VTE^{*}_{avg}(\delta)\end{aligned} \tag{53}$$

Using the law of iterated expectations as before, we sum (53) for some positive integer $t \in [0, T-1]$ and get the following (54).

$$\begin{aligned}&E[L(Q_i(t)) - E[L(Q_i(0)] + V\sum_{t=0}^{T-1} E[E_{avg}(t)]\\ &\qquad\le TH - \sum_{t=0}^{T-1} E[Q_i(t)]\delta + VTE^{*}_{avg}(\delta)\end{aligned} \tag{54}$$

After omitting the non-negative quantities, (54) is divided by $T\delta$ and VT respectively, and thus we get (55) and (56).

$$\begin{aligned}&\frac{1}{\mathrm{T}}\sum_{\mathrm{t=0}}^{\mathrm{T\text{-}1}} E[Q_i(t)]\\ &\qquad\le \frac{H}{\delta} + \frac{VE^{*}_{avg}(\delta) - \frac{V}{T}\sum_{t=0}^{T-1} E[E_{avg}(t)]}{\delta} + \frac{E[L(Q(0))]}{T\delta}\end{aligned} \tag{55}$$

$$\begin{aligned}&\frac{1}{\mathrm{T}}\sum_{t=0}^{T-1} E[E_{avg}(t)]\\ &\qquad\le \frac{H}{V} + E^{*}_{avg}(\delta) + \frac{E[L(Q(0))]}{VT}\end{aligned} \tag{56}$$

Taking limits of (55) and (56) as $T \to \infty$ respectively, we deduce (57) and (58) below, where E^{*}_{avg} is the optimal long-term average energy consumption achieved by any policy.

$$\lim_{T\to\infty}\sup\frac{1}{T}\sum_{T=0}^{T-1} E[Q_i(t)] \le \frac{H + V[E^{*}_{avg}(\delta) - E^{*}_{avg}]}{\delta} \tag{57}$$

$$\lim_{T\to\infty}\sup\frac{1}{T}\sum_{T=0}^{T-1} E[E_{avg}(t)] \le \frac{H}{V} + E^{*}_{avg}(\delta) \tag{58}$$

The bounds (57) and (58) indicate queue stability and demonstrate an $[O(V), O(1/V)]$ tradeoff between average delay and average energy consumption. We can use an arbitrarily large V to make M/V arbitrarily small, so that the (58) illustrates that with the increasing of parameter V, the energy consumption is closer to $E^{*}_{avg}(\delta)$. However, when V is too large, (57) illustrates that the data queue is not stable, which means the delay will exceeded the predefined system expected finish time; when V decreases, (57) illustrates that the data queue tends to be stable, but (58) illustrates that the average energy consumption will increase. Tuning the parameter V can change queue stability and delay at the same time. There will be a trade-off between the average queue delay and average energy consumption.

VI. IMPROVED DNSLA BASED OPTIMIZATION

Considering that the mathematical model of our proposed optimization problem which falls into the class of NP-hard problems [4], we develop an improved DNSLA to solve it and thus obtain a trade-off offloading decision. If the offloaded task is a delay-sensitive type, then its delay will be considered

first when making the trade-off decision; on the contrary, if the offloaded task is an energy-sensitive type, then its energy will be considered first when making the trade-off decision.

DNSLA is a dynamic niche-based evolutionary algorithm, which carries out a self-organizing learning process consisting of global learning, neighborhood learning and self-learning. In the global learning, the learning rate of an individual is dynamically adjusted based on the fitness of the individual and the average fitness of the population to which the individual belongs. If the fitness of the individual is greater than the average fitness of the population, the global learning rate is enhanced to help the best individual in the current whole ecosystem to carry out an intensive search. In the neighborhood learning, the learning rate of an individual is dynamically adjusted based on the Hamming distance between the individual and the best individual of the current population to which the individual belongs. With the dynamic learning rate, the Hamming distance between them is shortened as much as possible to assist in the best individual of the current population to carry out an intensive search. In the self-learning, the learning rate of an individual is dynamically adjusted depending on the ratio of the average fitness of the population to the fitness of the individual. If the fitness of the individual is worse than the average fitness of the population, its self-learning rate will quickly rise to a relatively high level and obtain the dual individual by the dual mapping to the current individual. With the ongoing learning, when a niche population finds a better global solution than before, the other populations will assign some of their individuals into the existing niche population to carry out an intensive search; however, if a niche population never finds a better global solution than before during its evolutionary process, all the individuals in the population will be gradually transferred to the other niche populations and finally this niche population will disappear. The ecosystem is divided into many niche populations, so both carrying out the study and search synchronously in each population and exchanging individual dynamically among the populations can accelerate the speed of the search process to gain an optimal task offloading scheme.

