Knowledge-Defined Edge Computing Networks Assisted Long-term Optimization of Computation Offloading and Resource Allocation Strategy

Knowledge-Defined Edge Computing Networks Assisted Long-term Optimization of Computation Offloading and Resource Allocation Strategy

Kaiqi Yang, Xingwei Wang*, Qiang He, Liang Zhao, *Member, IEEE*, Yufei Liu, Daniele Tarchi, *Senior Member, IEEE*

***Abstract*—With the proliferation of devices connected to the Internet of Things (IoT), the complexity of network management has increased. To intelligently manage large-scale networks, we propose a Knowledge-Defined Edge Computing Networks (KDECN) architecture. Edge Nodes (ENs) deployed in the KDECN architecture are responsible for collecting and preprocessing the relevant information uploaded by User Devices (UDs), and provide computation resources for UDs. Futhermore, since multiple UDs share system computation resources, one computing decision will affect the subsequent decision-making of other UDs. Thus, accurately predicting the demands for UD task requests is a key challenge to maximize long-term execution utility. To this end, we deploy the LSTM-based Task Request Demand Prediction (TRDP) method on the management plane of KDECN architecture to predict the task request quantity of UDs in each future time slot. In order to maximize long-term execution utility of the system, we propose a Deep Reinforcement Learning (DRL)-based Long-term Computation Offloading and computation Resource Allocation (L-CORA) algorithm. Specifically, the proposed L-CORA algorithm makes computing decisions based on the prediction of the offloading task quantity and the personalized demands of UDs to ensure the long-term quality of computing service. Extensive experiments with Shanghai real-world datasets to prove that the KDECN-based L-CORA algorithm effectively improves the average utility of the system.**

***Index Terms*—Computation offloading, knowledge-defined networking, mobile edge computing, deep reinforcement learning.**

I. Introduction

In recent years, with the continuous development of the Internet of Things (IoT) [1]–[3], a variety of applications have emerged to provide users with services in industrial, military, entertainment and other fields [4]–[7]. Network applications bring convenience to people's life, which subsequently places a heavy burden on User Devices (UDs). These applications are usually time-sensitive and computation-complex, and are limited by maximum delay tolerance. Therefore, the limited computation resources of UDs often struggle to meet the computing requirements of the applications, which leads to the fact that the Quality of Services (QoS) cannot be guaranteed [8]. Computation offloading is one of the technologies to address the issues stated above.

Computation offloading is the process of sending computation tasks to other devices with powerful computing power via wireless connections to perform the tasks instead of the requesting devices. Mobile Edge Computing (MEC) takes advantage of its closer proximity to UDs, enabling faster service response for computation offloading [9]–[12]. Moreover, with the continuous development of wireless networks, the prospect of computation offloading is quite optimistic. Nevertheless, it is one-sided to solely consider a single UD or a specific region for the computation offloading and resource allocation decision-making [13]–[15]. Thus, the decision-making should consider the impact of a UD decision on the status of other UDs, which needs global information to support the decision-making. In addition, there are conflicts between the goals of different stakeholders (i.e., service providers, users, etc.) in the network, a network architecture is needed to support the mediation of these conflicts [16].

Knowledge-Defined Networking (KDN) [17] is a new networking paradigm based on Software Defined Network (SDN) [18], Knowledge Plane (KP) [19] and network analysis [20]. The KP provides services and recommendations to other elements of the system by creating, coordinating, and maintaining high-level views, and utilizing machine learning (ML) techniques. It is continuously trained through the global view and network analysis provided by the control and management plane, and then acquires knowledge stored in the knowledge base. After that, according to the learned knowledge, a reasonable decision or recommendation scheme is formulated on account of the requirements of the system [21]. Furthermore, the KP iteratively optimizes its decision models based on system feedback, so that it can reconcile conflicts caused by differences in UDs individual demands and operator objectives, and then optimize the QoS of the network [22]–[24]. KDN divides the system into four planes: data plane, control plane, management plane and KP. The layered architecture of KDN decouples each function and completes their work in a distributed manner, making the system control and management more efficient. However, the powerful function of the KDN relies on the availability of a significant amount of UD data, and all UDs upload their own information to the control and management plane, which may cause severe interference, information redundancy and waste of network resources. Therefore, a more efficient computation

K. Yang, X. Wang and Y. Liu are with Colledge of Computer Science and Engineering, Northeastern University, Shenyang, China. Xingwei Wang is the corresponding author.

Q. He is with College of Medicine and Biological Information Engineering, Northeastern University, Shenyang, China (heqiangcai@gmail.cn).

L. Zhao is with the School of Computer Science, Shenyang Aerospace University, Shenyang, China (lzhao@sau.edu.cn).

D. Tarchi is with the College of Electronic and Information Engineering, University of Bologna, Bologna, Italy (daniele.tarchi@unibo.it).

offloading and resource allocation optimization architecture is needed to solve these issues.

In summary, there are some key challenges in optimizing computation offloading and resource allocation strategies. One major challenge lies in the dynamic and complex nature of large-scale networks, rendering traditional network management systems inadequate for adapting to the increasingly complexity of the network. Secondly, the centralized network architecture relies on real-time collection of UDs information for dynamic network management. However, the real-time uploading of relevant information by UDs often results in higher energy consumption and severe signal interference. Moreover, the information uploaded by UDs may have the problems of information redundancy and unreliability, resulting in low efficiency of information uploading and reduced accuracy of subsequent analysis. Another challenge is that the system computation resources are shared among all UDs, any computing decision made by one UD will have a certain impact on the subsequent UDs' decisions. Therefore, it is a key challenge to jointly optimize the computation offloading and resource allocation decisions of large-scale UDs to maximize long-term utility of the system. In short, an intelligent network architecture that optimizes long-term execution utility is required to solve the issues caused by the above challenges.

Consequently, inspired by the KDN architecture, we propose a new network architecture Knowledge-Defined Edge Computing Networks (KDECN), which can collect information, self-learn and make decisions or recommendations for large-scale network systems. The Edge Nodes (ENs) deployed in KDECN upload the collected and pre-processed information to the control and management plane to obtain a global view and analyze the UD requirements for KP training and learning. Moreover, KP needs the deployment of powerful ML algorithm to optimize long-term computing decisions for UDs in large-scale networks. In this paper, we propose a Long Short-Term Memory (LSTM)-based Task Request Demand Prediction (TRDP) method deployed on the management plane, consider the effects of spatial-temporal and continuity factors on task requests to predict the task request quantity of UDs in each future time slot. Afterwards, in order to achieve the purpose of maximizing long-term average execution utility of the system, we deploy the Long-term Computation Offloading and computation Resource Allocation (L-CORA) algorithm on KP to optimize the long-term computing decision.

Specifically, the advantages of building the KDECN architecture are as follows: Firstly, the deployment of ENs on the data plane effectively alleviates the pressure of UDs transmitting data, removes redundant and untrusted information, achieves system data interconnection, and improves the efficiency and accuracy data collection; Secondly, the management plane analyzes and predicts the long-term task request demands of UDs, which can assist KP in optimizing long-term computing decisions in large-scale networks; Thirdly, the KP optimizes computing decisions from the perspective of operators based on UD demand-related data and predictions provided by the control and management plane, which can better reconcile goal conflicts between UDs and operators.

Specially, our contributions can be summarized as follows:

- *We propose a new network architecture (KDECN).* The KDECN architecture considers deploying ENs on the data plane to collect UD data, provide computation resources and preprocess the data, thereby reducing data redundancy and improving data validity. As a result, the accuracy of the system data has been improved, and the energy consumption and interference of the UD data transmission have been reduced.
- *LSTM-based TRDP method is proposed and deployed in the management plane.* The TRDP method is to predict the future task request quantity with spatial-temporal and continuous characteristics, which is necessary to support the long-term computing decision-making of the KP.
- *The DRL-based L-CORA algorithm is proposed.* The future demand of UDs is innovatively considered as an influencing factor of the current decision, to maximize the average execution utility of global UDs over a long period of time and consider the influence between decisions. In addition, the personalized needs of UDs are also taken into account in long-term computing decision-making to better mediate the conflict between different stakeholders in the system.
- *Experimental Verification.* We validate our proposed KDECN architecture and the L-CORA algorithm by using the realworld datasets of base station telecom in Shanghai, China. Extensive simulations also verify that the long-term execution utility in large-scale networks has been significantly improved.

The remainder of this article is organized as follows. In Section II, we present the related work. Next, in Section III, we formulate the system model and the optimization problem. Then, Section IV introduces our proposed L-CORA algorithm in detail. The evaluation results are shown in Section V. Finally, the conclusion is drawn in Section VI.

