Joint Service Migration and Resource Allocation in Edge IoT System Based on Deep Reinforcement Learning

Fangzheng Liu, Hao Yu, Jiwei Huang, *Senior Member, IEEE*, and Tarik Taleb, *Senior Member, IEEE*

***Abstract*—Multi-access Edge Computing (MEC) provides services for resource-sensitive and delay-sensitive Internet of Things (IoT) applications by extending the capabilities of cloud computing to the edge of the networks. However, the high mobility of IoT devices (e.g., vehicles) and the limited resources of edge servers (ESs) affect the service continuity and access latency. Service migration and reasonable resource (re-)allocation consequently become needed to ensure quality of service (QoS). However, service migration results in additional latency. In addition, different mobile IoT users have different resource requirements and different resource allocation policies of target edge servers also determine whether service migration is necessary. Subsequently, how to jointly optimize service migration and resource allocation is a challenge that needs to be carefully addressed. To this end, this paper investigates the joint optimization problem of service migration and resource allocation (SMRA) in MEC environments to minimize the access delay of IoT users. It proposes a joint SMRA algorithm based on deep reinforcement learning (DRL), which takes into account the mobility of IoT users and decides whether to migrate services, where to migrate, and how to allocate resources through the long short time memory (LSTM) algorithm and the parameterized deep Q-network (PDQN) algorithm. Moreover, the PDQN algorithm effectively solves the discrete-continuous hybrid action space challenge in the SMRA problem. Finally, we conduct evaluation using a real-world dataset of Beijing cab trajectories to verify the effectiveness and superiority of our proposed SMRA solution.**

***Index Terms*—Internet of Things (IoT), service migration, resource allocation, deep reinforcement learning (DRL), long short term memory (LSTM), parametrized deep Q-Networks (PDQN), MEC, edge computing, and cloud.**

I. Introduction

Multi-access Edge Computing (MEC) is a new computing paradigm which solves the access delay problem of traditional cloud computing by deploying services at edge servers (ESs) closer to Internet of Things (IoT) users. As IoT devices are becoming increasingly intelligent, emerging IoT applications (i.e., autonomous driving, online gaming, ultra-high-definition video, etc.) place high requirements on service performance and resources. Nevertheless, ESs are normally equipped with limited heterogeneous resources and have limited coverage, while different IoT users have different resource requirements and different mobility patterns. For delay-sensitive and resource-intensive IoT applications, how to improve resource utilization and reduce access delay while maintaining service continuity is a huge challenge for edge system performance optimization.

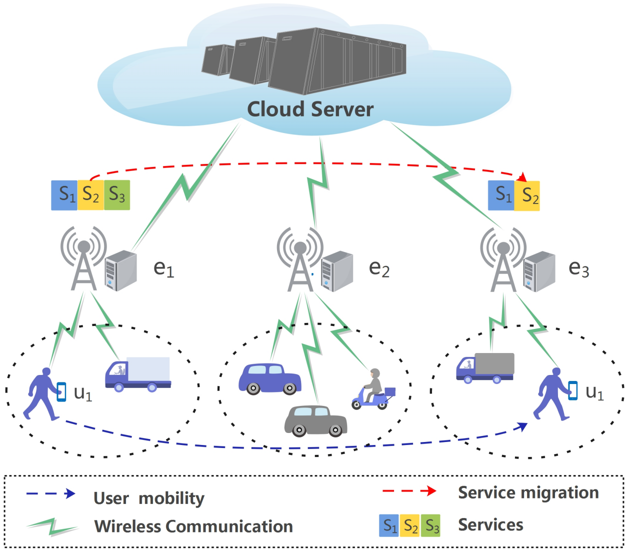


Fig. 1. A service migration scenario.

As shown in Fig. 1, we consider the following scenario whereby user u_1 connects to edge server e_1 at time t and requests service s_2 on edge server e_1. We assume that user u_1 will move to the vicinity of edge server e_3 at time $t+1$. On the one hand, if the user continues to connect to the source edge server e_1, it will result in a long communication delay between the mobile user u_1 and the source edge server e_1. On the other hand, we can perform service migration to migrate the service s_2 from the source edge server e_1 to the target edge server e_3,

This work is supported by National Natural Science Foundation of China (No. 61972414) and Beijing Nova Program (No. Z201100006820082). This research work was also conducted in ICTFICIAL OY and is partially supported by the European Union's Horizon Europe program for Research and Innovation through the aerOS project under Grant No. 101069732. It was also partially supported by the Business Finland 6Bridge 6Core project under Grant No. 8410/31/2022, the Research Council of Finland 6Genesis project under Grant No. 318927 and the Research Council of Finland IDEA-MILL project under Grant No. 352428. *(Corresponding author: Jiwei Huang.)*

Fangzheng Liu and Jiwei Huang are with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China. (e-mail: 2019310704@student.cup.edu.cn; huangjw@cup.edu.cn).

Hao Yu and Tarik Taleb are with the Center of Wireless Communications, University of Oulu, 90570 Oulu, Finland. (e-mail: Hao.Yu@oulu.fi; tarik.taleb@oulu.fi).

so as to shorten the communication distance and reduce the communication delay. Service migration is an effective means of ensuring service continuity, but it will additionally induce the migration delay. In addition, performing service migration will change the association between IoT users and ESs, and we need to consider the heterogeneity and limited resources of the target edge servers. The resource allocation policy of target edge servers determines the computation latency and communication latency of mobile users after service migration, which in turn affects the decision of whether to migrate or not. Therefore, we need to consider the service migration problem and resource allocation problem jointly and only perform service migration that is necessary and beneficial for reducing access delay.

To ensure service continuity and reduce access delay, future smart IoT systems should be able to perceive IoT users' mobile behavior and have the ability to execute service migration and resource allocation schemes in advance. Unfortunately, most previous research works have studied service migration and resource allocation as separate optimization problems, ignoring the fact that service migration decision and resource allocation scheme are mutually influential. Service migration is the factor that affects service continuity, while resource allocation strategy is another factor affecting the service migration decision. In addition, different IoT users may have different resource requirements and mobility patterns. Furthermore, the resources of ESs may be limited and heterogeneous. They dynamically change over time. These characteristics increase the difficulty of jointly optimizing service migration and resource allocation. To fill the above gaps, we investigate the SMRA problem by jointly considering these two factors to determine whether service migration is needed, where to move the services, and how to allocate resources, ultimately, to accommodate the mobility of IoT users, optimize IoT system resource utilization, minimize user access delay, effectively avoid service disruptions, and improve overall quality of service (QoS).

With the above background, we try to solve the joint optimization problem of service migration and resource allocation (SMRA), establish a joint optimization model by integrating resource constraints, and propose a deep reinforcement learning (DRL)-based SMRA algorithm which is able to predict mobile behavior of users and find an optimal service migration policy and resource allocation scheme to ensure the service continuity and minimize the access delay of users. Specifically, the main contributions of this paper are summarized as follows:

- Different from existing studies, this paper investigates the joint optimization problem of dynamic service migration and resource allocation, comprehensively considering the heterogeneity and limited resources of edge servers, as well as the different resource requirements and mobility of IoT users. The obtained optimal service migration policy and resource allocation scheme can ensure service continuity, minimize the access delay of users and improve resource utilization under the limited communication and computing resource constraints.
- Considering the mobility of IoT users and the dynamic changes of available resources of edge servers, this paper formulates the joint optimization SMRA problem as a Markov Decision Process (MDP) and proposes a DRL-based SMRA algorithm, which uses long short term memory (LSTM) predicting the mobility behavior of IoT users to improve migration accuracy and reduce the action space. Then, the parameterized deep Q-network (PDQN) algorithm is used to solve the continuous-discrete hybrid action space challenge faced in the joint optimization process without destroying the original structure of the action space.
- In order to verify the performance of our SMRA algorithm, we conducted experiments using the Beijing cab GPS track dataset. The experimental results indicate that our SMRA algorithm can obtain a smaller average task processing latency and has better performance compared to other baseline methods.

The rest of this paper is organized as follows. In Section II, some related work are reviewed and the remaining challenges of service migration and resource allocation are summarized. Section III presents the system model and the problem formulation. Section IV analyzes the service migration and resource allocation problem and proposes a DRL-based SMRA algorithm. Section V evaluates the performance of SMRA algorithm with real-world cab trajectory dataset. Section VI concludes this paper.

II. Related Work

Currently, there are few studies on the joint optimization of SMRA problems, so we mainly discuss the related work from the aspects of service migration and resource allocation, separately.