Algorithm 1 Initial Solutions Acquisition

Input: $W_{(i,j)}^{(t)}$
Output: $\pi^*(t), \lambda^*(t)$

```
FOR each time slot t DO
  Initialize TU_opt ← ∞
  FOR each MEC server j ∈ J DO
    Set I_(i,j)(t) ← 1, I_(i,k)(t) ← 0, ∀k ∈ J(k ≠ j)
    Let the derivate of (45) with respect to λ_i(t) is zero
    and derive feasible solution TU_min that makes
    objective minimum
    IF TU_min < TU_opt THEN
      TU_opt ← TU_min
      I*_(i,j) ← I_(i,j)(t)
      λ*_i(t) ← λ_i(t)
    END IF
  END FOR
  The optimal solution at time slot t is I*_(i,j) and λ*_i(t)
END FOR
RETURN π*(t) and λ*(t).
```

A. INITIAL SOLUTIONS ACQUISITION

We first solve the relaxation problem of OSOP via Algorithm 1, where has an additional constraint (59). The solutions will be regarded as the initial solutions of our improved DNSLA.

$$\sum_{j\in J} I_{(i,j)}^{m}(t) = 1, \quad \forall i \tag{59}$$

B. RELATED DEFINITION

Each solution is viewed as an individual in the niche population. Let $E = \{P_1, P_2, \cdots, P_k, \cdots, P_n\}$ be an ecosystem consisting of n niche populations, N_k represents the number

$$TU = \min_{\lambda_i(t),\pi(t)} V\left(\frac{\sum_{i=1}^{M}\sum_{j=1}^{\lambda_i(t)} I_{(i,j)}^{l}(t)(\alpha(f_{i,l})^{\gamma}\frac{C_{(i,j)}^{(t)}}{f_{i,l}}) + I_{(i,j)}^{m}(t)(\frac{P_i^{up}D_{(i,j)}^{(t)}}{R_i})}{\sum_{i=1}^{M}\lambda_i(t)}\right) + Q_i(t)\left(\frac{\sum_{i=1}^{M}\sum_{j=1}^{\lambda_i(t)} I_{(i,j)}^{l}(t)(\frac{C_{(i,j)}^{(t)}}{f_{i,l}} + \sum_{j=1}^{Q_i(t)}\frac{C_{(i,j)}^{(t)}}{f_{i,l}}) + I_{(i,j)}^{m}(t)(\frac{D_{(i,j)}^{(t)}}{R_{i,j}} + \frac{C_{(i,j)}^{(t)}}{f_{i,m}})}{\sum_{i=1}^{M}\lambda_i(t)}\right) \tag{45}$$

$$S.t.\ I_{(i,j)}^{l}(t), I_{(i,j)}^{m}(t) \in \{0,1\}, \quad \forall i,j \tag{46}$$

$$I_{(i,j)}^{l}(t) + I_{(i,j)}^{m}(t) = 1 \tag{47}$$

$$N_i^{local}(t) + N_i^{mec}(t) = \lambda_i(t) \tag{48}$$

$$N_i^{local}(t) = \sum\nolimits_{i\in X_i^{local}(t)} I_{(i,j)}^{l}(t) \tag{49}$$

$$N_i^{mec}(t) = \sum\nolimits_{i\in X_i^{mec}(t)} I_{(i,j)}^{m}(t) \tag{50}$$

of individuals in population P_k, P_k^i represents the ith individual in P_k. The fitness f_k^i of individual P_k^i is calculated according to (14). P_k^{best} denotes the individual with the best fitness in P_k, $\overline{f_k}$ denotes the average fitness of P_k in the current generation, $\overline{f_E}$ denotes the average fitness of the ecosystem in the current generation, P^{best} denotes the individual with the best fitness in the current ecosystem, $I_{\max}$ denotes the largest number of iterations.

The self-organizing learning consists of three kinds of learning strategies: global learning, neighborhood learning and self-learning, which are described below respectively.