II. Related Work

Future 6G wireless communication network will have intelligent and automatic features, which requires a network architecture with intelligent analysis and learning ability to replace network managers for network self-management and self-repair [32]. D. Clark et al. put forward the concept of KP [19], which has attracted wide attention. KP is a new network architecture based on ML and cognition technology, which brings great advantages to network management and operation. However, ML deployed on the KP requires a large amount of data to support learning and training, and it is difficult to provide management and control for the complex distributed network with only a partial view. Therefore, Albert et al. combined SDN, network telemetry and KP to propose a new paradigm of KDN [17]. Alejandra et al. proposed an intelligent system for effectively identifying heavy-hitters based on KDN. The system allows the use of ML integrated behavior models to detect traffic patterns in the network, and reduces the incidence of network congestion based on traffic thresholds [33]. In addition, SDN can provide global view and network programmability function [34], which provides the foundation for KDN paradigm. The KP adopts ML technology to convert

TABLE I
COMPARISON WITH EXISTING WORK

Reference	Computing resource	Optimization objective	System architecture	Predictive offloading	Proposed solution
[25]	Fog computing	power consumption	Multitiered fog computing	✔	Lyapunov optimization
[26]	Edge computing	System latency	Edge cloud-empowered self-dirving	✔	DDPG
[27]	Edge cloud	Long-term cost	Trusted cooperative offloading	✗	Lyapunov optimization
[28]	Vehicular edge computing	Utility of vehicles	Vehicular edge computing	✗	Double deep Q-network
[29]	Mobile edge computing	Average long-term cost	MEC system	✗	Temporal attentional deterministic policy gradient
[30]	Mobile edge computing	Energy consumption and transmission delay	heterogeneous vehicular network	✗	DDPG
[31]	Edge computing	Computing utility	Edge computing	✗	Game theory and actor-critic network
Our paper	Edge computing	Long-term average execution utility	KDECN	✔	LSTM-based and DDPG-based methods

the collected data into knowledge, and uses the knowledge to enable intelligent self-control and management capabilities of the network. Daniela M et al. innovatively incorporated KP to the SDN-based routing method and designed a routing decision optimization using reinforcement learning based on link state information [35]. In terms of network configuration, KDN has the advantage of providing closed-loop network management to solve the complicated management problems caused by the surge in the number of connected devices. The author Careglio et al. are committed to architecting a converged 5G-enabled infrastructure [36], which relies on ML technology in KDN architecture to complete automatic deployment, operation, monitoring and network troubleshooting, and realize intelligent network closed-loop management.

Furthermore, the deployment of KDN in optimizing resource management is mainly to provide decision-making model and reasoning process for the system. Since computation tasks and traffic in a data center network can be highly dynamic, accurate knowledge of tasks and traffic is required to make network orchestration more effective. Therefore, Lu et al. proposed knowledge-defined network choreography [37], carried out AI-assisted analysis through rich global view and telemetry information provided by KDN controller, and made more efficient network choreography decisions through deep learning of abstract knowledge. Rafiq et al. proposed an autonomous driving system based on KDN to achieve optimal path selection for deploying service function chaining and reactive traffic routing between edge clouds [38]. Herrera et al. proposed that the utilization of the KDN architecture can enhance the control and management of network resources, and identify video streaming services when data traffic increases, thus ensuring network performance [39].

However, there are few studies that make current decisions based on the future task request demands of UDs, which is conducive to maximizing long-term execution utility of the system and increasing resource utilization. Therefore, in the dynamic time-varying network system, it becomes extremely important to have global information and accurately predict the task requests and resource requirements of UDs. The former can be realized by deploying KDN network architecture, while the latter can be realized by predictive offloading. Gao et al. proposed dynamic offloading and resource allocation with traffic prediction in the multi-layer fog computing system. By predicting queues and arrival queues, the Lyapunov optimization problem was adopted to minimize the average energy consumption of all queues in the system [25]. As providing computation offloading and content caching services for autonomous vehicles, Tian et al. proposed to jointly optimize offloading and caching decisions on account of predicting future content popularity changing over time [26]. Li et al. adopted delayed online learning technology in MEC, and took the delay prediction as the input of queue-based offloading control strategy, thus reducing system execution cost [27].

In the large-scale access network, the resources and states in the system are time-varying, the traditional methods are difficult to solve such complicated problems. Deep Reinforcement Learning (DRL) is a naturally inspired approach to intelligent cognition and decision-making that is close to the human mind [28], [40]. Chen et al. adopted DRL to solve the joint optimization problem of dynamic computation offloading and resource allocation in MEC system [29], and proposed a temporal attentional deterministic policy gradient to tackle decision-making issue. Zhou et al. proposed to jointly optimize computation offloading and resource allocation in a multi-user dynamic MEC system to minimize system energy consumption by considering delay constraints and uncertain resource demands of heterogeneous computation tasks [41]. In the vehicular network, in order to tradeoff between the cost of energy consumption and data transmission delay, an adaptive computation offloading method based on DRL is proposed to solve the continuous action space problem [30]. Zhan et al. designed a decision-making method based on game theory in the scenario where users share their information, and proposed a DRL-based decentralized approach without prior knowledge to learn the optimal offloading policy in the case of no information sharing [31].

Different from these works, we propose a KDECN architecture and deploy ENs to optimize the UD information collection process and provide computation resources to the system. In order to maximize long-term average execution utility of the system, a DRL-based L-CORA algorithm is proposed in this paper. In addition, the system computation resources are shared among the whole UDs, a computing decision made by UD will have a certain impact on the subsequent UDs' decisions. It is necessary to accurately predict the task request demands of UDs in the future period, so a proposed LSTM-based TRDP method is deployed on the management plane. Considering the mixing continuous and discrete actions, we resort to a Deep Deterministic Policy Gradient (DDPG)-based algorithm to solve this problem. Table I shows a brief

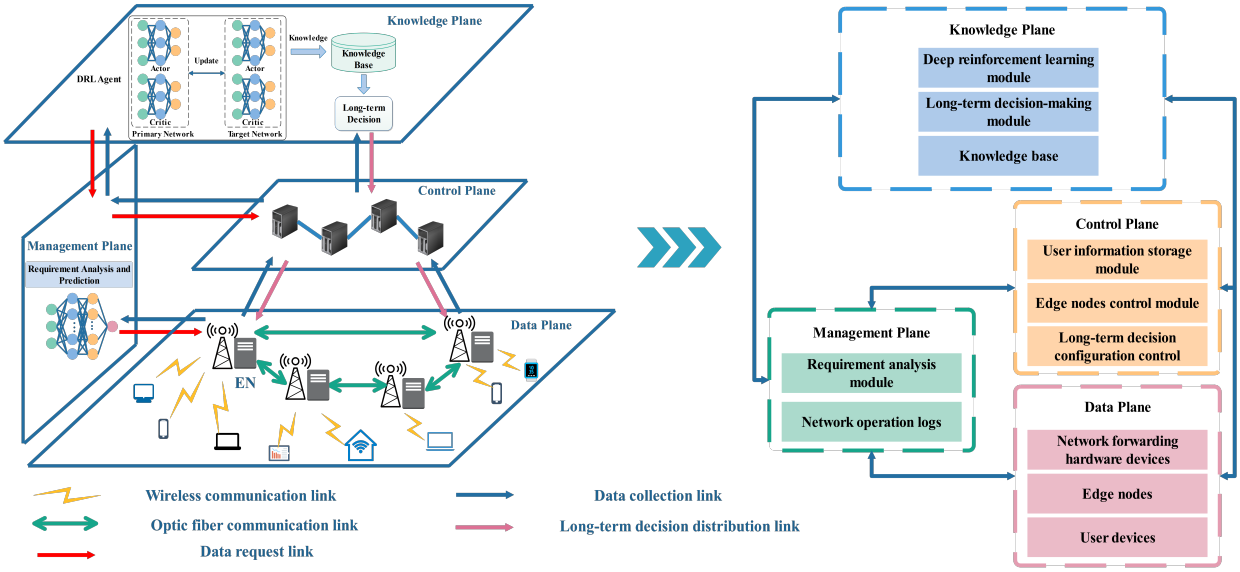

Fig. 1. System Model

TABLE II
NOTATIONS

Notations	Description
M	Number of ENs
N	Number of UDs
$r_{n,m}$	The wireless transmission rate between UD n and EN m
r_{wired}	The transmission rate between ENs through optical fiber links
C_n	Requested CPU
D_n	Task data size
T_n^{max}	The upper limit of task delay
F_n^{CPU}	Computation resources of UD n
$F_n^{EN_n^{com}}$	Computation resources allocated to user n by EN_n^{com}
$F_n^{EN_n^{exe}}$	Computation resources allocated to user n by EN_n^{exe}
F_{EN}^{max}	The maximum available computation resources of ENs
e_n^{CPU}	Energy consumption per CPU cycle
e_n^{tran}	Energy consumption of transmitting unit data
p_{EN}	Computation resource price per unit
$\alpha/\beta/\gamma$	Weights of execution time/ energy consumption/ price

comparison of some existing works.

III. System Model and Problem Formulation

This section introduces the system model of KDECN, and Fig. 1 illustrates the network architecture of the system. The KDECN architecture consists of four planes: data plane, control plane, management plane and KP. Since the KDECN architecture, communication and computation models are the key factors of computation offloading, we will introduce the three model in the following. In addition, the main parameters of this paper are shown in Table II.