A. Service Migration

Service migration is an effective way to ensure service continuity and avoid service disruptions [1]. There are already some preliminary studies on service migration. The literature [2] presented using the Lyapunov optimization technique to address the challenges of service placement and service migration, which focused on the case where an edge server allocates an equal amount of computational resources to the users it serves. However, the fact is that resource requirements vary from user to user and change dynamically over time. The literature [3] presented a comprehensive analysis of user mobility, service types, and system environment characteristics, and gave an initial placement and migration scheme for the service. This study initially solve the service migration problem but ignores the impact of the resource allocation policy of the target edge server on the service migration results. Since different edge servers have different available resources, if the target edge server has fewer resources (e.g., computation resource and communication resource) relative to the source edge server and cannot allocate enough resources to IoT users, it will inevitably increase the computation and communication time of tasks after service migration, which in turn affects the result of whether to migrate or not.

The literature [4] proposed a dual-delay deep deterministic policy gradient (DDPG) algorithm to address long-term dynamic task allocation and service migration issues. Nevertheless, the mobility of users is unknown, and if the system performs the service migration operation after the users have deviated from the coverage of the source edge server will increase the access latency and degrade the user experience. Thus, it is necessary for the system to be able to predict the mobility of users and perform migration operations in advance so that seamless handover can be achieved and thus service continuity can be guaranteed. The literature [5] proposed to solve the service disruption problem caused by user mobility through base station switching, however, the authors neglected the change in computational resources due to base station switching. In contrast, our proposed scheme considers user mobility and the change in resources during service migration. This complicated issue has not been studied in service migration.

In addition, there are several studies focusing on service slice migration and resource management. The authors of [6] designed, modeled, and evaluated two DRL-based algorithms for allowing a fine-grained selection of system-based triggers regarding network slice movement patterns. This work aims to make their defined triggers more intelligent while keeping system resources stable, but does not ensure reduced network resource overhead. To address this issue, the authors in [7] further developed a network-aware agent capable of selecting accurate bandwidth values while ensuring fast and reliable service migration. However, system resources include not only bandwidth resources but also computational resources, and we need to manage system resources in an integrated manner to achieve optimized QoS.

B. Resource Allocation

In the IoT environment, the requirements of mobile users for resources and the remaining resources of edge servers are dynamically changing over time, and unreasonable resource allocation strategies will have an impact on service migration decisions. The literature [8] proposed to solve service migration and resource allocation by a relaxation and rounding-based approach with the aim of maximizing the weighted sum of the offload rate and migration cost. This paper mainly considered a static and stable edge computing scenario, and the authors assumed that the migration cost of each user is fixed, ignoring the dynamic nature of system state. However, in real IoT systems, the available resources of the edge servers and the location of IoT users change over time, and service migration and resource allocation policies need to be dynamically adjusted according to the real-time changes of system to ensure service performance and resource utilization. In addition, the authors did not consider the real computing power (e.g., CPU cycles) of edge server in order to simplify the model, but defined computing power as the number of users it can serve.

For multi-user MEC systems, dynamic service migration decisions and resource management are more complicated since it involves sharing of resources among multiple mobile users. However, existing studies on the dynamic optimisation of resource allocation for multiple users mainly focus on the optimization of offloading policies. The literature [9] considered stochastic vehicular traffic, dynamic computational requests and time-varying communication conditions, and proposed a DRL-based approach to find computational offloading and resource allocation strategies in vehicular edge computing networks to maximize the long-term utility of the network. The literature [10] focused on the changing communication and computational resources in a end-edge-cloud coordination network and proposed a DRL-based computational offloading and resource allocation algorithm to reduce energy consumption. To obtain discrete offloading decisions, the authors employed a rounding technique to refine the continuous values output by the DDPG algorithm, but this will inevitably affect the results. In contrast, we use the PDQN algorithm applicable to the continuous-discrete hybrid action space to solve the complex model of service migration and resource allocation established in this paper.

Different from the above studies, we explore an unknown direction, and investigate the joint optimization problem of dynamic service migration and resource allocation in a multi-user IoT environment, considering the heterogeneity of edge servers and limited resources, as well as the resource requirements and mobility of different IoT users, with the aim of ensuring service continuity, minimizing user access time, and maximizing resource utilization.

III. System Model and Problem Formulation

In this section, we first illustrate the system model. We then formulate the joint optimization SMRA problem as a total latency minimization problem under communication and computational resource constraints. The key notations, used in this paper, are summarized in Table I.

A. System Model

As shown in Fig. 1, we consider an IoT system which includes a set of mobile users $\mathcal{U} \triangleq \{1, 2, ..., U\}$, a set of base stations (BSs) integrating edge servers $\mathcal{E} \triangleq \{1, ..., E\}$ and a cloud server. We assume that in the case where a mobile user does not establish a connection with any other BS, it can only connect to the nearest BS and can only access the resources on the edge servers equipped by that BS [10], [11], [12]. We consider a scenario where each mobile user has a latency-sensitive computational task to process. We use $u \in \mathcal{U}$ and $e \in \mathcal{E}$ to represent the u-th mobile user and the e-th edge server, respectively. We use $D_{u,e}$ to denote the input data-size (in bits) for processing the task and $C_{u,e}$ represents the number of CPU cycles required to compute one-bit data of this task. Each BS is equipped with an edge server, which has limited computational and transmission resources. In contrast, the cloud server have powerful computing resources and acts as a controller of the IoT system in this paper which is responsible for service migration and resource allocation.

On each edge server, the underlying operating system (OS) is used as an underlying support environment to provide support for the task processing of different intelligent services

TABLE I
LIST OF KEY NOTATIONS.

Notation	Description
$r_{u,e}$	Data communication rate of user u to edge server e
$b_{u,e}$	Channel bandwidth
p_u	Transmission powers of user u
$g_{u,e}$	Channel gain between mobile user u and edge server e
σ^2	Gaussian noise power
$D_{u,e}(t)$	The task data size of user u
$I_{u,e}^{comm}(t)$	Transmission time of task data from user u to edge server e
$f_{u,e}(t)$	Computation resource allocated to user u by edge server e
$C_{u,e}(t)$	The number of CPU cycles for computing one-bit task data
$I_{u,e}^{comp}(t)$	Computation time for ES e to process the tasks of user u
$I_{u,e'}^{sus}(t)$	SI suspending time of user u
$D_{u,e'}^{mig}(t)$	The size of state context data of user u
$C_{u,e'}^{sus}(t)$	The processing intensity required to suspend the SI of user u
$\beta_{u,e'}(t)$	The proportion of computational resources allocated to checkpoint function
$f_{u,e'}^{SF}(t)$	The computing capacity of SF required by user u
$I_{u,e'}^{res}(t)$	SI resuming time of user u
$C_{u,e'}^{res}(t)$	The processing intensity required to resume the SI of user u
$I_{u,e'}^{syn}(t)$	State context data synchronization time
$r_{e,e'}(t)$	Data communication rate of edge server e to e'
$I_{u,e}^{mig}(t)$	The service migration time of user u
$I_{u,e'}^{comm}(t)$	Transmission time of task data from user u to edge server e'
$I_{u,e'}^{comp}(t)$	Computation time of ES e' to process tasks from user u
$f_{u,e'}(t)$	Computation resource allocated to user u by edge server e'
$I_{u,e'}^{nm}(t)$	Total task processing delay in non-migration case
$I_{u,e}^{m}(t)$	Total task processing delay in migration case
$I_u(t)$	Total task processing delay of user u at time t
$w_{u,e,e'}^{nm}(t)$	Binary service non-migration decision variable
$w_{u,e,e'}^{m}(t)$	Binary service migration decision variable
$\alpha_{u,e}(t)$	Link relationship between user u and edge server e at time t
$F_{u,e'}(t)$	Available CPU capacity of edge server e' at time t
$r_{e,e'}(t)$	Data communication rate of edge server e to e'
$B_{u,e'}(t)$	Available bandwidth of edge server e' at time t
$L_u^{pre}(t)$	The predicted location of user u at time $t+1$
$L_{u,e'}^{pre}(t)$	Location of the nearest edge server to the predicted user location.

and the operation of upper-layer software. On the OS, various edge service functions (SF) at the edge are deployed to meet the specialized task processing needs of different applications. At the same time, SFs store and manage stateless contextual information, such as public databases, executable code, etc., that is relevant only to a particular type of business but not to a particular user. Each SF can serve multiple users performing the same type of service. Once the user's computational task is assigned to an SF that has been activated on the edge server for processing, the SF first instantiates a Service Instance (SI) for that user, which is dedicated to processing the user's computation task and storing and managing some user-specific state context data, such as task real-time processing status, private data, etc.