C. GLOBAL LEARNING

All the individuals in each population start to learn from the individual with the best fitness in the whole ecosystem in this phase, that is, P^{best}. The global learning rate of P_k^i is defined in (60), where *Grate* is an initial value of the global learning rate.

$$Grate_k^i = Grate + \left(f_k^i / \overline{f_k}\right) - 1 \tag{60}$$

D. NEIGHBORHOOD LEARNING

All the individuals in a population P_k start to learn from the individual with the best fitness in the population in this phase, that is, P_k^{best}. The neighborhood learning rate of P_k^i is defined in (61), where *Nrate* is an initial value of the neighborhood learning rate, $HD_{k,h}$ denotes the Hamming distance between the individual and P_k^{best}, *Length* denotes the encoding length of identifying an individual in the ecosystem.

$$Nrate_k^i = Nrate - \left(HD_{k,h}/Length\right) + 1 \tag{61}$$

E. SELF-LEARNING

An individual starts to learn from itself based on dual mapping in this phase. The self-learning rate of P_k^i is defined in (62), where *Srate* is an initial value of self-learning rate.

$$Srate_k^i = Srate \times \left(\overline{f_k}/f_k^i\right) \tag{62}$$

Our improved DNSLA is described in the following Algorithm 2. Here, lines 1–2 are the initialization phase; lines 5–17 are to calculate the fitness of population; lines 18–26 are the self-organizing learning process; lines 29–30 are the exchange process of the best individuals among the populations when the average fitness and the best individual of ecosystem do not change in the last generation.

F. COMPUTATION COMPLEXITY ANALYSIS

Our improved DNSLA initializes the relevant parameters in $O(N_1 + N_2 + \cdots + N_k + \cdots N_n)$. At each iteration, it takes $O(N_1 + N_2 + \cdots + N_k + \cdots N_n) + O(n)$ to calculate the fitness: f_k^i, $\overline{f_k}$ and $\overline{f_E}$; updating P_k^{best} and P_{best} needs $O(n)$ time; for each P_k, P_k^i has the worst case (that is, global learning, neighborhood learning and self-learning are all executed) time complexity of $O(N_k^2)$, and thus for all the populations the worst case time complexity is $O(N_1^2 + N_2^2 + \cdots + N_k^2 + \cdots N_n^2)$; calculating new $\overline{f_E}$ and P_{best} requires $O(N_1 + N_2 + \cdots + N_k + \cdots N_n) = O(n)$. Iterations repeat $I_{\max}$ times, so the total time complexity is

Algorithm 2 Improved DNSLA

Input: $\pi^*(t)$, $\lambda^*(t)$
Output: $\pi^*_{opt}(t)$, $\lambda^*_{opt}(t)$

1: Initialize the related parameters: $n \leftarrow n_0$, $N_k \leftarrow n_k$, $Grate \leftarrow 0.5$, $Nrate \leftarrow 0.5$, $Srate \leftarrow 0.8$, $F_k^i \leftarrow 0$, $iter \leftarrow 1$;
   //n_0 and n_k denote the initial values of n and N_k respectively.
   // To keep the population diversity and guarantee a reasonable evolution rate,
   // we set $Grate \leftarrow 0.5$, $Nrate \leftarrow 0.5$ and $Srate \leftarrow 0.8$ according to [32].
2: Get $P_k^i \leftarrow \pi_k^i$ by Algorithm 1;
3: **For** $iter \leftarrow 1$ to $I_{\max}$ **do**
4: **For** $k \leftarrow 1$ to n **do**
5: Calculate $f_k^i \leftarrow TU_k^i$ of individuals in P_k.
6: $\overline{f_k} \leftarrow \left(\sum f_k^i\right)/N_k$; //Calculate average fitness $\overline{f_k}$ of P_k.
7: **If** the individual with the best fitness in P_k is unique **then**
8: Set it as P_k^{best};
9: **Else** set the individuals with the best fitness in P_k as $P_k^{best\_1}, P_k^{best\_2}, \ldots, P_k^{best\_m_k}$;
   //Assume the number of the individuals with the best fitness in P_k is m_k.
10: Store all the individuals with the best fitness inP_k to a set of elitist solutions:
   $ES_k^{best} = \left\{P_k^{best\_1}, P_k^{best\_2}, \ldots, P_k^{best\_m_k}\right\}$;
11: **End-If**
12: **If** the individual with the best fitness in the ecosystem is unique **then**
13: Set it as P^{best};
14: **Else** set the individuals with the best fitness in the ecosystem as $P^{best\_1}, P^{best\_2}, \ldots, P^{best\_m}$;
   //Assume the number of the individuals with the best fitness in P_k is m.
15: Store all the individuals with the best fitness in the ecosystem to a set of elitist solutions:

$$ES^{best} = \left\{P^{best\_1}, P^{best\_2}, \ldots, P^{best\_m}\right\};$$

16: **End-If**
17: $\overline{f_E} \leftarrow \left(\sum \overline{f_k}\right)/n$; //Calculate the average fitness of the ecosystem.
18: $F_k^i \leftarrow Grate_k^i$; //Carry out the global learning and F_k^i is a temporary variable to store learning rate.
19: Update f_k^i;
20: **If** the fitness of P_k^i is not improved **then**
21: $F_k^i \leftarrow Nrate_k^i$; //Carry out the neighborhood learning.
22: Update f_k^i;
23: **If** the fitness of P_k^i is still not improved **then**
24: $F_k^i \leftarrow Srate_k^i$; //Carry out the self-learning.
25: Update f_k^i;
26: **End-If**
27: **End-For**
28: Update $\overline{f_k}$, P_k^{best} (or ES_k^{best}), $\overline{f_E}$ and P^{best} (or ES^{best});
29: **If** $\overline{f_E}$ and P^{best} (or ES^{best}) do not change before and after this round of iteration **then**
30: $P_k^{best} \leftrightarrow P_l^{best}$, $l \in \{1, 2, \cdots, n\}$, $l \neq k$; //Exchange the best individuals among populations.
31: **End-For**
32: Obtain P^{best};
33: **Return** $\pi^*_{opt}(t)$ and $\lambda^*_{opt}(t)$.

TABLE 2. Parameters settings in our simulation.

Parameter	default value
M	10
$f_{i,l}$	1-1.5GHz
$f_{i,m}$	10GHz
W	1MHz
N_0	-174dBm/Hz
p_i^{up}	0.1W
$D_{(i,j)}^{(t)}$	[5-20]M
l	15min

$O\left(n+I_{\max}\times\left(n+n+\left(N_1^2+N_2^2+\cdots+N_k^2+\cdots N_n^2\right)+n\right)\right)=O\left(n^2\right)$.

VII. SIMULATION

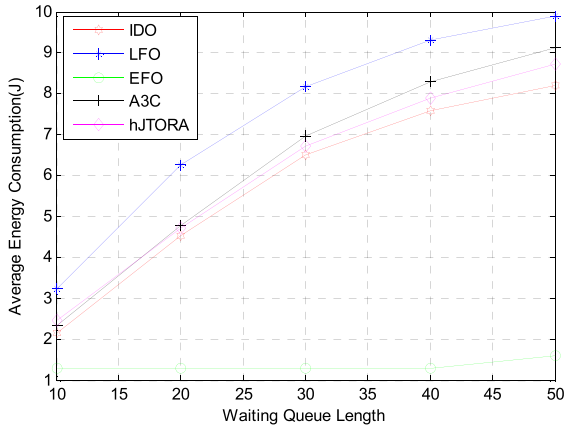
A. ENVIRONMENT SETTING

The scenario of simulation experiment includes 5 mobile edge servers and 30 terminal devices, which are randomly distributed within the range of 500m×500m. The unit time slot length is 5 minutes, and the maximum queue length is 50. We assume that 1) the generated tasks in our experiments follow the 18 kinds of service classes according to the definition from ITU-T [33]; 2) the task arrival rate $\lambda_i(t)$ of the terminal device in simulation obeys Poisson distribution; 3) the data size of the tasks in each time slot follows uniform distribution; and 4) the number of CPU cycles required to complete the tasks follows an exponential distribution. In addition, CPU frequency of the edge node is set to 20GHz, and that of the terminal node is set to 1GHz-1.5GHz. Simulation parameters are shown in Table 2. Simulation results are averaged over 100 independent processes.