A. KDECN Architecture

The data plane consists of the UDs, the network forwarding hardware devices (i.e., the forwarding device in the network communication link, which is omitted in Fig. 1) and the ENs. We set up a set $M = \{1, 2, 3, ..., m\}$ to represent the ENs with a certain computation power deployed near the micro base stations, and $N = \{1, 2, 3, ..., n\}$ to represent the UDs. With closer proximity to the UDs, ENs can provide UDs with computation offloading services and faster service response. Additionally, the relevant information of UDs is collected by the EN that chooses to provide services, and then the collected information will be preprocessed. Information preprocessing ensures the authenticity and validity of data and avoids unnecessary waste of resources caused by information redundancy.

The control plane deploys several controllers, including the UD information storage module, the EN control module, and the long-term decision configuration control module. The main function of the control plane is to update the matching rules and transformation rules with the data plane. The UD information storage module stores the basic information of all UDs in the system sent by ENs (i.e., UD location, network topology, computation power, etc.) and forms a complete global view. The EN control module stores the information related to ENs (i.e., node location, available computation resource, regions, waiting queues, etc.). The long-term decision configuration control module is designed to convert the long-term decision made by the KP into imperative language to program the ENs of the data plane through the south interface, so that the ENs can understand the computation decision made by the KP.

The management plane contains the requirement analysis module and the network operation logs. The requirement analysis module collects the information related to the UD requirements provided by ENs in the data plane (application type, data size, request frequency, request time, continuous time of task request type, etc.) and the global view provided by the control plane. Furthermore, the LSTM-based TRDP method is used to predict the requirements of UDs for network

applications due to the continuity of the task types requested by the UD. The importance of requirements analysis lies in providing raw data for training and critical network analysis for the KP, as well as reducing the complexity of knowledge creation. The network operation logs store historical data such as network operation status, network events and the long-term decisions adopted by UDs. The purpose of network operation logs is to provide the necessary data support for the smooth operation of the network and to promote the progress of learning model.

The KP consists of DRL module, long-term decision-making module and knowledge base. The core function of KP is to provide necessary management suggestions or decisions for the system through sufficient view and analysis of the network system provided by the control and management plane. In addition, the process of knowledge generation is obtained by continuous training and learning through DRL module, and the generated knowledge is stored in the knowledge base. When the decision-making encounters similar states and events, the knowledge learned in the knowledge base can be adopted to develop excellent computing decisions. In the KDECN architecture, the KP formulates reasonable computation offloading and computation resource allocation strategies for global UDs rather than individual UDs by considering the impact of one UD decision on subsequent decisions.

B. Communication Model

In this subsection, Analytic Hierarchy Process (AHP) [42] is firstly introduced to select the most appropriate device for sending relevant information and offloading task of UD n among the ENs capable of establishing communication links (EN_n^{com}). In addition, AHP can be used to elect the most suitable EN in the region to perform the offloading computation task of UD n (EN_n^{exe}). Incidentally, the EN_n^{com} and EN_n^{exe} elected by AHP can be the same device.

AHP is a kind of hierarchical weight analysis method to measure which EN can provide better service for UDs. AHP divides decision-related elements into three levels: objective level, criterion level and alternative level. Ultimately, AHP determines the priority of ENs selection according to the weight coefficient of each influencing factor in offloading the computation task, and selects the service node with the highest priority. According to the status of the UD and the requirements of the request task type, this paper takes the communication distance, communication interference and the available computation resources as the evaluation indexes of ENs. UDs pay different attention to evaluation indicators due to different types of computation tasks. Accordingly, in order to find the optimal service node, it is necessary to determine the selected priority of the service node according to the weight of each evaluation index of the task. Subsequently, the process of selecting service node EN by AHP method is introduced.

1) Establishment of AHP model: We describe the problem of electing optimal ENs as a three-level architecture model, with all candidate ENs as alternatives at the bottom; In the middle is the evaluation criterion level, which is the evaluation factor of EN; The top level is the objective level which is UD's choice of the candidate EN.

TABLE III
AVERAGE RANDOM CONSISTENCY INDEX RI

h	1	2	3	4	5	6	...
RI	0	0	0.58	0.90	1.12	1.24	...

2) Establish judgment matrix: Two judgment matrices need to be established, respectively: a) the judgment matrix of the evaluation criteria $Y = y_{i,j}$ is to compare the criteria with each other, that is, to judge the importance of criteria I to criteria J to the objective level; b) the judgment matrix of alternative nodes $Z = z_{i,j}$, is to compare two alternative nodes and judge the importance of alternative node I to node J to the same judgment criterion. Importance indicators are generally measured on a scale of 1-9 to judge the degree of importance. Nevertheless, due to the subjectivity of the judgment matrix, it may lead to inconsistency and large errors. Therefore, consistency checking is very important in AHP method. The consistency ratio CR can be used to determine whether the matrix has errors, which can be expressed as

$$CI = \frac{\lambda_{\max} - h}{h - 1}, \tag{1}$$

$$CR = \frac{CI}{RI}, \tag{2}$$

where $\lambda_{\max}$ is the largest eigenvalue of the judgment matrix, h is the number of evaluation indexes, that is, the order of the judgment matrix. Additionally, RI is the average random consistency index, whose value depends on the number of evaluation indexes, as shown in Table III. When $CR < 0.1$, the judgment matrix is considered to pass the consistency check. Otherwise, adjust the matrix until it is consistent.

3) Optimal EN selection: Finally, the weight vector is obtained by calculating the maximum eigenvalue and its corresponding eigenvalue vector. According to these weight vectors, the priority ranking of service nodes and the selection scheme of optimal service EN are obtained.

Next, two types of communication models in the system are introduced. One is the communication between UDs and ENs by establishing wireless links; The other is optical fiber link communication between ENs. The data transmission rate of offloading computation tasks between the UD n and the selected EN m is expressed as follows

$$r_{n,m} = W log_2(1 + \frac{B_n H_{n,m}}{N_0 + \sum_{i \in N, i \neq n} B_n H_{i,m}}), \tag{3}$$

where W is the channel bandwidth; B_n is the transmission power of the UD, it is assumed that the transmission power of UDs is the same; N_0 is the background noise power; $H_{n,m}$ is the channel gain between the UD n and the EN m; $\sum_{i \in N, i \neq n} B_n H_{i,m}$ represents the channel gain between UDs excluding UD n send data to EN m via wireless channel simultaneously. From Eq.(3), we can draw the conclusion that if the more other devices send data to the same target device EN m through the wireless channel at the same time,

which will cause severe interference and the reduction of transmission rate of UD n. In addition, we consider that ENs communicate with each other through optical fiber links, which can maintain stable and fast data transmission even if it is transmitted over long distances. The interference of optical fiber transmission is not considered in this paper, and the transmission rate is r_{wired}.

C. Computation Model

In this subsection, we will introduce the computation model. In our scenario, UD n will generate the computation task $I_n = (C_n, D_n, T_n^{max}), I_n \in I$ that needs to be processed in real-time. C_n is the number of Central Processing Unit (CPU) cycles required for performing computation task I_n; D_n is the data size of offloading computation task I_n; And T_n^{max} denotes the maximum tolerance delay for task completion. If the task is not completed within T_n^{max}, the task is judged to have failed. In addition, we assume that the number of CPU cycles required to execute a computation task is proportional to the data size of offloading computation task. In our scenario, UDs can execute computation tasks locally, or choose to offload computation tasks to ENs to perform computation tasks instead of themselves. Since computation tasks executed locally by UD can reduce the system resource load, we consider that computation tasks executed locally if the time of tasks executed locally can meet the maximum tolerable delay. The AHP method mentioned above can select the most appropriate EN_n^{com} to collect the information related to UD offloading, and can also provide UDs with computation resources. However, there may be a long computation queue of EN_n^{com} selected by UDs, and it may not be the optimal computation decision for the EN_n^{com} to execute the computation task I_n. Hence, EN_n^{com} can forward the computation task I_n to the relatively idle EN_n^{exe} to perform the computation task instead of the UD n. We define $K = k_n \in \left\{k_n^{local}, k_n^{EN_n^{com}}, k_n^{EN_n^{exe}}\right\}$ to be the execution scheme choice of UD n. Here, the set $\{local, EN_n^{com}, EN_n^{exe}\}$ indicates that computation tasks are executed at local, EN_n^{com} or by EN_n^{com} relays computation tasks to EN_n^{exe} through optcal fiber links. If $k_n = k_n^{\vartheta} = 1$, it indicates that UD n chooses ϑ to perform the computation task I_n, and $k_n^{local} + k_n^{EN_n^{com}} + k_n^{EN_n^{exe}} \leq 1$, that is, UD can only select one device to execute computation task. Next, we will focus on the time, energy consumption, and the price of renting computation resources of each strategy.