As in [13], a mobile user sends requests to the nearest edge server by a wireless link connection. Therefore, based on the Shannon formula, the data communication rate (in bit/s) of mobile user u is given by

$$r_{u,e} = b_{u,e}\log_2\left(1 + \frac{p_u g_{u,e}}{\sigma^2}\right), \tag{1}$$

where $b_{u,e}$ is the channel bandwidth, $g_{u,e}$ represents the channel gain, which is related to the distance between mobile user u and edge server e, p_u denotes the transmission power of mobile user u, and σ^2 is the white Gaussian noise power.

In this paper, we will optimize and update the IoT system according to each time slot t. At the beginning of each time slot, the IoT system controller needs to make a choice based on migration delay and non-migration delay to determine whether to migrate or not, as well as the resource allocation policy for target edge server in case of migration.

1) Non-migration Model: If the service does not need to be migrated, the delay of mobile user u accessing the source edge server e consists mainly of communication delay and computation delay.

- *Communication delay:* The wireless transmission delay of tasks offloading from mobile user u to source edge server e can be written by

$$I_{u,e}^{comm}(t) = \frac{D_{u,e}(t)}{r_{u,e}(t)}. \tag{2}$$

  Since the size of the result data obtained after task processing is very small relative to the size of task data, we omit the transmission delay of the downlink in this paper.
- *Computing delay:* After receiving the tasks from mobile users, the source edge server will allocate the available computing resources to each task for computation. We denote the computational resources allocated by the source edge server e to the mobile user u by $f_{u,e}(t)$ (in CPU cycles/s). Thus, the computational delay of the source edge server e to process tasks from mobile user u can be written as

$$I_{u,e}^{comp}(t) = \frac{D_{u,e}(t)C_{u,e}(t)}{f_{u,e}(t)}. \tag{3}$$

To sum up, when a mobile user moves from the source edge server e to the vicinity of the target edge server e', if no service migration is performed and user u continues to access the source edge server e, the task processing delay can be expressed as follows:

$$I_{u,e}^{nm}(t) = I_{u,e}^{comm}(t) + I_{u,e}^{comp}(t). \tag{4}$$

2) Migration Model: In the case where services need to be migrated, the total delay from the start of executing service migration until the mobile user u completes access to the target edge server e' includes:

- The service migration delay from the source edge server e to the target edge server e';
- The communication delay between the mobile user u and the target edge server e';
- The computational delay of the target edge server e' to perform the tasks from the mobile user u.

(a) *Migration delay*: Service migration aims to synchronize user-dependent context data from source edge server to target edge server. As in [4], executing service migration includes the following steps: Suspending the user's SI; Preparing the user-dependent SF environment; Synchronizing the user's state context data; Resuming the user's SI and service processes.

The suspend and resume of user's SIs can be implemented using Checkpoint functional modules [14]. In this paper, we assume that this function module is embedded in each SF with constant processing power. Therefore, the SI suspend time $I^{sus}_{u,e'}(t)$ and resume time $I^{res}_{u,e'}(t)$ for user u at time t are respectively given by

$$I^{sus}_{u,e'}(t) = \frac{D^{mig}_{u,e'}(t)C^{sus}_{u,e'}(t)}{\beta_{u,e'}(t)f^{SF}_{u,e'}(t)}, \tag{5}$$

$$I^{res}_{u,e'}(t) = \frac{D^{mig}_{u,e'}(t)C^{res}_{u,e'}(t)}{\beta_{u,e'}(t)f^{SF}_{u,e'}(t)}, \tag{6}$$

where $D^{mig}_{u,e'}(t)$ (in bits) represents the size of the state context data of user u, $f^{SF}_{u,e'}(t)$ represents the computing capacity of the SF required by u (in CPU cycles/s), $\beta_{u,e'}(t)$ is the proportion of computational resources allocated by the SF to the embedded Checkpoint function, and $C^{sus}_{u,e'}(t)$ and $C^{res}_{u,e'}(t)$ (in CPU cycles/bit) represent the processing intensity required to suspend SI and resume SI of user u, respectively (i.e., the number of CPU cycles required by compute one-bit of state context data).

The synchronization time of the user state context data relies on the state context data size and the data transmission rate between source edge server and target edge server [2], which is written as

$$I^{syn}_{u,e'}(t) = \frac{D^{mig}_{u,e'}(t)}{r_{e,e'}(t)}, \tag{7}$$

where $r_{e,e'}$ denotes the data communication rate between the source edge server e and the target edge server e', similar to Eq. 1.

There are two cases for the time to prepare the SF environment required by user u. If the SF that the user u relies on is already activated on the target edge server, it is not necessary to reactivate a new one, so the SF environment preparation time can be ignored. Instead, the target edge server need to activate a new SF. In this paper, to simplify the processing, we only consider the case where the SF preparation time is constant for each application, and the SF environment can be activated and adjusted with a short control frame at the beginning of the state context information synchronization.

To sum up, the service migration delay from the source edge server e to the target edge server e' is given by

$$I^{mig}_{u,e'}(t) = I^{sus}_{u,e'}(t) + I^{res}_{u,e'}(t) + I^{syn}_{u,e'}(t). \tag{8}$$

(b) *Communication delay*: When the service migration is complete, mobile user u can access the target edge server e'. The transmission delay of the tasks offloading from user u to the target edge server e' can be expressed as follows:

$$I^{comm}_{u,e'}(t) = \frac{D_{u,e}(t)}{r_{u,e'}(t)}, \tag{9}$$

where $r_{u,e'}$ denotes the data communication rate between the user u and the target edge server e', similar to Eq. 1.

(c) *Computing delay*: We use $f_{u,e'}$ to represent the computational resources allocated by the target edge server e' to the mobile user u, then the computational delay of the target edge server e' to process the tasks from the mobile user u can be written as follows:

$$I^{comp}_{u,e'}(t) = \frac{D_{u,e}(t)C_{u,e}(t)}{f_{u,e'}(t)}. \tag{10}$$

Thus, when a user moves from the edge server e to the vicinity of the edge server e', if service migration is performed and the service is migrated from the source edge server e to the target edge server e', the task processing delay can be expressed as follows:

$$I^{m}_{u,e'}(t) = I^{comm}_{u,e'}(t) + I^{comp}_{u,e'}(t) + I^{mig}_{u,e'}(t). \tag{11}$$

B. Problem Formulation

In this paper, we define the target edge server as the closest edge server to the predicted user location. According to Eq. 4 and Eq. 11, if the user continues to access the source edge server after the move, the task processing latency is $I^{nm}_{u,e}(t)$. If the server is migrated to the target edge server and the task is processed through the target edge server, the task processing latency is $I^{m}_{u,e'}(t)$. We need to decide whether users continue to access the source edge server or perform a service migration and access the target edge server based on the computing and communication resource allocation policy of the target edge server. Therefore. we set $I_u(t)$ to be the total task processing delay of user u at time t, and we have

$$I_u(t) = w^{nm}_{u,e,e'}(t)I^{nm}_{u,e}(t) + w^{m}_{u,e,e'}(t)I^{m}_{u,e'}(t), \tag{12}$$

where $w^{nm}_{u,e,e'}(t)$ and $w^{m}_{u,e,e'}(t)$ are the binary service migration decision variable, when $w^{nm}_{u,e,e'}(t) = 1$ and $w^{m}_{u,e,e'}(t) = 0$, the service in the source edge server e are not migrated to the target edge server e', otherwise, $w^{nm}_{u,e,e'}(t) = 0$ and $w^{m}_{u,e,e'}(t) = 1$. We aim to find the optimal service migration policy and resource allocation scheme that minimizes the total task processing delay, and considering the target edge server resource constraint, the optimization problem can be formulated as

$$\begin{aligned}
\mathcal{P}: \underset{\substack{w^{nm}_{u,e,e'}(t), w^{m}_{u,e,e'}(t),\\ f_{u,e'}(t), b_{u,e'}(t)}}{minimize} \quad & \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T}\sum_{u\in\mathcal{U}} I_u(t) && \text{(13a)}\\
s.t. \quad & \sum_{u\in\mathcal{U}} \alpha_{u,e'}(t)\cdot f_{u,e'}(t) \le F_{u,e'}(t), && \text{(13b)}\\
& \sum_{u\in\mathcal{U}} \alpha_{u,e'}(t)\cdot b_{u,e'}(t) \le B_{u,e'}(t), && \text{(13c)}\\
& w^{nm}_{u,e,e'}(t) + w^{m}_{u,e,e'}(t) = 1, && \text{(13d)}\\
& w^{nm}_{u,e,e'}(t), w^{m}_{u,e,e'}(t) \in \{0,1\}, && \text{(13e)}\\
& \sum_{e'\in\mathcal{E}} \alpha_{u,e'}(t) = 1, && \text{(13f)}\\
& \alpha_{u,e'}(t) \in \{0,1\}, && \text{(13g)}\\
& \forall u \in \mathcal{U}, \forall e' \in \mathcal{E}, && \text{(13h)}
\end{aligned}$$

where $\alpha_{u,e'}(t)$ represents the link relationship between user u and target edge server e' at time t, and we have

$$\alpha_{u,e'}(t) = \begin{cases} 1, & \text{if user } u \text{ is served by edge server } e', \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Constraint (13b) denotes that the CPU requirements of all users connected to the edge server e' cannot exceed the available CPU capacity of the edge server e', where $F_{u,e'}(t)$ denotes the available computing capacity of e' at time t.