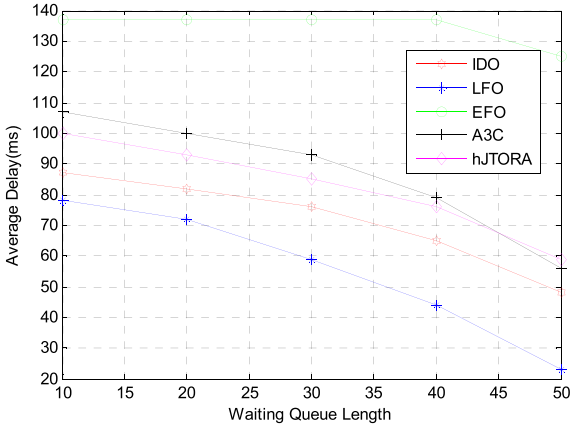
B. BENCHMARK SCHEMES

In order to evaluate the performance of our proposed algorithm, we take the "edge first" algorithm, the "local first" algorithm, the "asynchronous advantage actor-critic (A3C)" algorithm in [34] and the "heuristic joint task offloading scheduling and resource allocation (hJTORA)" algorithm in [11] as our benchmark.

- IDO scheme, namely our improved DNSLA based offloading scheme.
- LFO scheme, namely "local first" offloading scheme. The generating tasks are first processed locally until the waiting queue length preset in the local nodes is reached, and the remaining tasks will be sent to process in the edge nodes.
- EFO scheme, namely "edge first" offloading scheme. The task distribution strategy tends to first send the generating tasks to process in the edge nodes.



(a) Average energy consumption vs. waiting queue length



(b) Average delay vs. waiting queue length

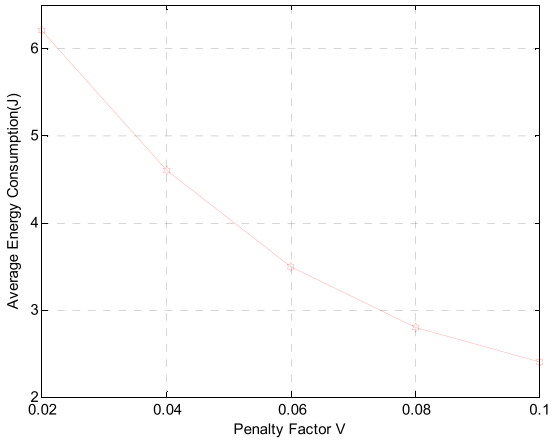
FIGURE 3. The impact of waiting queue length on performance.

- A3C scheme, namely an online task offloading algorithm based on a state-of-the-art deep reinforcement learning technique.
- hJTORA scheme, namely a novel heuristic algorithm to tackle the task offloading problem that achieves a suboptimal solution in polynomial time.

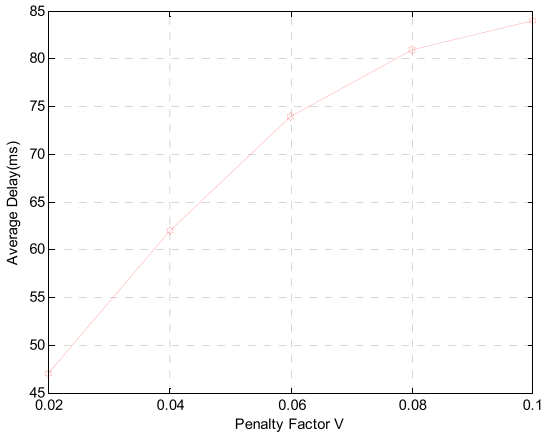
C. PERFORMANCE EVALUATION

1) WAITING QUEUE LENGTH IMPACT

The comparison results on average energy consumption and average delay with different limitation of the backlog in the waiting queue shown in Fig. 3 indicate that for the LFO scheme, the average energy consumption increases and the average delay decreases when the limitation of the allowed maximum length in the waiting queue increases. This is because more applications are allocated to the local nodes for their processing. A3C scheme and our proposed IDO scheme have the same energy consumption trend and average delay trend as the LFO scheme. A3C scheme costs more average energy and average delay than our proposed IDO scheme, because A3C scheme does not find a tradeoff online solution compared to IDO scheme. The hJTORA scheme performs closely to average energy consumption and average delay