1) Local Execution: In the situation, if UD n can execute computation task I_n locally to meet the delay tolerance constraint, the UD will choose to execute task locally, and $k_n^{local} = 1$. Therefore, the execution time and energy consumption to execute the computation task I_n of UD n locally are respectively calculated as

$$T_n^{Loc} = C_n / F_n^{CPU}, \tag{4}$$

$$E_n^{Loc} = C_n \times e_n^{CPU}, \tag{5}$$

$$U_n^{local} = \alpha ln\left[1 + (T_n^{max} - T_n^{Loc})^{+}\right] - \beta E_n^{Loc}, \tag{6}$$

where the local execution time depends on the computation resources of UD n (F_n^{CPU}) and the energy consumption hinge on the energy consumption per CPU cycle (e_n^{CPU}). In addition, Eq.(6) is the utility function of UD n executing the computation task I_n locally, where $(\chi)^{+} = max\{0, \chi\}$, which means that the satisfaction of the execution time is kept non-negative; α and β are the utility weights of execution time and energy consumption, which are used to measure the importance of different cost factors.

2) Computation Offloading: In the circumstance, as the execution time for UD n to perform the computation task I_n locally exceeds the maximum tolerable delay, consider offloading the task to other devices over a wireless connection. Furthermore, each EN has set up a First-Come-First-Serve waiting queue to temporarily store computation tasks for subsequent offloads. After the EN_n^{com} receives the computation task I_n offloaded by UD n, it executes the task locally or forwards to EN_n^{exe} through optical fiber link according the task execution decision. Similar to [43] and [44], we assume that the transmission delay of the computation result is ignored, since the data size is small enough.

a) EN_n^{com} performs the computation task:

In this case, EN_n^{com} allocates computation resources to UD n to perform the task I_n. The execution time of the computation task I_n depends on the waiting time for the computing queue of EN_n^{com} ($T_{Wait}^{EN_n^{com}}$), the rate of transmitting the computation task over wireless link, and the amount of computation resource allocated ($F_n^{EN_n^{com}}$). The energy consumption of computation task I_n performed by EN_n^{com} is decided by the energy consumption of transmitting data of unit size (e_n^{tran}). Moreover, UDs need to pay EN_n^{com} for providing computation resources, which depends on the price per unit of computation resources (p_{EN}). Therefore, the execution time, energy consumption and price of EN_n^{com} to perform the computation task I_n instead of UD n can be expressed as

$$T_n^{EN_n^{com}} = T_{Wait}^{EN_n^{com}} + \frac{D_n}{r_{n,m}} + \frac{C_n}{F_n^{EN_n^{com}}}, \tag{7}$$

$$E_n^{EN_n^{com}} = D_n \times e_n^{tran}, \tag{8}$$

$$P_n^{EN_n^{com}} = F_n^{EN_n^{com}} \times p_{EN}, \tag{9}$$

and

$$U_n^{EN^{com}} = \alpha ln\left[1 + (T_n^{max} - T_n^{EN_n^{com}})^{+}\right] - \beta E_n^{EN_n^{com}} - \gamma P_n^{EN_n^{com}}, \tag{10}$$

where Eq.(10) is the utility function as EN_n^{com} performs the computation task I_n, and α, β and γ are the weight coefficients of execution time, energy consumption and price, respectively. These coefficients are utilized to fine-tune the significance of each parameter in the utility function according to UD preferences. For instance, as the weight assigned to execution time is higher, a shorter execution time is considered a more favorable computing decision.

b) EN_n^{exe} performs the computation task:

In this case, EN_n^{com} forwards the computation task to the optimal EN_n^{exe}, and EN_n^{exe} executes the computation task I_n of UD n. Accordingly, the costs and utility function of EN_n^{exe} to perform the computation task I_n is given as

$$T_n^{EN_n^{exe}} = T_{Wait}^{EN_n^{exe}} + \frac{D_n}{r_{n,m}} + \frac{D_n}{r_{wired}} + \frac{C_n}{F_n^{EN_n^{exe}}}, \quad (11)$$

$$E_n^{EN_n^{exe}} = D_n \times e_n^{tran}, \quad (12)$$

$$P_n^{EN_n^{exe}} = F_n^{EN_n^{exe}} \times p_{EN}, \quad (13)$$

and

$$U_n^{EN^{exe}} = \alpha ln\left[1 + (T_n^{max} - T_n^{EN_n^{exe}})^+\right] - \beta E_n^{EN_n^{exe}} - \gamma P_n^{EN_n^{exe}}. \quad (14)$$

Consequently, the execution utility of computation task I_n (U_n) is given as

$$U_n = k_n^{local} \times U_n^{local} + k_n^{EN_n^{com}} \times U_n^{EN^{com}} + k_n^{EN_n^{exe}} \times U_n^{EN^{exe}}. \quad (15)$$

D. Problem Formulation

The long-term computing decision-making aims at making optimal computation offloading and resource allocation strategies for global UDs with the goal of maximizing average execution utility over a period of time T. The long-term computing decision-making better considers the impact of one decision-making on other decisions in the future. Therefore, the long-term decision may not be a decision to maximize the current utility for a single UD in the short-term, it is a necessary solution for the long-term stable operation and reasonable utilization of resources of the whole system. Then, the problem of long-term decision-making can be formulated as

$$\max_{K,F} \frac{1}{N} \sum_i^N \sum_{t=1}^T U_n \quad (16)$$

$$\begin{aligned} &s.t. C1: F_n^{CPU} \geq 0, \forall n \in N, \\ &C2: 0 \leq F_n^{EN_n^{com}} \leq F_{EN}^{max}, \forall n \in N, \\ &C3: 0 \leq F_n^{EN_n^{exe}} \leq F_{EN}^{max}, \forall n \in N, \\ &C4: \sum_{j=1}^N F_j^{EN_n^{com}} \leq F_{EN}^{max}, \forall j \in N, \\ &C5: \sum_{j=1}^N F_j^{EN_n^{exe}} \leq F_{EN}^{max}, \forall j \in N, \\ &C6: k_n^{local} + k_n^{EN_n^{com}} + k_n^{EN_n^{exe}} \leq 1, \forall n \in N, \\ &C7: k_n^x = \{0,1\}, k_n^x \in K, n \in N, x \in \\ &\{local, EN_n^{com}, EN_n^{exe}\}, \\ &C8: k_n^{local} T_n^{Loc} + k_n^{EN_n^{com}} T_n^{EN_n^{com}} + k_n^{EN_n^{exe}} T_n^{EN_n^{exe}} \leq T_n^{max}. \end{aligned}$$

$C1$ indicates that the local computation resources of UD are non-negative. $C2$ and $C3$ are the constraints of computation resources allocated by EN_n^{com} and EN_n^{exe} to UD n. $C4$ and $C5$ are the constraints of total computation resources for EN_n^{com} and EN_n^{exe}, respectively. $C6$ and $C7$ indicate that the computation task of UD can only be executed on one device. And $C8$ is the time-delay constraint of the computation task.

Note that, K and F are decision variables related to UD n. To solve the problem stated in Eq.(16), it is necessary to find the optimal results for the offloading decision vector $K = k_n^{\vartheta}, n \in N$ in each time slot, the computation resource allocation vector $F = \left\{F_n^{\vartheta} | n \in N\right\}$ to maximize the average execution utility of the system. Specifically, the offloading decision variable K is a binary variable, and the computation resource allocation vector F is dynamically changing. It requires formulating large-scale global decisions for computation offloading and resource allocation based on the current state of the network. Furthermore, since current computing decisions affect the formulation of future UD computation task decisions, it is essential to make informed decisions for the current task by considering the predicted task request demands of future UDs. Therefore, the objective function is a mixed integer nonlinear problem and is undoubtedly NP-hard [45]. The feasible set of the problem is not convex, and the complexity exhibits exponential growth with the number of UDs. Due to the limitations of traditional methods in adapting to dynamically changing systems and formulating intelligent decisions for computation offloading and resource allocation, we propose an approach based on reinforcement learning to address the aforementioned problem.

IV. Long-term Computation Offloading and Computation Resource Allocation

In order to maximize the utility of multi-UD task execution over a long period of time, accurate UD demand analysis is necessary to support optimized long-term computation offloading and resource allocation decisions. Therefore, the LSTM-based TRDP method is deployed on the management plane of the KDECN architecture to provide the necessary demand analysis for KP learning and training. The predicted UD request task type reflects the UD preference and the characteristics of continuous task requests. The details of the method to predict the UD demand are elaborated below.

A. LSTM-Based TRDP Method

The content or application with high popularity in a region is most likely to be requested. Hence, most existing work assumes that the popularity of content generally follows Zipf distribution [46]. Considering the dynamic resource requirements, the probability of the UD request application type is time-varying, a pure distribution will not always be realized. Moreover, requests for the same type of application tend to be continuous for a certain amount of time, e.g., game-type applications often require long-term continuous requests. Therefore, it is a challenge to accurately predict the UD tasks request quantity in each future time slot and maximize long-term execution utility by optimizing the computation offloading and resource allocation strategies.