Constraint (13c) represents that the bandwidth allocated to all users connected to the edge server e' for offloading cannot exceed the available bandwidth of the edge server e', where $B_{u,e'}(t)$ denotes the available bandwidth of e' at time t. In particular, after the user u-dependent services are migrated, the corresponding computational and bandwidth resources on the source edge server need to be released, meanwhile, the corresponding resources on the target edge server are occupied, so that $F_{u,e'}(t)$ and $B_{u,e'}(t)$ in constraints 13(b) and 13(c) are dynamically changing over time. Constraint (13d) and (13e) indicate that for migration decisions within each decision cycle t, there are just two options for non-migration and migration. Constraint (13f) and (13g) indicate that each user can be served by only one edge server in each decision cycle t.

Since the service migration decision variables $w^{nm}_{u,e,e'}(t)$ and $w^{m}_{u,e,e'}(t)$ are binary variables, the problem $\mathcal{P}$ is non-convex. In addition, we consider the user mobility, edge server resource heterogeneity and limited, which are difficult to solve with traditional optimization algorithms. In contrast, deep reinforcement learning can update the service migration and resource allocation policies by interacting with the environment in real time, which makes it possible to adapt to the dynamic changes of service requirements in emerging IoT systems. Therefore, we present a DRL-based algorithm to solve the problem $\mathcal{P}$.

IV. DRL-Based SMRA approach

In this section, we first reformulate the delay minimization problem as MDP and define the state, action, and reward function. Then we propose a DRL-based SMRA algorithm.

A. DRL-Based Framework

The problem $\mathcal{P}$ is described as MDP which mainly consists of three parts: state, action and reward.

(1) *State*: At the beginning of each time slot, the controller observes system states, which includes the predicted user location, the location of the edge server closest to the predicted location, resources allocated to users by the source edge server, and the available resources of the target edge server. In particular, according to the problem $\mathcal{P}$, when the computational and bandwidth resources allocated to the user by the source edge server e are known, the resources available at the source edge server e do not have an impact on the total task processing latency. Thus, the computational resources and bandwidth of source edge server e need not be considered in the state space of reinforcement learning. The system state at time t can be defined as follows:

$$s_t \triangleq \{L^{pre}_u(t), L^{pre}_{u,e'}(t), f_{u,e}(t), b_{u,e}(t), F_{u,e'}(t), B_{u,e'}(t) | u \in \mathcal{U}\}, \tag{15}$$

where

- $L^{pre}_u(t)$: represents the predicted location of the user at the next time t, where to determine the location of the target edge server capable of service migration, we predict the user's location $L^{pre}_u(t)$ at time $t+1$ by the LSTM [18] algorithm, and the edge server closest to the $L^{pre}_u(t)$ is the target edge server for service migration. The prediction process is described in detail in the next *subsection B*.
- $L^{pre}_{u,e'}(t)$: We use e' represents the target edge server which is the nearest edge server to $L^{pre}_u(t)$, and $L^{pre}_{u,e'}(t)$ is the location of the target edge server e'. There are many edge servers in the vicinity of the user, and we take the edge server closest to the predicted user location as the target edge server and add it to the current state as a known value. Firstly, the size of the state and action space can be greatly reduced, avoiding the exponential computational complexity due to global search. Second, in general, if the user device has not yet established a connection with another base station, it will automatically connect to the nearest base station [10], [11], [12]. If the user device wants to access resources on other edge servers, it needs to perform base station handover first. According to the X2 handover principle in 5G environments, the user device needs to continuously send measurement reports to the currently connected base station, which decides whether handover is required based on these measurement reports, and if handover is required, the base station will notify the core network to decide which target base station to handover to. However, too much switching increases the complexity of scheduling between edge servers, core network, base stations, and user devices, consumes a lot of energy, and increases latency [16], [17]. Therefore, considering the computational complexity and handoff overhead, this paper defines the edge server closest to the predicted user location as the target edge server.
- $f_{u,e}(t)$: denotes the computing resources allocated to users by edge server e at time t.
- $b_{u,e}(t)$: denotes the bandwidth resources allocated to users by edge server e at time t.
- $F_{u,e'}(t)$: denotes the remaining available computing resources of the target edge server e' at time t.
- $B_{u,e'}(t)$: denotes the remaining available bandwidth of the target edge server e' at time t.

(2) *Action*: In state s_t, the system controller needs to decide whether to perform service migration and determine computation and communication resources that the target edge server allocates to users. Therefore, the action of mobile user u at time t can be defined as follows:

$$a_t \triangleq \{w^{nm}_{u,e,e'}(t), w^{m}_{u,e,e'}(t), f_{u,e'}(t), b_{u,e'}(t) | u \in \mathcal{U}\}, \tag{16}$$

where

- $w^{nm}_{u,e,e'}(t), w^{m}_{u,e,e'}(t)$: represents service migration decisions of user u at time t.
- $f_{u,e'}(t)$: represents the computing resources allocated to user u by the target edge server e' at time t.
- $b_{u,e'}(t)$: represents the bandwidth resources allocated to user u by the target edge server e' at time t.

(3) *Reward*: In state s_t, the system executing the action a_t will obtain a reward value. For the problem $\mathcal{P}$, our optimization objective is to minimize the task processing delay.

Therefore, the reward function can be expressed by

$$r_t = -\mathbb{E}[\sum_{u\in\mathcal{U}} I_u(t)]. \tag{17}$$

Since $I_u(t)$ is determined by weighting the task processing time $I_{u,e'}^{m}(t)$ in the migration case and the task processing time $I_{u,e}^{nm}(t)$ in the non-migration case, as in Eq. 4, Eq. 11 and Eq. 12. If the target server has insufficient or even no available remaining computational and bandwidth resources, then it cannot allocate sufficient resources for users to perform task processing, resulting in a longer task processing time for the target edge server than for the source edge server, i.e., the reward value obtained from the migration action is lower than that obtained from the non-migration action, and our migration decision variables $w_{u,e,e'}^{nm}(t)$ and $w_{u,e,e'}^{m}(t)$ are 0 or 1 decisions and our goal is to find a strategy that yields a high reward value, so we would choose not to migrate the service and continue to access the source edge server. Otherwise, we will choose to migrate the service.

In reinforcement learning, the policy is defined as a mapping from state to actions $\pi : \mathcal{S} \to \mathcal{A}$, where $\mathcal{S}$ represents the system states, and $\mathcal{A}$ represents the system actions. The objective is to find the optimal policy π^* that maximizes long-run expected rewards through interaction with the environment.

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t}^{T} \gamma^t r\left(s_t, a_t\right)\right], \tag{18}$$

where T represents the episode termination time step and $\gamma \in [0, 1]$ is the discount factor. To solve π^*, Q-learning algorithm [15] uses Q-values to represent state-action pairs, which are stored in a Q-matrix that is updated according to the rewards of each policy. However, the huge size of $\mathcal{S}$ and $\mathcal{A}$ will result in a huge size of Q-matrix. To the above issues, DRL combines deep neural network (DNN) and RL by inputting the state-action pairs into a DNN to output Q-values. The Bellman equation of action-value function (i.e. Q-function) is given by

$$Q(s_t, a_t) = \mathop{\mathbb{E}}_{r_t, s_{t+1}}\left[r_t + \gamma \max_{a_{t+1}\in\mathcal{A}} Q\left(s_{t+1}, a_{t+1}\right)\right]. \tag{19}$$

In the classical DRL algorithm, in order to address the shortcomings in solving the Q-function with DNN, the deep Q-network (DQN) algorithm uses "empirical replay" techniques and asynchronous updating of the Q-network. However, DQN is only applicable to handle discrete action spaces, for the continuous action space, DQN cannot efficiently calculate the optimal action $a^*(s) = \max_a Q^*(s, a)$, and it is difficult to obtain greedy strategies. To overcome this problem, the DDPG algorithm is proposed, which introduces a deterministic actor network to output approximate maximized Q-value and updates the actor network by gradient ascent. However, the action space shown in Eq. 16 is a complex discrete-continuous hybrid action space, and just using DQN or DDPG is not sufficient to solve this challenge. For this reason, we use PDQN to solve the hybrid action space challenge, which combines the features of DQN and DDPG.