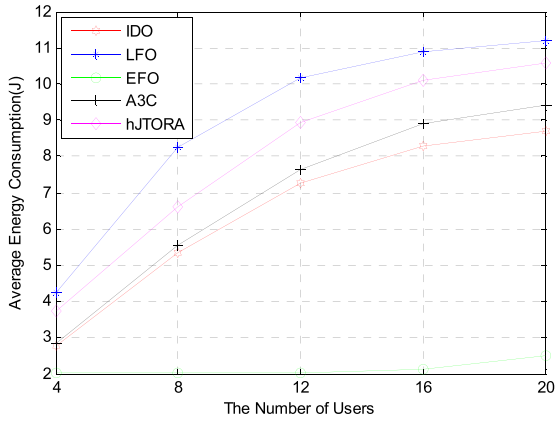


(a) Average energy consumption vs. penalty factor

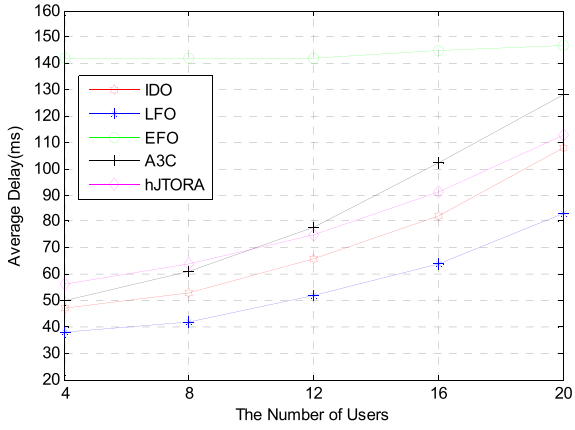


(b) Average delay vs. penalty factor

FIGURE 4. The impact of penalty factor on performance.



(a) Average energy consumption vs. the number of users



(b) Average delay vs. the number of users

FIGURE 5. The impact of user number on performance.

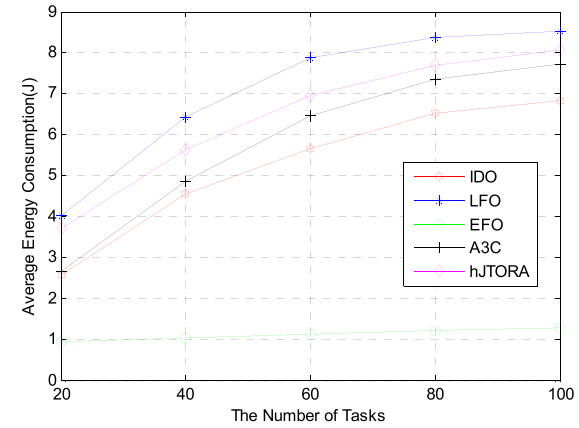
of the proposed IDO scheme while both schemes significantly outperform the other baselines. However, IDO scheme scheme still outperforms the hJTORA scheme for both average energy and average delay. We further observe that the average energy consumption and average delay of the EFO scheme remain nearly constant because of the task distribution strategy in the EFO scheme being independent of waiting queue length in the local nodes. As for the tradeoff between average energy consumption and average delay, from Fig. 3. there is evidence to suggest that our proposed IDO scheme has the best profile among five schemes.

2) PENALTY FACTOR IMPACT

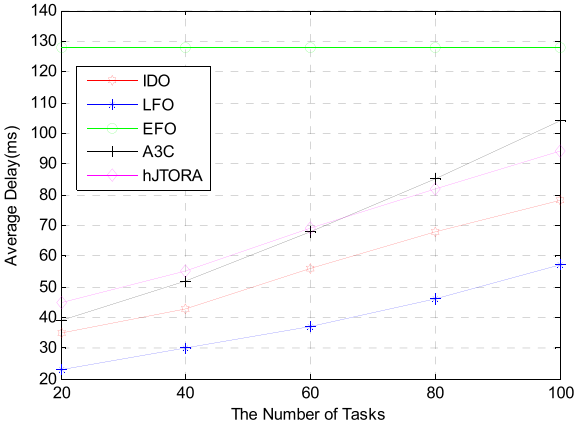
Following, we investigate the impacts of the penalty factor V on system performance in our proposed IDO scheme, we observed how the average energy consumption and average delay change accordingly when we set V with different values. In our above definition, V represents the importance of the average energy consumption will impact the system performance. As depicted in Fig. 4, when the value of V is a relatively small one, we observed that the average delay remains small while the average energy consumption remains large. This is due to the penalty of energy consumption is low in the system, so that the tasks are processed locally in order to get a better delay. However, if we set V to a relatively large value, the significant decrement of the average energy consumption and increment of the average delay can be observed. Furthermore, when the average delay drops, the average energy consumption grows rapidly.