For this reason, we proposed a TRDP method on account of spatial-temporal and task request continuity deployed on the management plane. LSTM relies on the cyclically connected subnet to achieve long time dependence, which contains functional modules of the memory unit and gate. The gate control mechanism controls the flow of information between the input, output and the cell memory. LSTM processes the input sequence by adding new information to the memory and controlling the state of information memory, the abandoned state of the old information, and the available state of the

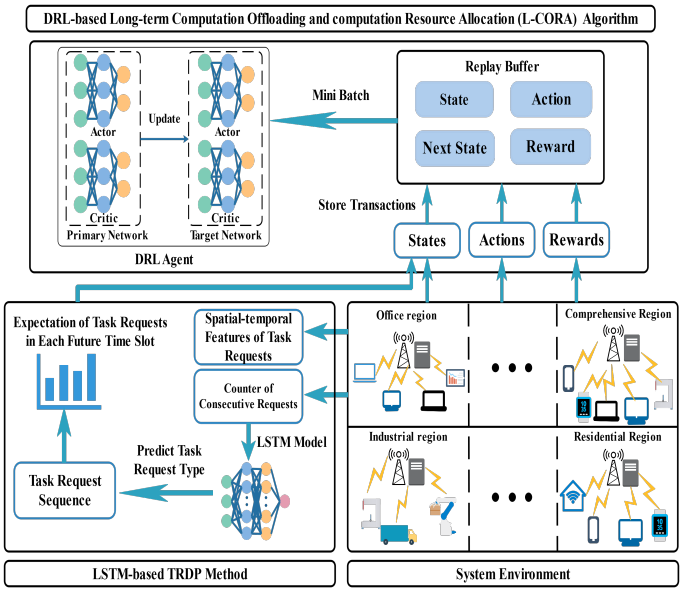

Fig. 2. Long-term computation offloading and computation resource allocation decision-making process.

current information, thus realizing the prediction that depends on the time-series. Since the UD request task type has the characteristics of continuity, the TRDP method is designed based on LSTM model to train and predict the future UD request task quantity.

Additionally, in order to capture the distribution of requests in different regions, the whole area is divided into several equally sized grids based on functional areas such as industrial, residential, office, and comprehensive regions. We define $\omega_x^g(t)$ to be the count of the request task type x generated by the UDs at time slot t in region g. Moreover, in order to reflect the impact of different tasks on request continuity, we set up a counter to measure the number of time slots for consecutively selecting the same type of task. Since the continuous request time of a task type has an upper limit, the probability of continuously requesting the same task will gradually decrease with the increase of time. The history request task record and related feature data are fed into the LSTM model for training. With a lot of training, the task request probability $p_x^g(t)$ of task type x in the region g in time slot t can be calculated and updated according to the task type selected by UDs in a future time slot. The calculation of $p_x^g(t)$ can be defined as

$$p_x^g(t) = \frac{\omega_x^g(t)}{\sum_{i=1}^{X} \omega_i^g(t)}, \forall k \in K. \tag{17}$$

Therefore, the request probability of all task types in the time slot t of different regions g can be expressed as follows:

$$P(t) = \left\{ p_1^1, \ldots, p_X^1, p_1^2, \ldots, p_X^2, \ldots, p_1^g, \ldots, p_X^g \right\}. \tag{18}$$

We consider adopting the predicted task quantity expectation of the UDs to reflect the impact of the task quantity requested by the UDs in future time slots on the current offloading decision-making. The expectation of requested task type x generated from UDs in the region g at time slot t, which is given as

$$E_n(t) = \sum_n^N p_x^g(t) C_n^x. \tag{19}$$

Afterwards, according to the predicted expectation of task request quantity, an online computation offloading and computation resource allocation algorithm is proposed to maximize the average execution utility of UDs over a period of time and improve the QoS of task execution in the system. The long-term computation offloading and computation resource allocation decision-making process is shown in Fig. 2. Then, the developed optimization problem is modeled as a Markov Decision Process (MDP), and the components are defined for the state in combination with the predicted request task quantity expectation and UD personalization demands.

B. DRL-based Long-term Computing Decision-making

Since the Eq.(16) is NP-hard, it is difficult to effectively maximize long-term average execution utility of the system based on traditional methods such as game theory. Nowadays, DRL has increasingly become a promising method and is widely adopted in various scenarios to tackle problems. specifically, DRL can solves problems in a human-like way by combining the perception of deep learning and the decision-making power of reinforcement learning.

Inspired by the powerful decision-making capacity of DRL, we deploy the DRL module on KP to make long-term decision. According to the global views and UD demand analysis provided by the control and management planes, the DRL module gets the optimal policy after a lot of training and stores it in the knowledge base, then the policy is continuously updated to adapt to the dynamic changes of the system. Next, we approximate the optimization problem as MDP, and define the predicted task request quantity expectation and the personalized demands of UDs as the components of state to maximize long-term execution utility.

Firstly, the states, actions and rewards are defined in detail.

a) States: The state space reflects the observed network environment. In our system, two types of parameters are mainly considered, which related to the utility of task execution and the computation resource allocation. The state space s_t can be expressed as:

$$s_t = \left\{ H_{n,m}(t), Z_n(t), E_n(t), \sigma_n(t), T_m^{wait}(t), F_m(t) \right\}, \tag{20}$$

where $H_{n,m}(t)$ represents the signal to noise of wireless communication between UD n and EN m, which can establish a wireless link; $Z_n(t)$ indicates the task type requested by UD n; $E_n(t)$ represents the expectation of task quantity predicted by LSTM-based TRDP method, so as to reflect the demand for resources of UDs in the future; $\sigma_n(t)$ represents the personalized demand of different cost weights by UDs; $T_m^{wait}(t)$ is the waiting time for queue release of EN m at the time t; And $F_m(t)$ is the available computation resource of EN m.

b) Actions: According to the observed state of the environment, the agent selects EN_n^{com} or EN_n^{exe} elected by

AHP method to perform the computation task offloaded by UD n and how many computation resources should be allocated to UDs. Hence, the action space a_t is given by

$$a_t = \{k_{n,m}(t), f_{n,m}(t)\}, \tag{21}$$

where $k_{n,m}(t)$ is the binary discrete action indicating whether EN_n^{com} or EN_n^{exe} is selected by UD n as the device to perform the offloading computation task; And $f_{n,m}(t)$ is the computation resources that EN m allocates to UDs, which is a continuous variable.

c) Rewards: In the long-term decision-making problem, the goal is to maximize the average execution utility of the whole system over a long period of time. The long-term reward space can be expressed as $R_t = \sum_t^T \gamma^t r(s_t, a_t)$. Here, $\gamma \in [0, 1]$ is the discount factor; $r(s_t, a_t)$ is the immediate reward obtained by the agent taking action a_t and it defined as

$$r(s_t, a_t) = \begin{cases} punishment, if\ T_n > T_n^{max}, \\ 0, if\ T_n < T_n^{max}, t < T, \\ \frac{1}{A}\sum_t^T U_n, if\ T_n < T_n^{max}, t = T, \end{cases} \tag{22}$$

where A is the number of the tasks requested by the UDs in time slot T. In order to make the agent better achieve the system goal, we define that when the execution time of the task is greater than the maximum tolerable delay, the agent will get a punishment. Moreover, under the above condition, the immediate reward of the last time slot in an episode is the average utility of the tasks, while the other time slots are 0. The purpose of the immediate reward is to maximize the average utility of the system, not to maximize individual benefits. After UDs have taken actions during the time slot T, the agent calculates the average utility based on the utility function and begins the next episode.

According to the expectation of the requested task quantity in a long period time T predicted by LSTM-based TRDP method and the individual demands of the UDs, the DRL method is adopted to obtain the optimal solution. In this paper, considering the existence of discrete and continuous mixed action space, the DDPG algorithm is selected. DDPG algorithm is a model-free and similar to actor-critic framework, that can tackle the continuous actions by the policy gradient method. Furthermore, DDPG is divided into two major networks: policy network and value network, which is an extension of DQN, including actor, critic and replay buffer. The Actor network learns to output a deterministic action based on the current state, rather than a probability distribution over actions, to enable learning in continuous action spaces. Consequently, in order to maximize the reward and obtain the optimal solution, we designed a DDPG-based L-CORA algorithm. The details of the DDPG-based algorithm are introduced as follows.