B. SMRA Algorithm based on LSTM and PDQN

In this paper, we need to find a optimal service migration and resource allocation policy based on the current state of the IoT system, including determining where to migrate, whether to migrate and how to allocate resources. To determine the service migration location, we first use the LSTM [18] algorithm to predict the user's future location $L_u^{pre}(t)$ from historical user location data and add the edge server closest to the user's next location as the target edge server for service migration to the current state. Then, we use the PDQN algorithm to determine whether to perform service migration and how to allocate resources. DRL is a long-term iterative optimisation process, while LSTM is suitable for handling time series data. We use the location predicted by LSTM as one of the states of DRL, allowing the intelligent system to learn both the location prediction model and the decision model. In addition, the SMRA algorithm based on LSTM and PDQN can be embedded directly into the intelligent system without relying on external mapping applications, making it easier to deploy and maintain. We hope that the SMRA algorithm will improve the intelligence of the system, allowing the intelligent system to make optimal service migration and resource allocation plans before the user moves, ensuring service continuity and resource utilisation.

The SMRA algorithm gives actions for service migration and resource allocation based on the real-time state of the system and calculates the reward values after the actions are executed, after which the system enters a new state. Based on these empirical data of states, actions, action reward and next state values, the DRL model is trained and updated. Finally, the trained DRL model is used to implement intelligent management of the IoT system. We separate the action space of Eq. 16 and use k to represent the discrete actions $\{w_{u,e,e'}^{nm}(t), w_{u,e,e'}^{m}(t)\}$ and x_k ($x_k \in \mathcal{X}_k$, $\mathcal{X}_k$ to represent the set of continuous actions) to represent the continuous actions $\{f_{u,e'}(t), b_{u,e'}(t)\}$. More specifically, we can denote the discrete-continuous hybrid action space $\mathcal{A}$ as

$$\mathcal{A} = \left\{(k, x_k) \mid x_k \in \mathcal{X}_k, \text{ for all } k \in K\right\}, \tag{20}$$

where K is the set of discrete actions. We represent the action value function $Q(s, a)$ by $Q(s, k, x_k)$, in which $s \in \mathcal{S}$, $a \in \mathcal{A}$, $k \in K$, and $x_k \in \mathcal{X}_k$. We use k_t to represent the discrete action chosen at time t and x_{k_t} to represent the corresponding continuous parameter. Then the Bellman equation can be written by

$$Q(s_t, k_t, x_{k_t}) = \mathop{\mathbb{E}}_{r_t, s_{t+1}}\left[r_t + \gamma \max_{k\in K} \sup_{x_k\in\mathcal{X}_k} Q\left(s_{t+1}, k, x_k\right)\right]. \tag{21}$$

For the above Bellman equation, we need to compute $x_k^* = \operatorname{argsup}_{x_k\in\mathcal{X}_k} Q\left(s_{t+1}, k, x_k\right)$ for each $k \in K$, and choose the largest $Q\left(s_{t+1}, k, x_k^*\right)$. When function Q is fixed, $x_k^* = \operatorname{argsup}_{x_k\in\mathcal{X}_k} Q\left(s_{t+1}, k, x_k\right)$ can be regarded as a function $x_k^Q : \mathcal{S} \to \mathcal{X}_k$ for $\forall s \in \mathcal{S}$ and $\forall k \in K$ that maps the state space to the continuous domain of action parameters.

Then the Bellman equation of Eq. 21 can be rewritten by

$$Q\left(s_t, k_t, x_{k_t}\right) = \underset{r_t, s_{t+1}}{\mathbb{E}}\left[r_t + \gamma \max_{k \in [K]} Q\left(s_{t+1}, k, x_k^Q\left(s_{t+1}\right)\right)\right]. \tag{22}$$

Similar to DQN, we use DNN $Q\left(s, k, x_k; \omega\right)$ to approximate $Q\left(s, k, x_k\right)$, in which w represents the network parameters. For such $Q(s, k, x_k; w)$, similar to DDPG, we approximate $x_k^Q(s)$ by a deterministic policy network $x_k(\cdot; \theta): \mathcal{S} \rightarrow \mathcal{X}_k$, in which θ represents the parameter of deterministic policy network. It means that when w is fixed, the objective of PDQN is to find the corresponding parameter θ such that

$$Q\left(s, k, x_k(s; \theta); \omega\right) \approx \sup_{x_k \in \mathcal{X}_k} Q\left(s, k, x_k; \omega\right), \forall k \in K. \tag{23}$$

Then we use the following least mean squares loss function for w similar to DQN.

$$\ell_t^Q(\omega) = \frac{1}{2}\left[Q\left(s_t, k_t, x_{k_t}; \omega\right) - y_t\right]^2, \tag{24}$$

where

$$y_t = \begin{cases} r_t, \text{if } s_{t+1} \text{ is the terminal state,} \\ r_t + max_{k \in K} \gamma Q\left(s_{t+1}, k, x_t(s_{t+1}; \theta_t); w_t\right), \text{else.} \end{cases} \tag{25}$$

is evaluated by the target network and s_{t+1} denotes the next state following the adoption of the hybrid action (k, x_k). Furthermore, as our objective is to find θ which maximize $Q\left(s, k, x_k(s; \theta); \omega\right)$ while w is fixed, we perform the following loss function for θ

$$\ell_t^{\Theta}(\theta) = -\sum_{k=1}^{K} Q\left(s_t, k, x_t(s_t; \theta); w_t\right), \tag{26}$$

In addition, the value network and deterministic policy network weights w_t and θ_t are updates via gradient descent according to

$$\omega_{t+1} \leftarrow \omega_t - \alpha_t \nabla_{\omega} \ell_t^Q\left(\omega_t\right), \tag{27}$$

$$\theta_{t+1} \leftarrow \theta_t - \beta_t \nabla_{\theta} \ell_t^{\Theta}\left(\theta_t\right), \tag{28}$$

where α_t and β_t are the learning rate.

Fig. 2 shows the service migration and resource allocation framework. First, x-network takes the current state s_t which is observed from IoT system as input and generates x_k for all actions $k \in K$, in which we explore the continuous action part using a noise process similar to DDPG. After that, the Q-network takes the state s_t and the parameters x_k generated by the x-network as input and outputs the Q values of all actions k. The ϵ-greedy strategy determines the desired actions a_t. Then, the action a_t is executed to obtain the reward value r_t and the next state s_{t+1} of system. Meanwhile, the possible location of user at time $t+1$ are predicted by the LSTM algorithm and added to the state s_t. Finally, the empirical data of system state s_t, action a_t, action reward r_t and next time state s_{t+1} are stored to the replay buffer $\mathcal{R}$ for subsequent PDQN training. In this process, we use the "empirical replay" [20] technique to store the empirical data (s_t, a_t, s_{t+1}, r_t) into replay buffer and update the network by retrieving continuously small batches of samples from the pool, which weaken the interference of data correlation and thus improves the learning efficiency.

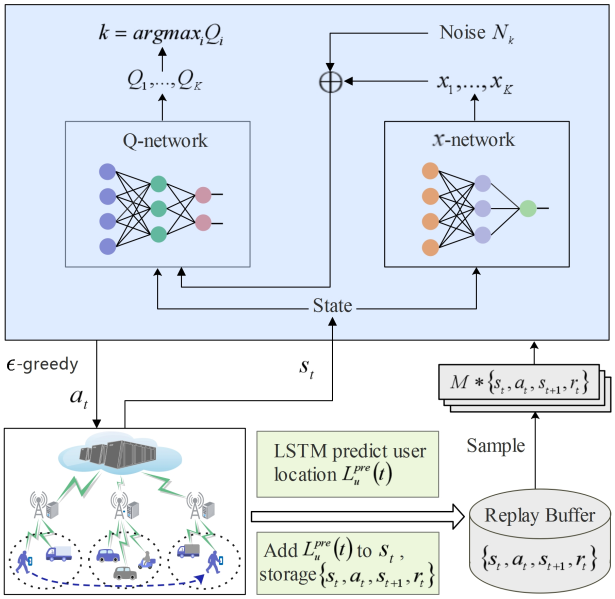

$k = argmax_i Q_i$
$Q_1, \ldots, Q_K$
Noise N_k
$x_1, \ldots, x_K$
Q-network
x-network
State
ϵ-greedy
a_t
s_t
$M * \{s_t, a_t, s_{t+1}, r_t\}$
Sample
LSTM predict user location $L_u^{pre}(t)$
Replay Buffer
$\{s_t, a_t, s_{t+1}, r_t\}$
Add $L_u^{pre}(t)$ to s_t, storage $\{s_t, a_t, s_{t+1}, r_t\}$

Fig. 2. Service migration and resource allocation framework.