3) USER NUMBER IMPACT

We now evaluate average energy consumption and average delay of the MEC system against different number of users wishing to offload their computing tasks, as shown in Fig. 5. In particular, we vary the number of users from 4 to 20 and perform the comparison among five schemes. From Fig. 5, we also observe that for the LFO scheme, IDO scheme and A3C scheme, the average delay increased sharply when the number of users is relatively large. This is because when there are more users competing for the limited resources, the chance that a user can benefit from offloading its task is lower than before. Furthermore, offloading more tasks to MEC servers also results in the rate of growth in the average energy consumption to decline when the number of users relatively large. For the hJTORA scheme, average energy



(a) Average energy consumption vs. the number of tasks



(b) Average delay vs. the number of tasks

FIGURE 6. The impact of task number on performance.

consumption and average delay vary almost linearly when there is a small number of users, but the former varies slowly and the latter varies sharply when there is a large number of users. For the EFO scheme, Fig. 5 shows the average energy consumption and the average delay vary very slowly when the number of users varies. The former remains far smaller values than other schemes and the latter always stays at far larger values than other schemes.

4) TASK NUMBER IMPACT

Figure 6 depicts the average energy consumption and average delay with varying number of tasks. It is observed that for the LFO scheme, IDO scheme, A3C scheme and hJTORA scheme, once the number of tasks is beyond local thresholds, the average delay begins to increase rapidly (but lower than the scenario in the varying number of users). The reason is that the increasing number of tasks require more computation resources, which leads to a part of tasks having to be offloaded to the MEC servers and thus resulted in longer task execution delay. In addition, we can also observe that the average energy consumption varies similarly to the scenario of the changing users. Unexpectedly, for the EFO scheme, the varying number of tasks has almost no influence on the average delay and only cause a slight increase in the average energy consumption. Compared to the other four schemes, our proposed IDO scheme performs excellently at relatively lower average energy consumption than LFO, A3C and hJTORA scheme, and significantly lower average delay than the EFO scheme. The gap among them illustrates the efficiency of optimal offloading achieved by jointly considering Lyapunov theory and improved DNSLA algorithm.

VIII. CONCLUSION

The rapid increasing Internet traffic can be processed as quickly as possible by offloading tasks to edge nodes, so the two-layer MEC system can provide users with efficient experience. In this paper, we propose a reliable online task offloading scheme to achieve the tradeoff between data processing delay and energy consumption and provide efficient data service. Simulation results indicate that our proposed scheme achieves the expected goal. In the future, we will further study the task offloading decision in three-layer MEC system. In particular, we shall construct a safe and reliable cloud-fog collaboration mode that takes task priority into consideration and is in line with the requirements of the actual various applications. Furthermore, we shall consider to extend our future research by generalizing and verifying our approaches from the current MEC testbed to various prototype systems and some classic real systems in our subsequent work.

REFERENCES

[1] Cisco. (2020). *Cisco Annual Internet Report (2018–2023) White Paper*. Accessed: Mar. 12, 2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/clooateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing-a key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper 11, 2015, pp. 1–16.

[3] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[4] B. Chen and G. Quan, "NP-hard problems of learning from examples," in *Proc. 5th Int. Conf. Fuzzy Syst. Knowl. Discovery*, Oct. 2008, pp. 182–186.

[5] J. Gao and J. Wang, "Multi-edge collaborative computing unloading scheme based on genetic algorithm," *Comput. Sci.*, vol. 48, no. 1, pp. 72–80, 2021.

[6] Z. Li and X. Zhang, "Resource allocation and offloading decision of edge computing for reducing core network congestion," *Comput. Sci.*, vol. 48, no. 2, pp. 281–288, 2021.

[7] K. Peng, J. Nie, N. Kumar, C. Cai, J. Kang, Z. Xiong, and Y. Zhang, "Joint optimization of service chain caching and task offloading in mobile edge computing," *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107142.

[8] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4642–4655, Oct. 2018.

[9] M.-H. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Wireless Commun.*, vol. 17, no. 10, pp. 6790–6805, Oct. 2018.

[10] Y. Maohi and T. P. Zhou Liu, "Multi-user task offloading based on delayed acceptance," *Comput. Sci.*, vol. 48, no. 1, pp. 49–57, 2021.