The information of the environment is observed by the agent, including the UD demand analysis and prediction data of the management plane, the status and available computation resources of the ENs and UDs provided by the control plane. After that, the actor network maps the current state to the action through the exploration policy, and selects the action a_t by adding noise to obtain better exploration performance of the behavior state, which expressed as

$$a_t = \mu(s_t \mid \theta^\mu) + N_t, \tag{23}$$

where $\mu(s_t \mid \theta^\mu)$ represents the policy explored by actor, and N_t is the random noise. After the action a_t is obtained according to the policy, the feedback reward can be calculated from the environment, and the next state s_{t+1} is updated. The four-tuple transitions are then stored in the replay buffer. When the length of experience tuples reaches the limit, a minibatch is randomly sampled for training to update network parameters. Afterwards, critic network updates the neural network parameter θ^Q by reducing the loss function, which is given by

$$L = E\left[(y_t - Q(s_t, a_t \mid \theta^Q))^2\right], \tag{24}$$

where $Q(s_t, a_t \mid \theta^Q)$ is the actor value function, which represents the score for the policy. Each long time period is a training episode, and the target value y_t of the training $Q^{'}$ and $\mu^{'}$ of the target critic network is represented as

$$y_t = r(s_t, a_t) + \gamma Q^{'}(s_{t+1}, \mu^{'}(s_{t+1} \mid \theta^{\mu^{'}}) \mid \theta^{Q^{'}}). \tag{25}$$

Then according to the $Q(s_t, a_t \mid \theta^Q)$ and the replay buffer tuples, the primary actor network parameter θ^μ updates by using policy gradient. The policy gradient is expressed as

$$\nabla_{\theta^\mu} J = \frac{1}{N}\sum_t^T \nabla_a Q(s, a \mid \theta^Q) \mid_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s \mid \theta^\mu) \mid_{s_t}. \tag{26}$$

In each iteration, the parameters of the target neural network are updated by soft update scheme, which is presented as

$$\theta^{\mu^{'}} = \omega\theta^\mu + (1-\omega)\theta^{\mu^{'}}, \tag{27}$$

$$\theta^{Q^{'}} = \omega\theta^Q + (1-\omega)\theta^{Q^{'}}, \tag{28}$$

where ω is the update coefficient.

In our scenario, we first judge whether the execution time of the local computation task executed by UD meets the maximum tolerable delay constraint. If so, the computing task is executed locally; Otherwise, L-CORA is executed to formulate the optimal computation offloading and resource allocation policy. The process of L-CORA algorithm is shown in Algorithm 1. The L-CORA algorithm starts by initializing four networks and a replay buffer. Next, according to the LSTM-based TRDP method, the predicted task quantity of UDs in each future time slots are obtained. As a parameter in the state space, the task quantity expectation of the future time slot affects the current computation decision-making and ensures the maximization of the long-term average execution utility. For each episode, the agent selects an action in each time slot t, and the immediate reward and next state are obtained. Afterwards, the four-tuple transition is stored in the replay buffer. As the network parameters are to be updated, the transitions are sampled from the buffer for training. Ultimately, the critic and actor networks, and the corresponding target networks are updated successively, each episode is cycled

Algorithm 1: Long-term Computation Offloading and computation Resource Allocation (L-CORA) Algorithm

Initialize replay memory;
Initialize actor network $\mu(s|\theta^{\mu})$ and critic network $Q(s,a|\theta^{\mu})$ with coefficients θ^{μ} and θ^{μ};
Initialize target networks $\mu^{'}$ and $Q^{'}$ with coefficients $\theta^{\mu^{'}} \leftarrow \theta^{\mu}$ and $\theta^{Q^{'}} \leftarrow \theta^{Q}$;
for *episode=1,M* **do**
  Receive the system initial observation state s;
  for *t=1,T* **do**
    Predict the task type of UDs in future T;
    Compute the expectation of the future task quantity of UDs in region g;
    Collect the state s;
    Select action a_t according to the current policy $\mu(s|\theta^{\mu})$ and exploration noise;
    Obtain the observe reward $r(s_t,a_t)$ and new stat S_{t+1};
    Store the transitions $(s_t,a_t,r(s_t,a_t),s_{t+1})$ into replay buffer;
    Sample a random N transitions from the replay buffer;
    Calculate the y_t by Eq.(25);
    Update critic parameter θ^{Q} to minimize the loss function by Eq.(24);
    Update the actor policy $\mu(s|\theta^{\mu})$ by Eq.(26);
    Update the target networks by Eq.(27) and Eq.(28);
  end
end

until the algorithm ends. After the algorithm is completed, the optimized computation offloading and resource allocation policy obtained from training is stored in the knowledge base. Since the system is dynamic, Algorithm 1 needs to be executed repeatedly at certain intervals to update the computing policy.

C. Complexity Analysis

The complexity of the proposed LSTM-based TRDP method and L-CORA algorithm is based on the computational complexity of neural networks. For the LSTM-based TRDP method, the complexity of LSTM per time step depends on the computational complexity of the input gate, output gate, forget gate, and cell update, set F_i is the input dimension and F_o is the output dimension. Additionally, there is a dense layer after the LSTM with a complexity of $O(F_o q)$, where q is the number of neurons in dense layer. Therefore, the time complexity of the LSTM-based TRDP method can be represented as $O(4(F_i F_o + F_o^2 + F_o) + F_o q)$. For the L-CORA algorithm, let I represent the number of multiplications in the propagation and backpropagation in neural networks by DRL agent, and M represent the number of episodes. Each episode executes T time steps. Within one episode, the LSTM-based TRDP method is executed to predict the future request task quantity of UDs. Therefore, the complexity of L-CORA is $O(IMT(4(F_i F_o + F_o^2 + F_o) + F_o q))$.

TABLE IV
EXPERIMENTAL EVALUATION

Simulation Parameter	Value
Computation cycles for tasks	[200,2250] Megacycles [47]
Data size for tasks	[300,1500] K bytes [47]
Upper limit of time delay tolerance	[0.8,1.0] s
Wireless channel bandwidth	20 MHz [43]
Maximum available computation resource of ENs	30 GHz
The computation resource price coefficient of EN	0.03 $/GHz [12]
Local computing energy per cycle	$0.8 * 10^{-13} J/cycle$
Transmission energy consumption per byte	$0.8 * 10^{-9} J/byte$

V. NUMERICAL RESULTS

A. Configuration

In this section, the performance of L-CORA algorithm under the KDECN architecture is evaluated. The performance evaluation of L-CORA algorithm is based on the telecom real-world datasets in Shanghai, China, inspired by [26] using real-world datasets to simulate the distribution of ENs with latitude from 30.912831 to 30.815709, and longitude from 121.141435 to 121.37282. In the simulation, the channel bandwidth of wireless communication is set as $W = 20MHz$; The background noise is set to $N_0 = -100dB$; The transmission rate of the optical fiber link is $r_{wired} = 50Mb/s$; And the wireless transmission power of the UD is $100mWatts$, the computing power of the UD is $[0.5, 1]GHz$. Since the local computing capacity may not guarantee that the task execution time is less than the maximum delay tolerance, the computation task can be offloaded to the EN selected by the AHP method. For the LSTM-based TRDP method, we constructed two hidden layers with 32 and 16 LSTM cells, respectively, and set a dense layer as the output layer. The DDPG-based L-CORA algorithm, we consider a fully connected network with one input layer, two hidden layers and one output layer, where the hidden layers have 400 and 300 hidden neurons. Moreover, we train the minibatch with the size of 256, the replay buffer with the size of 10^6, and the discount factor γ is set as 0.99. The simulation parameters of this article are shown in Table IV by considering the experimental parameters of related work [12], [47]. Moreover, The simulations are implemented in TensorFlow 1.2.0 with Python 3.6.

B. Performance

In order to verify our LSTM-based TRDP method, we compared it with three other commonly used baseline models, including the Depth Neural Network (DNN) model, the time learning model 1-D convolution and the Gate Recurrent Unit (GRU). Fig. 3 illustrates the prediction error comparison on other baseline models with different batch sizes under the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) evaluation indexes. It can be observed that our proposed TRDP method outperforms the other three baseline models in terms of RMSE and MAE index. In general, training

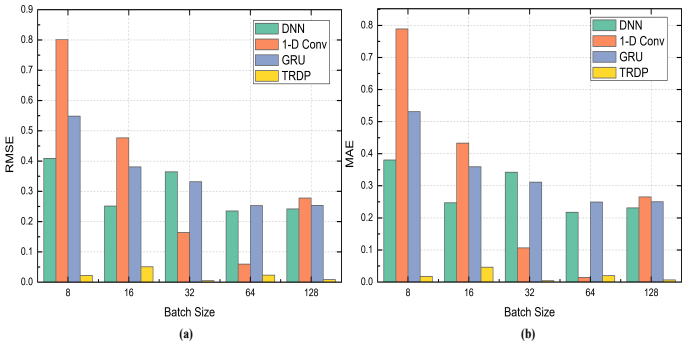

Fig. 3. Prediction error with different batch size.

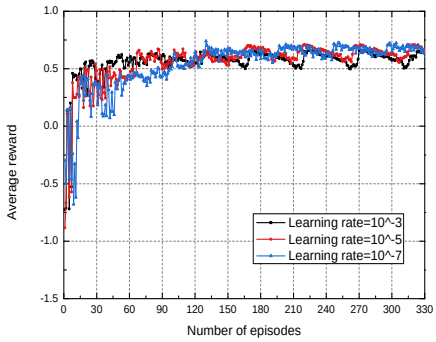

Fig. 4. Convergence performance of L-CORA at different learning rates.