The detailed steps of the SMRA algorithm based on LSTM and PDQN are shown in Algorithm 1, where line 1 initializes the relevant parameters. In line 3-6, the x-network generates x_k for all $k \in K$ with the current state s_t as input. This part uses a noise process $\mathcal{N}$ similar to the DDPG algorithm to explore continuous actions. Line 7 explores the action based on ϵ-greedy. Line 8 execute action a_t and observe reward r_t and new state s_{t+1}. Line 9 and line 10 use LSTM algorithm to predict the future location of user and concatenate this information into state s_t. Line 11 stores the obtained empirical data into the replay buffer $\mathcal{R}$. Line 13 sample a mini-batch transitions randomly from replay buffer $\mathcal{R}$ for training. Lines 14 and 15 describe the training process of the Q-network and the x-network, which is similar to the DDPG algorithm.

The complexity of Algorithm 1 is mainly determined by the calculation of neural network parameters and LSTM algorithm. Assuming that Q-network DNN contains H fully connected layers and and x-network DNN contains J fully connected layers. For each iteration of training, the time complexity for a fully connected layer is $w_{full} = \mathcal{O}(\sum_{h=0}^{H} n_{Q,h} \cdot n_{Q,h+1} + \sum_{j=0}^{J} n_{x,j} \cdot n_{x,j+1})$, where $n_{Q,h}$ and $n_{x,j}$ mean the unit number in the h-th Q-network DNN layer and the j-th x-network DNN layer, $n_{Q,0}$ and $n_{x,0}$ equal the input size [10], [21], [22], [23]. In addition, the time complexity of the LSTM algorithm for forward and backward propagation is related to the length of the sequence and the number of model layers. Therefore, the time complexity of LSTM is $w_{lstm} = \mathcal{O}(L \cdot p \cdot (4g^2 + 4g))$, where L is the sequence length, p is the input dimension and g is the number of hidden units. Let z be the total number of training iterations, the total time complexity of Algorithm 1 is $\mathcal{O}(z \cdot w_{full} + z \cdot w_{lstm})$.

Algorithm 1 is able to adapt to the dynamic changes of the complex IoT environment, and is adaptive and learning

Algorithm 1 SMRA Algorithm

```
Input: Stepsizes ${\alpha_t, \beta_t}_{t \geq 0}$, exploration parameter $\epsilon$, a probability distribution $\xi$, replay buffer size $B$, minibatch size $M$, user location $L_u$.
Output: Decisions for service migration $w^{nm}_{e,e'}(t), w^{m}_{e,e'}(t)$, decisions for resource allocation $f_{u,e'}(t), b_{u,e'}(t)$.
Initialize: replay buffer $\mathcal{R}$, network weights $w_1$ and $\theta_1$.
for each episode do
    Observe current state $s_t$
    Initialize a random process $\mathcal{N}$ for action exploration
    for $t = 1, ..., T$ do
        Compute action parameter $x_k \leftarrow x_k(s_t, \theta_t) + \mathcal{N}_k$
        Select action $a_t = (k_t, x_{k_t})$ based on $\epsilon$-greedy policy:
        $a_t = \begin{cases} \text{a sample from distribution } \xi, & \epsilon \\ k_t = argmax_{k \in K} Q(s_t, k, x_k; w_t), & 1-\epsilon \end{cases}$
        Execute action $a_t$, observe reward $r_t$ and new state $s_{t+1}$
        Compute the user's position for the previous $\tau$ times, obtaining a location vector $L_u(t-\tau), \ldots, L_u(t)$.
        Predict future locations by LSTM algorithm based on $L_u(t-\tau), \ldots, L_u(t)$ and add $L^{pre}_u(t)$ in $s_t$
        Store $(s_t, a_t, s_{t+1}, r_t)$ in replay buffer $\mathcal{R}$
        Update current state $s_t \leftarrow s_{t+1}$
        Sample $M$ transitions $(s_i, a_i, s_{i+1}, r_i)$ randomly from replay buffer $\mathcal{R}$
        Train Q-network:
        compute $y_t$ by (25) and loss by (24)
        perform stochastic gradient descent step by (27)
        Train x-network:
        compute loss by (26)
        perform stochastic gradient descent step by (28)
    end for
end for
```

TABLE II
SIMULATION PARAMETERS.

Parameter	Values
Available channel bandwidth, B	10MHz [28]
Transmission powers of user u, $p_{u,e}$	24dBm [28]
Gaussian noise power, σ	-100dBm/Hz [26]
Available CPU capacity of edge server e', $F_{e'}$	150×10^8 cycles/s [28]
The task data size, $D_{u,e}$	[200, 300]KB [4]
The number of CPU cycles required to compute one-bit task data, $C_{u,e}$	[300, 500]cycles/bit [4]
Computation resource allocated to user u by edge server e, $f_{u,e}$	$[f^{min}_{u,e}, f^{max}_{u,e}]$
Minimum value of computation resource allocated to user by edge server, $f^{min}_{u,e}$	10×10^7 cycles/s
Maximum value of computation resource allocated to user by edge server, $f^{max}_{u,e}$	80×10^7 cycles/s
Bandwidth resource allocated to user u by edge server e, $b_{u,e}$	$[b^{min}_{u,e}, b^{max}_{u,e}]$
Minimum value of bandwidth resource allocated to user by edge server, $b^{min}_{u,e}$	0.3 MHz
Maximum value of bandwidth resource allocated to user by edge server, $b^{max}_{u,e}$	5 MHz
The size of state context data of user u, $D^{mig}_{u,e'}$	[1, 8]MByte [4]
The proportion of computational resources allocated to the checkpoint function, $\beta_{u,e'}$	$[20, 30]\%$ [4]
Processing intensity required to suspend SI, $C^{sus}_{u,e'}$	[228, 538]cycles/bit [4]
Processing intensity required to resume SI, $C^{res}_{u,e'}$	[350, 450]cycles/bit [4]
Computing capacity of SF required by user, $f^{SF}_{u,e'}$	$[200, 300] \times 10^6$Hz [4]
Discount Factor, γ	0.9 [26]
Replay size, $\mathcal{R}$	1000 [26]
Mini-batch size M	25 [26]
The greedy policy parameter, ϵ	0.05
Averaging rate, υ	0.01 [27]

through interactive feedback with the IoT environment to learn the best service migration and resource allocation strategies. In addition, Algorithm 1 has location prediction capabilities to ensure that service migration and resource allocation decisions are made before the user moves to the next edge server coverage area, thus addressing service continuity and resource utilisation issues.

V. SIMULATION RESULTS AND DISCUSSION

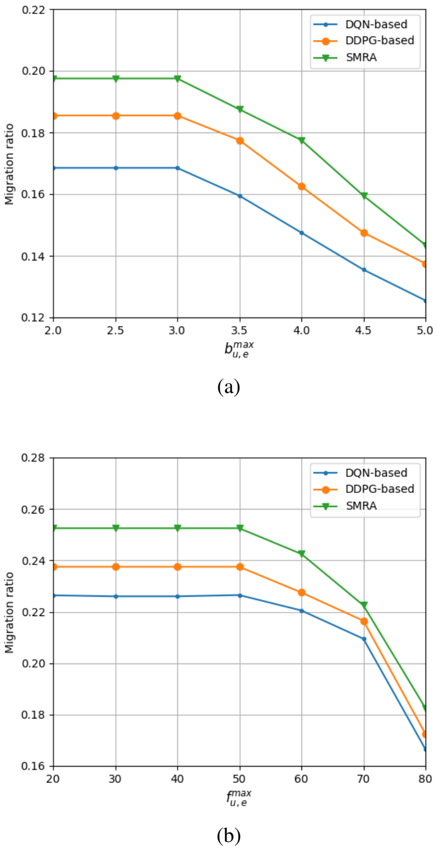
A. Experimental Settings

We validate our SMRA scheme using a real-world dataset containing the GPS trajectories of 10,357 cabs in Beijing [24]. The trajectory data of each cab includes longitude, latitude and time, and its longitude and latitude change dynamically with time. The dataset was released by Microsoft Research Asia, and the release date was from February 2 to February 8, 2008. In our experiments, we treat each mobile cab in the dataset as a mobile user.