[11] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

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

[13] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "Noma-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12244–12258, Dec. 2018.
[14] M. Zeng and V. Fodor, "Energy minimization for delay constrained mobile edge computing with orthogonal and non-orthogonal multiple access," *Ad Hoc Netw.*, vol. 98, Mar. 2020, Art. no. 102060.
[15] S. Xia, Z. Yao, Y. Xian, and Y. Li, "A distributed heterogeneous task offloading methodology for mobile edge computing," *J. Electron. Inf. Technol.*, vol. 42, no. 12, pp. 2891–2898, 2020.
[16] R. Wang, H. Wu, Y. Cui, D. WU, and H. Zhang, "Edge offloading strategy for the multi-base station game in ultra-dense networks," *J. Xidian Univ.*, vol. 48, no. 4, pp. 1–9, 2021.
[17] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: A behavioral perspective," *IEEE Netw.*, vol. 32, no. 1, pp. 48–53, Jan. 2018.
[18] X. Hu, K.-K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2375–2388, Apr. 2018.
[19] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Trans. Mobile Comput.*, vol. 19, no. 1, pp. 29–43, Jan. 2020.
[20] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "A dynamic tradeoff data processing framework for delay-sensitive applications in cloud of things systems," *J. Parallel Distrib. Comput.*, vol. 112, pp. 53–66, Feb. 2018.
[21] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 726–738, Sep. 2019.
[22] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2443–2459, Apr. 2020.
[23] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, Mar. 2019.
[24] F. Liu, Z. Zhou, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2648–2658, Oct. 2014.
[25] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[26] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.
[27] M. Dai, Z. Liu, S. Guo, S. Shao, and X. Qiu, "A computation offloading and resource allocation mechanism based on minimizing devices energy consumption and system delay," *J. Electron. Inf. Technol.*, vol. 41, no. 11, pp. 2684–2690, 2019.
[28] M. J. Neely, "Queue stability and probability 1 convergence via Lyapunov optimization," 2010, *arXiv:1008.3519*.
[29] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "Cost-effective processing for delay-sensitive applications in cloud of things systems," in *Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA)*, Oct. 2016, pp. 162–169.
[30] M. J. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Nov. 2003.
[31] Y. Niu, B. Luo, F. Liu, J. Liu, and B. Li, "When hybrid cloud meets flash crowd: Towards cost-effective service provisioning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1044–1052.
[32] X. Wang, J. Zhang, M. Huang, and S. Yang, "A green intelligent routing algorithm supporting flexible QoS for many-to-many multicast," *Comput. Netw.*, vol. 126, pp. 229–245, Oct. 2017.
[33] *Network Performance Objectives for IP-based Services*, Standard ITU-T Y.1541, 2011.
[34] B. Gu, M. Alazab, Z. Lin, X. Zhang, and J. Huang, "AI-enabled task offloading for improving quality of computational experience in ultra dense networks," *ACM Trans. Internet Technol.*, vol. 22, no. 3, pp. 1–17, Aug. 2022.

RUIXIA LI received the M.S. degree in computer science from the Taiyuan University of Technology, China, in 2007. She is currently pursuing the Ph.D. degree with the Asia Pacific University of Technology and Innovation, Kuala Lumpur, Malaysia. She joined West Anhui University, Anhui, China, in 2007, where she is also an Associate Professor. Her research interests include the Internet of Things, edge computing, and network resource management.

CHIA SIEN LIM received the B.S. degree (Hons.) in mathematics and the M.S. degree in mathematics from Oklahoma State University, Stillwater, OK, USA, in 1994 and 1997, respectively, and the Ph.D. degree in mathematics from Michigan State University, East Lansing, MI, USA, in 2002. He is currently heading the Graduate School of Technology, Asia Pacific University of Technology and Innovation. His research interests include linear algebra and its application to machine learning and data science.

MUHAMMAD EHSAN RANA received the M.S. degree in computer science and mathematics, and the Ph.D. degree in software engineering from Universiti Putra Malaysia. He possesses more than 20 years of experience in teaching, research, and academic management. He joined with the Asia Pacific University of Technology & Innovation (APU), Malaysia, in 2008. He is currently an Associate Professor and holding the Manager TNE Program Role. He has authored and coauthored many journal articles and conference publications in the areas specified. He is also a reviewer of several indexed journals and has participated in the organization of IEEE and other international conferences. His research interests include software architecture, cloud computing, blockchain, and the IoT.

XIANCUN ZHOU received the B.S. degree in electrical engineering from Anhui University, Hefei, China, in 1997, and the M.S. degree in computer science from the Heifei University of Technology, Heifei, in 2004. She was a Visiting Scholar with the University of Science and Technology of China, China, from 2010 to 2011. She is currently a Professor with West Anhui University, Anhui, China. Her current research interests include the Internet of Things, computer networking, and mobile computing.