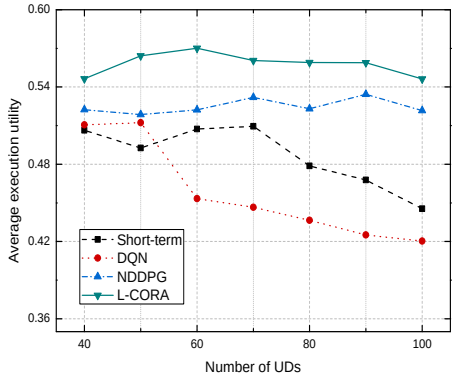

Fig. 5. Average execution utility versus the number of UDs in one region.

efficiency can be improved by increasing batch size. However, when the batch size is 32, the error of the TRDP method is the smallest, and the performance is reduced at larger batch sizes. Although the other three baseline models can capture the spatial-temporal features of data, their performance is inferior to that of the TRDP method in the analysis of large-scale and time-dependent scenarios. The LSTM-based TRDP method solves long-term dependence of spatial-temporal correlation and task demand, and has high performance in analyzing long-term task request demand of large-scale UDs.

Then, we evaluate L-CORA algorithm performance. Specifically, we evaluate the performance of the algorithm using metrics of the system's average execution utility, execution time, success rate and reward. In addition, we compared with three other baselines, as follows:

- Short-term decision (Short-term) [12]: Short-term decision-making is to make computation offloading and resource allocation decisions with the goal of maximizing the execution utility of the current computation task. Short-term decision describes the computing decision-making problem as a computation offloading strategy game, and constructs partial Lagrange function and bisection method to make computation resource allocation scheme.
- DQN-based Algorithm (DQN) [30]: A DQN-based computation offloading and resource allocation algorithm has been implemented. Since the action space of DQN is discrete, we quantify the continuous action in the simulation environment and approximate replace the real value with a finite number of discrete values. Thus we allocated computation resources from the action space into 10 levels in the simulation environment.
- No prediction DDPG (NDDPG) [43]: In order to verify the influence of task request demand prediction on computation offloading and resource allocation scheme formulation, we simulate computation offloading and resource allocation algorithms without predicting UD request demand.

Firstly, Fig. 4 shows the convergence performance of our proposed algorithm under different learning rates. In the L-CORA algorithm, the actor network selects the action according to the policy, and the critic network evaluates the performance of the action. The learning rate of the actor and critic network will affect the convergence performance and convergence speed. Intuitively, when the episode is from 0 to 120, the average execution utility of the system rises rapidly. As the episode exceeds 120, the curve tends to stabilize. From Fig. 4, it can be observed that when the learning rate is lower, the convergence performance will be better, but at the same time, the convergence speed will be slower. As the learning rate is 10^{-7}, the convergence performance is better, and the slow convergence speed will also lead to reduced algorithm performance. Therefore, the learning rate of 10^{-5} is a relatively optimal choice.

Fig. 5 illustrates a comparison of the average execution utility of the system with different numbers of UDs within a single functional region. We conducted computing performance evaluations for 40 to 100 UDs in one functional region, with 5 ENs deployed in this region. Since L-CORA is based on UD task request demand prediction, and aims to maximize long-term average execution utility of the system to develop computation offloading and resource allocation schemes, it shows excellent performance as the number of UDs increases. With the increase of the number of UDs, the average execution utility of our proposed L-CORA is increased from 2.7% to 35.6% compared with other baselines.

As shown in Fig. 6, the performance of task execution in the system as the number of large-scale UDs changes across regions. We primarily evaluate the performance of the L-CORA algorithm by comparing it with other baseline

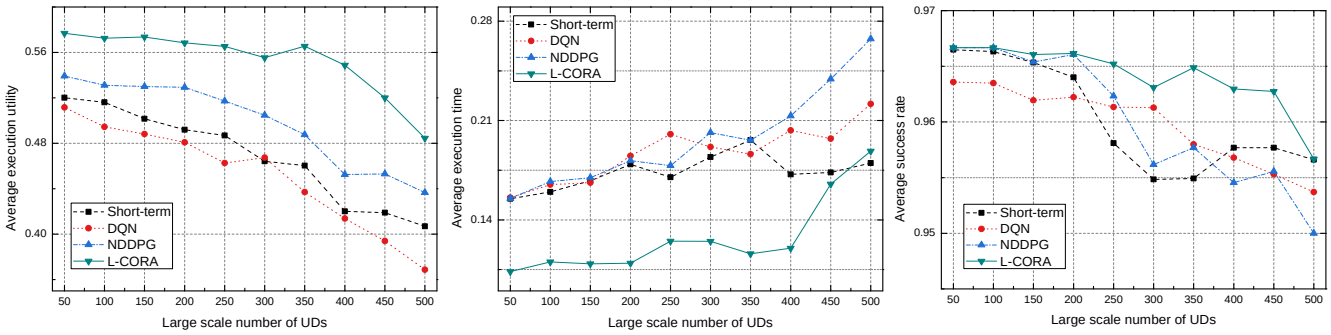

Fig. 6. Average execution utility, execution time and success rate of large-scale UDs.

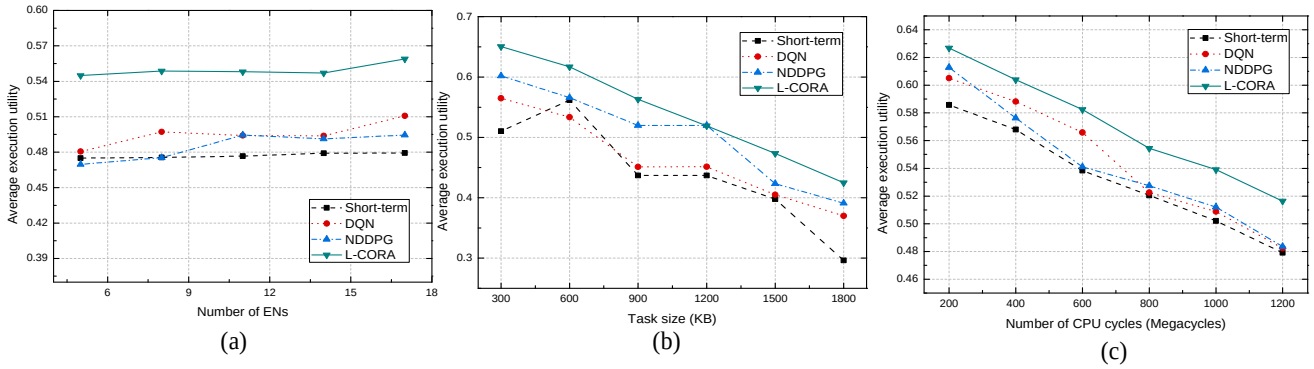

Fig. 7. Average execution utility with different number of ENs, offload task sizes and task CPU cycles.

models in terms of average execution utility, average execution time, and average task execution success rate. We set the number of UDs ranging from 50 to 500 and deployed 24 ENs in the system, which consists of multiple functional regions. It can be observed from the figure that although the increase of the number of UDs will have a certain impact on system computing performance, the L-CORA algorithm can still maximize long-term average execution utility and reduce the task execution time. Due to the limited action space of DQN algorithm, it is difficult to explore better strategies in joint optimization of computation offloading and computation resource allocation. In addition, short-term decision-making aims at maximizing the current decision utility, and traditional algorithms are adopted to allocate computation resources, which cannot make decisions to optimize the long-term system performance. Moreover, compared to NDDPG, L-CORA achieves better performance by considering the influence between UD decisions. The numerical results show that compared with other baselines, the average execution utility of the proposed L-CORA improves from 6.8% to 26.5%, the average execution time is reduced by 4.94% to 41.98%, and the average task execution success rate is increased by 1.04%.

In addition, Fig. 7 (a) shows the influence of the number of EN in the region on the average execution utility of the system. We conducted experiments by fixing the number of UDs at 100 within a single functional region and varying the deployment of ENs from 5 to 17. It can be observed that with the increase of the number of EN, the average execution utility of the system shows an upward trend. This is because the increase in the number of EN will increase the computation resources provided by the region for UD to reduce the task execution time. We set that there are 100 UDs in this region, as the number of EN is 5, L-CORA increases by 13% to 16% compared with other algorithms; When the number of EN is 17, the average execution utility of our algorithm L-CORA is 9.4% to 16.6% higher than other algorithms. The experimental results show that our proposed algorithms can formulate excellent long-term computation offloading and resource allocation schemes to maximize the execution utility of the system in the case of changing available computation resources.

In Fig. 7 (b) and 7 (c), we examine the impact of offloaded data size and the number of CPU cycles changed simultaneously on the system average execution utility in the four models. In the simulation, we set a fixed number of 100 UDs and deployed 8 ENs within a single functional area, and we varied the offloaded data size and the number of CPU processing cycles respectively. It can be observed in Fig. 7 (b) that the increase of data quantity affects the network transmission delay and energy consumption, thus reducing the execution utility of computation offloading. The L-CORA algorithm makes current computing decisions based on the future task requests of UDs, which enables the system to explore better performance to the maximum extent, thus ensuring the QoS of the system. The numerical results show that the average execution utility of L-CORA increases by

63.6% compared with other algorithms. Fig. 7 (c) illustrates that with the change of the number of CPU cycles required to perform tasks, it can be observed that the average execution utility of the system presents a downward trend. The reason is that the increase of task size affects the execution time of the task, and the proposed L-CORA algorithm can reasonably allocate computation resources for UDs under different task sizes, thus ensuring the system service performance. The data shows that the average execution utility of our proposed L-CORA is increased by 7.3% compared with other algorithms.