In the default case, we randomly select the trajectory data of 70 mobile users in a certain area from the dataset and deploy 60 edge servers, setting the distance between the edge servers to 5km to ensure that the deployed edge servers can cover the selected 70 users. The channel gain is set to $(d_{u,e})^{-\alpha}$, where $d_{u,e}$ represents the distance between user u and edge server e and $\alpha = 3$ is the pathloss factor [25]. We use Euclidean distance formula to calculate the distance. The default experiment parameters settings are as table II. With the experimental parameters taking default values, we set the execution period of the cloud server (i.e. service migration and resource allocation execution cycle) to every 10 minutes.

To verify the superiority of SMRA algorithm presented in this paper, we compare it by the following five methods:

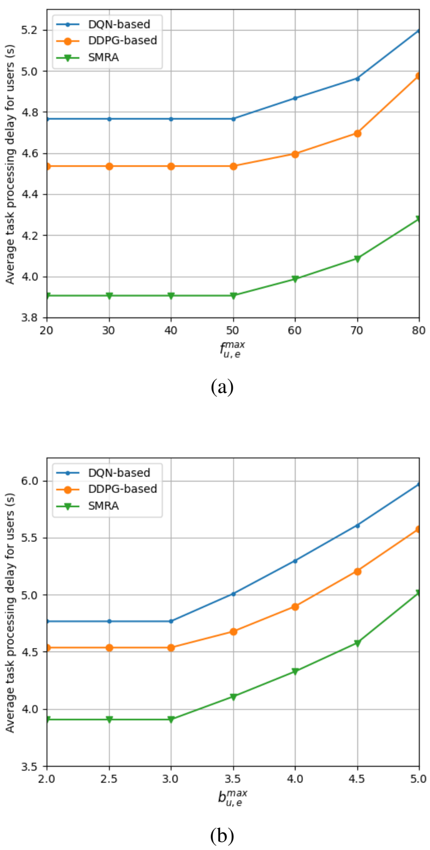
- ***DDPG-based scheme***: We selecte the state-of-the-art method from literature [10] for our comparison experiments. This literature used a DDPG-based algorithm to solve the computational offloading and resource allocation problem. In order to adapt DDPG to a hybrid action space with discrete and continuous actions, the literature uses a rounding technique to refine the binary offloading decision.
- ***DQN-based scheme***: We discretize the continuous resource allocation values in the action space of this paper by approximating each x_t with a discrete subset, and then solve them using the DQN-based algorithm.
- ***Random migrate scheme (RMS)***: When the edge server closest to the user at time t and $t+1$ is the same edge server, we do not perform migration, otherwise, the service randomly chooses whether to migrated to the edge



(a)

(b)

Fig. 3. Impact of different maximum user resource requirement ranges on service migration ratio. (a) Maximum value of bandwidth resource allocated to user by edge server. (b) Maximum value of computation resource allocated to user by edge server.



(a)

(b)

Fig. 4. Impact of different maximum user resource requirement ranges on average user task processing delay. (a) Maximum value of bandwidth resource allocated to user by edge server. (b) Maximum value of computation resource allocated to user by edge server.

server closest to the user.

- ***Always migrate scheme (AMS)***: This migration scheme migrates the services required by the user to the edge server closest to the user as the user moves.
- ***Never migrate scheme (NMS)***: The service will not be migrated no matter how the user moves.

B. Parameter Analysis

In practice, different users have different resource requirements, so we set the computing resources and bandwidth allocated to users by the edge server varying within a certain range, and we adjust the size of the maximum value of the range of computing resources and bandwidth and conduct two sets of experiments to compare the proportion of users migrating and the average task processing latency, respectively. Fig. 3 shows the variation of the proportion of user migration for different algorithms when the upper limits of the random range of computational resources and bandwidth allocated to users are increased. We can see from the figure that when the allocated computational resources and bandwidth to users are relatively small, the resource demand of each user can be satisfied, so the migration rate remains constant as the resource demand of users increases. When the resources allocated to users reach a certain value, the edge server will not be able to satisfy the resource demand of each user, which will result in fewer users that can be served by the edge server and a lower migration ratio.

Fig. 4 shows the variation in the average task processing delay of users when the upper limit of the random range of allocated computing resources and bandwidth increases. We can see from the figure that when the allocated computing resources and bandwidth to users are relatively small, the resource demand of each user can be satisfied and the average task processing latency of each user remains constant. When the resources allocated to users reach a certain value, the edge server will not be able to satisfy the resource demand for each user, This will result in fewer users to be served by edge servers and lower migration rates.

Fig. 5 illustrates the variation of the average task processing delay of users as the number of iterations increases for discount factors of 0.1, 0.2, 0.5, 0.9, respectively, from which it can be seen that the average delay decreases gradually with the number of iterations for different discount factors and finally converges to an optimal solution.

C. Comparison Experiments

Fig. 6 compares the performance of the six algorithms by adjusting the number of users. From the figure, we can

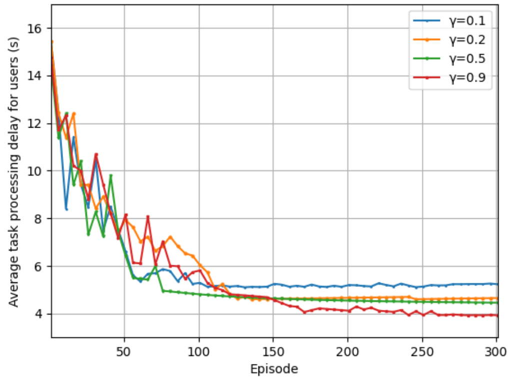


Fig. 5. Impact of discount factors on convergence.

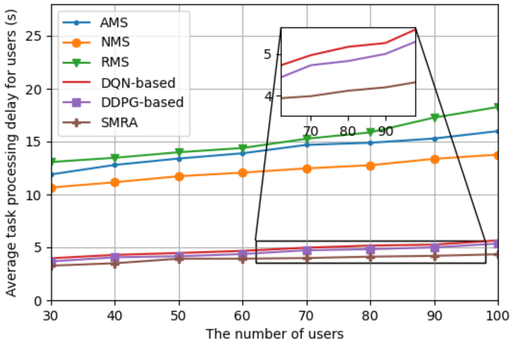


Fig. 6. Impact of the number of users on average task processing delay.

see that as the number of users increases, the average task processing latency also increases, but the DRL-based scheme grows relatively slowly, and using the DRL-based scheme to address the service migration and resource allocation issues is significantly better than the other three schemes. More specifically, compared with always migration scheme, the DRL-based scheme can choose the appropriate policy for service migration according to the current system state, thus avoiding frequent service migration, and compared with never migration scheme, the DRL-based scheme avoids excessive latency between edge servers and users due to long communication distance. In addition, among the three DRL-based schemes, the SMRA scheme proposed in this paper can achieve lower average task processing latency compared to DQN-based and DDPG-based schemes due to better preservation of the action space structure.

Fig. 7 illustrates the performance comparison of six schemes as the number of edge servers increases. We can see from the figure as the number of edge servers increases, the resources of edge servers also increase and therefore the average task processing latency becomes smaller. The deep reinforcement learning based methods have a smaller average task processing latency compared to the other three solutions because of the trade-off between migration latency and non-migration latency. The SMRA scheme proposed in this paper is consistently optimal in different performance comparisons.

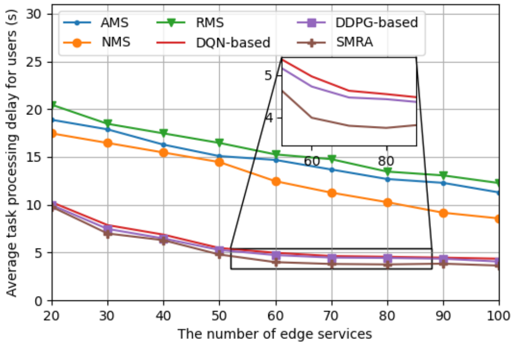


Fig. 7. Impact of the number of edge servers on average task processing delay.

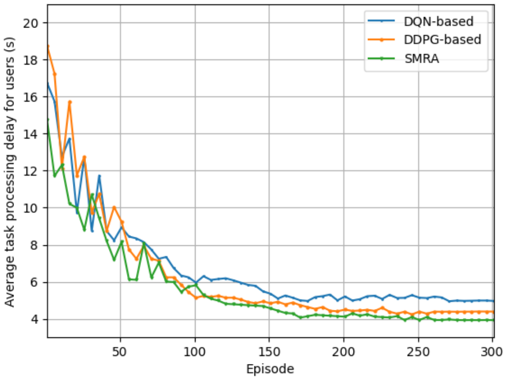


Fig. 8. Iterative changes of different schemes.

In order to analyze the superiority of SMRA algorithm from multiple perspectives, we further compare the SMRA algorithm with the DQN-based scheme and the DDPG-based scheme. Fig. 8 presents the variation of user average task processing latency with increasing number of iterations for different deep reinforcement learning based schemes, from which we can can find the average task processing delay of all three schemes gradually decreases as the number of iterations increases and finally converges to an optimal solution. However, the SMRA scheme can guarantee the minimum average user task processing delay for the final convergence.

VI. Conclusion

This paper investigated the joint optimization problem of service migration and resource allocation in edge IoT systems, fully considering the different resource requirements and mobility of IoT users, as well as the limited resources and heterogeneity constraints of edge servers, with the aim of minimizing the access delay of IoT users. The paper formulated the joint optimization SMRA problem as a MDP and proposed a DRL-based joint SMRA scheme, whereby the LSTM algorithm and PDQN algorithm can sense the user mobility and decide whether to migrate a service, where to migrate it, and how to reallocate the resources, effectively ensuring the resource utilization and service continuity of the

IoT system and reducing the access delay of IoT users. In addition, the PDQN algorithm effectively solves the complex discrete-continuous hybrid action space problem in the SMRA problem. Finally, we conducted simulation experiments using real datasets of Beijing cab trajectories to verify the effectiveness of SMRA algorithm and conducted comparison experiments to prove the superiority of the SMRA algorithm.

References

[1] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-Me Cloud: When Cloud Services Follow Mobile Users," in *IEEE Trans. on Cloud Computing*, Vol. 7, No. 2, pp. 369 – 382, Jun. 2019.

[2] T. Ouyang, Z. Zhou, and X. Chen. "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333-2345, 2018.

[3] B. Mada, M. Bagaa, T. Taleb, and H. Flinck, "Latency-aware Service Placement and Live Migrations in 5G and Beyond Mobile Systems," in *Prof. IEEE ICC'20*, Dublin, Ireland, Jun. 2020.

[4] Y. Chen, Y. Sun, C. Wang and T. Taleb, "Dynamic Task Allocation and Service Migration in Edge-Cloud IoT System Based on Deep Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16742-16757, 2022.

[5] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on markov decision process," IEEE/ACM Trans. Netw., vol. 27, no. 3, pp. 1272–1288, 2019.

[6] R. A. Addad, D. L. C. Dutra, T. Taleb, and H. Flinck, "Toward using reinforcement learning for trigger selection in network slice mobility," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2241–2253, Jul. 2021.

[7] R. A. Addad, D. Dutra, T. Taleb, and H. Flinck, "AI-based network-aware Service Function Chain migration in 5G and beyond networks," in *IEEE Transactions on Network and Service Management*, Vol. 19, No. 1, pp. 472 – 484, Mar. 2022.

[8] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang, "Multi-cell mobile edge computing: Joint service migration and resource allocation," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 5898–5912, Sep. 2021.

[9] Liu Y, Yu H, Xie S, et al. "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158-11168, 2019.

[10] Y. Dai, K. Zhang, S. Maharjan and Y. Zhang, "Edge Intelligence for Energy-Efficient Computation Offloading and Resource Allocation in 5G Beyond," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12175-12186, Oct. 2020.

[11] 3GPP, 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (Release 8), TS36.331 V8.20.0, Jun. 2013.

[12] Liu J, Zhang S, Nishiyama H, et al. "A stochastic geometry analysis of D2D overlaying multi-channel downlink cellular networks," *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015: 46-54.

[13] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432-7445, Aug. 2017.

[14] M. Horii, Y. Kojima, and K. Fukuda. "Stateful process migration for edge computing applications," in *Proceeding of 2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1-6, Barcelona, Spain, April, 2018.

[15] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[16] Lee C P, Lin P. "Modeling delay timer algorithm for handover reduction in heterogeneous radio access networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1144-1156, 2016.

[17] Wang X, Wang L, Cavdar C, et al. "Handover reduction in virtualized cloud radio access networks using TWDM-PON fronthaul," *Journal of Optical Communications and Networking*, vol. 8, no. 12, pp. B124-B134, 2016.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] Wang H, He H, Bai Y, et al. "Parameterized deep Q-network based energy management with balanced energy economy and battery life for hybrid electric vehicles," *Applied Energy*, 2022, 320: 119270

[20] Mnih V, et al. "Human-level control through deep reinforcement learning," *nature*, vol. 518. no. 7540, pp. 529–33, 2015.

[21] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (ddpg)-based energy harvesting wireless communications," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8577–8588, 2019.

[22] Tang Q, Xie R, Yu F R, et al. "Distributed task scheduling in serverless edge computing networks for the internet of things: A learning approach," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19634-19648, 2022.

[23] Y. Liu, H. Wang, M. Peng, J. Guan, and Y. Wang, "An incentive mechanism for privacy-preserving crowdsensing via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8616–8631, May 2021.

[24] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 316–324, 2011.

[25] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[26] Yuan Q, Li J, Zhou H, et al. "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9041-9052, 2020.

[27] Dorokhova M, Martinson Y, Ballif C, et al. "Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation," *Applied Energy*, 2021, 301: 117504.

[28] Ren J, Yu G, He Y, et al. "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031-5044, 2019.

Fangzheng Liu received the M.S. degree in computer science and technology from the North China University of Technology, China, in 2017. She is currently pursuing the Ph.D. degree with the School of Information Science and Engineering, China University of Petroleum, Beijing, China. From July 2022 to July 2023, she was a visiting Ph.D. student with the Center for Wireless Communications, University of Oulu, Finland. Her current research interests include edge/cloud/services computing, performance modeling and optimization.

Hao Yu (Member, IEEE) received the B.S. and Ph.D degree in communication engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2015 and 2020. He was also a Joint-Supervised Ph.D. Student with the Politecnico di Milano, Milano, Italy. He is currently a Postdoctoral Researcher with the Center of Wireless Communications, Oulu University, Oulu, Finland. His research interests include intelligent edge network, time sensitive networks, 6G deterministic networking.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3332421

Jiwei Huang (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in computer science and technology from Tsinghua University, Beijing, China, in 2009 and 2014, respectively. He was a Visiting Scholar with the Georgia Institute of Technology, GA, USA. He is currently a Professor and the Vice Dean of College of Information Science and Engineering / College of Artificial Intelligence, China University of Petroleum, Beijing, China, and the Director of the Beijing Key Laboratory of Petroleum Data Mining. He has authored or coauthored one book and more than 60 articles in international journals and conference proceedings, including IEEE Transactions on Mobile Computing, IEEE Transactions on Services Computing, IEEE Transactions on Cloud Computing, IEEE Transactions on Vehicular Technology, IEEE Internet of Things Journal, ACM SIGMETRICS, IEEE ICWS, etc. His research interests include Internet of Things, edge computing, services computing, etc. He is currently on the Editorial Board of Chinese Journal of Electronics and Scientific Programming, and served as TPC members of IEEE ICWS, CollaborateCom, PRICAI, etc. He is a senior member of IEEE and CCF.

Tarik Taleb (Senior Member, IEEE) received the B.E. degree (Distinction) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively.

He is currently a Professor at the Center of Wireless Communications, University of Oulu, Finland. He is the Founder and the Director of the MOSA!C Lab, Oulu. From October 2014 and December 2021, he was a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. Prior to that, he was working as a Senior Researcher and the 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. Before joining NEC and till March 2009, he worked as an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, in a lab fully funded by KDDI, the second largest mobile operator in Japan. From October 2005 to March 2006, he worked as a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. His research interests lie in the field of telco cloud, network softwarization and network slicing, AI-based software-defined security, immersive communications, mobile multimedia streaming, and next-generation mobile networking.

Dr. Taleb served as the General Chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference held in Marrakech, Morocco. He was the Guest Editor-in-Chief of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Series on Network Softwarization and Enablers. He was on the editorial board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE Wireless Communications Magazine, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and a number of Wiley journals. Till December 2016, he served as the Chair of the Wireless Communications Technical Committee. He has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group 2.