VI. Conclusion

In this paper, we innovatively propose the KDECN architecture to better provide computing services for large-scale IoT UDs. In addition, considering that the making of a computation decision has a certain impact on the making of subsequent computation offloading decisions, we propose LSTM-based TRDP method deployed on the management plane. TRDP method can predict and calculate the expectation of requested task quantity in each future time slot, which can provide the KP with necessary UD data analysis and prediction. In order to maximize long-term average execution utility of the system, we propose the L-CORA algorithm deployed on KP. L-CORA algorithm considers the future task request and personalized demand for task execution cost of UDs to formulate the optimal computation offloading and computation resource allocation scheme. Our proposed KDECN architecture and L-CORA algorithm can solve the complex problem of large-scale UD computation decision-making, and better mediate the target conflict between UDs and operators in the system. Numerical results show that our proposed L-CORA can effectively improve the average execution utility of the system.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No.62032013, 92267206, 61872073, 62202089, U22A2004; the Major International (Regional) Joint Research Project of NSFC under Grant No.71620107003.

References

[1] Y. Liu, Z. Su, and Y. Wang, "Energy-efficient and physical layer secure computation offloading in blockchain-empowered internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[2] F. Gu, X. Hu, M. Ramezani, D. Acharya, and J. Shang, "Indoor localization improved by spatial context—a survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 64:1–35, 2019.

[3] T. Chen, Y. Yang, H. Zhang, H. Kim, and K. Horneman, "Network energy saving technologies for green wireless access networks," *IEEE Wireless Communications*, vol. 18, no. 5, pp. 30–38, 2011.

[4] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma, and Z. Liu, "A novel cost optimization strategy for sdn-enabled uav-assisted vehicular computation offloading," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3664–3674, 2021.

[5] P. H. L. Rettore, J. Loevenich, and R. R. F. Lopes, "Tnt: A tactical network test platform to evaluate military systems over ever-changing scenarios," *IEEE Access*, vol. 10, pp. 100 939–100 954, 2022.

[6] A. Hazra and T. Amgoth, "Ceco: Cost-efficient computation offloading of iot applications in green industrial fog networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6255–6263, 2022.

[7] Q. He, H. Fang, J. Zhang, and X. Wang, "Dynamic opinion maximization in social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 350–361, 2023.

[8] T. He, H. Khamfroush, S. Wang, T. F. L. Porta, and S. Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources," *international conference on distributed computing systems*, 2018.

[9] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 207–215.

[10] P. Dai, K. Hu, X. Wu, H. Xing, F. Teng, and Z. Yu, "A probabilistic approach for cooperative computation offloading in mec-assisted vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 899–911, 2022.

[11] L. Zhang, R. Chai, T. Yang, and Q. Chen, "Min-max worst-case design for computation offloading in multi-user mec system," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 1075–1080.

[12] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, 2019.

[13] C. Tang, C. Zhu, N. Zhang, M. Guizani, and J. J. P. C. Rodrigues, "Sdn-assisted mobile edge computing for collaborative computation offloading in industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 253–24 263, 2022.

[14] J. Baek and G. Kaddoum, "Online partial offloading and task scheduling in sdn-fog networks with deep recurrent reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 578–11 589, 2022.

[15] L. Zhao, K. Yang, Z. Tan, H. Song, A. Al-Dubai, A. Y. Zomaya, and X. Li, "Vehicular computation offloading for industrial mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7871–7881, 2021.

[16] J. Xu, S. Wang, N. Zhang, F. Yang, and X. Shen, "Reward or penalty: Aligning incentives of stakeholders in crowdsourcing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 974–985, 2019.

[17] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, 2017.

[18] J. Du, C. Jiang, A. Benslimane, S. Guo, and Y. Ren, "Sdn-based resource allocation in edge and cloud computing systems: An evolutionary stackelberg differential game approach," *IEEE/ACM Transactions on Networking*, vol. 30, no. 4, pp. 1613–1628, 2022.

[19] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 3–10, 2003.

[20] D. Hevey, "Network analysis: a brief overview and tutorial," *Health Psychology and Behavioral Medicine*, vol. 6, no. 1, pp. 301–328, 2018.

[21] O. Wu, Z. Li, J. Wang, Y. Zheng, M. Li, and Q. Huang, "Vision and challenges for knowledge centric networking.pdf," *Journal: IEEE wireless communications*, 2018.

[22] Q. Li, H. Fang, D. Li, J. Peng, J. Kong, W. Lu, and Z. Zhu, "Scalable knowledge-defined orchestration for hybrid optical–electrical datacenter networks," *Journal of Optical Communications and Networking*, vol. 12, no. 2, pp. A113–A122, 2020.

[23] S. Ashtari, I. Zhou, M. Abolhasan, N. Shariati, J. Lipman, and W. Ni, "Knowledge-defined networking: Applications, challenges and future work," *Array*, 2022.

[24] Q. He, Y. Wang, X. Wang, W. Xu, F. Li, K. Yang, and L. Ma, "Routing optimization with deep reinforcement learning in knowledge defined networking," *IEEE Transactions on Mobile Computing*, pp. 1–12, 2023.

[25] 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, 2020.

[26] H. Tian, X. Xu, L. Qi, X. Zhang, W. Dou, S. Yu, and Q. Ni, "Copace: Edge computation offloading and caching for self-driving with deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 13 281–13 293, 2021.

[27] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2833–2849, 2020.

[28] J. Shi, J. Du, Y. Shen, J. Wang, J. Yuan, and Z. Han, "Drl-based v2v computation offloading for blockchain-enabled vehicular networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

[29] J. Chen, H. Xing, Z. Xiao, L. Xu, and T. Tao, "A drl agent for jointly optimizing computation offloading and resource allocation in mec," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 508–17 524, 2021.
[30] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for mec in heterogeneous vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7916–7929, 2020.
[31] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Transactions on Computers*, vol. 69, no. 6, pp. 883–893, 2020.
[32] J. Hyun and J. W.-K. Hong, "Knowledge-defined networking using in-band network telemetry," pp. 54–57, 2017.
[33] A. Duque-Torres, F. Amezquita-Suárez, O. M. Caicedo Rendon, A. Ordóñez, and W. Y. Campo, "An approach based on knowledge-defined networking for identifying heavy-hitter flows in data center networks," *Applied Sciences*, vol. 9, no. 22, p. 4808, 2019.
[34] B. Cao, J. Zhang, X. Liu, Z. Sun, W. Cao, R. M. Nowak, and Z. Lv, "Edge–cloud resource scheduling in space–air–ground-integrated networks for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5765–5772, 2022.
[35] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. da Fonseca, "Intelligent routing based on reinforcement learning for software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 870–881, 2020.
[36] D. Careglio, S. Spadaro, A. Cabellos, J. A. Lazaro, J. Perello, P. Barlet, J. M. Gené, and J. Paillisse, "Alliance project: Architecting a knowledge-defined 5g-enabled network infrastructure," *international conference on transparent optical networks*, 2018.
[37] W. Lu, L. Liang, B. Kong, B. Li, and Z. Zhu, "Ai-assisted knowledge-defined network orchestration for energy-efficient data center networks," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 86–92, 2020.
[38] A. Rafiq, S. Rehman, R. Young, W.-C. Song, M. A. Khan, S. Kadry, and G. Srivastava, "Knowledge defined networks on the edge for service function chaining and reactive traffic steering," *Cluster Computing*, vol. 26, no. 1, pp. 613–634, 2023.
[39] L. M. Castaneda Herrera, A. Duque Torres, and W. Y. Campo Munoz, "An approach based on knowledge-defined networking for identifying video streaming flows in 5g networks," *IEEE Latin America Transactions*, vol. 19, no. 10, pp. 1737–1744, 2021.
[40] Z. Gao, G. Wu, Y. Shen, H. Zhang, S. Shen, and Q. Cao, "Drl-based optimization of privacy protection and computation performance in mec computation offloading," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.
[41] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1517–1530, 2022.
[42] H. Zhang, Z. Wang, and K. Liu, "V2x offloading and resource allocation in sdn-assisted mec-based vehicular networks," *China Communications*, vol. 17, no. 5, pp. 266–283, 2020.
[43] L. Chen, G. Gong, K. Jiang, H. Zhou, and R. Chen, "Ddpg-based computation offloading and service caching in mobile edge computing," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.
[44] X. Wang, Z. Ning, and S. Guo, "Multi-agent imitation learning for pervasive edge computing: A decentralized computation offloading algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 411–425, 2020.
[45] Y. Peng, X. Tang, Y. Zhou, J. Li, Y. Qi, L. Liu, and H. Lin, "Computing and communication cost-aware service migration enabled by transfer reinforcement learning for dynamic vehicular edge computing networks," *IEEE Transactions on Mobile Computing*, 2022.
[46] Q. Li, Y. Zhang, Y. Li, Y. Xiao, and X. Ge, "Capacity-aware edge caching in fog computing networks," *IEEE Transactions on Vehicular Technology*, 2020.
[47] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.