

A Service Migration Method Based on Dynamic Awareness in Mobile Edge Computing

Menglei Zhang

State Key Laboratory of Networking
and Switching Technology
Beijing University of Posts and
Telecommunications
Beijing, China
mlzhang@bupt.edu.cn

Haoqiu Huang

China Aerospace Science and Industry
Corporation Limited Network
Information Department
Beijing, China
906449452@qq.com

LanLan Rui

State Key Laboratory of Networking
and Switching Technology
Beijing University of Posts and
Telecommunications
Beijing, China
llrui@bupt.edu.cn

Guo Hui

State Key Laboratory of Networking
and Switching Technology
Beijing University of Posts and
Telecommunications
Beijing, China
guohui@bupt.edu.cn

Ying Wang

State Key Laboratory of Networking
and Switching Technology
Beijing University of Posts and
Telecommunications
Beijing, China
ywang@bupt.edu.cn

Xuesong Qiu

State Key Laboratory of Networking
and Switching Technology
Beijing University of Posts and
Telecommunications
Beijing, China
xsqiu@bupt.edu.cn

Abstract—Cloud computing technologies can not satisfy the requirements of applications on the mobile terminals because of their disadvantages in delay, link load and energy. So Mobile Edge Computing (MEC) is proposed as a kind of novel computing technology. As an important research direction of MEC, service migration methods still have limitations that they cannot learn migration paths and be adaptive in dynamic situation and user movement. In this paper, we propose a novel service migration policy method based on reinforcement learning. We firstly investigate user movement, four different edge network situations and traditional migration policies. Then we formulate the system requirements by Satisfiability Modulo Theory (SMT) logic to acquire the migration policy space. We further propose a dynamic-awareness deep Q-learning algorithm to select paths from the policy space iteratively and conduct dynamic awareness to adjust learning rate adaptively. Meanwhile, the optimal convergence of our algorithm is proved theoretically. Finally, the experimental results highlight the effectiveness as migration successful rate, service interruption time and load balance of our method compared to the other solutions.

Keywords—Service Migration, Mobile Edge Computing, User Movement, Dynamic Awareness

I. INTRODUCTION

Under the background of the Internet of Things, Mobile Edge Computing (MEC) has become a very important computing paradigm, which can alleviate the load on the link, decline service delay and improve user's satisfaction. At the same time, the movement of users will cause the migration of service running on the edge server. Surely, the completion of the migration process requires the cooperation of the user and the edge server. The traditional scheme to solve service migration problem is to use the Markov Decision Process (MDP) to model the change of service states and use the Value Iteration to calculate the optimal result [1]. When making migration decisions, network conditions are determined by various dynamically changing factors, which affects the migration success probability.

But based on the current researches in service migration techniques in MEC, there are also two main problems to be solved. The first one is that most migration methods are only efficient to intact edge network. If some edge nodes or links

break down, they cannot execute the migration process successfully. Secondly, migration controller cannot learn migration paths in previous research. A few methods only adopt reinforcement learning technology to decide whether to migrate the service on work, but cannot learn the migration paths.

In this paper, we propose a Dynamic-Awareness Deep Q-learning method for Service Migration (DASM) in MEC. Distinguished from the existing solutions, we innovatively analyze the dynamic change in the network quantitatively and apply deep Q-learning into migration methods, which can conduct dynamic awareness and adjust the learning rate adaptively to implement rapid convergence in learning process. The main contributions of this paper are shown in four aspects: (1) We build the system model, edge network model including nodes and transmission links and service migration policy model, (2) Based on the practical condition, we consider several network requirements including delay, link capacity and node resource requirements, (3) For the optimal migration policy, we design an intelligent DASM algorithm to complete service migration request with movement of the user, (4) To evaluate the performance of proposed DASM, we also conduct a series of simulations. The results highlight the superior performance of our algorithm in terms of successful migration performance comparing to the typical solutions.

The rest of the paper is organized as follows: Section II explains the related work. In Section III, system overview, edge network model and migration policy model are explained respectively. In Section IV, our proposed DASM algorithm is presented in a detail. Section V shows the experimental evaluation about DASM algorithm. Section VI makes the conclusion of this paper.

II. RELATED WORK

As for service migration models, the papers in [1][2] define Markov Decision Process model for state transition during the service migration. The model designs proper decisions determining whether to carry out migration by using value or policy iteration solutions, which can obtain the balance between migration cost and the experience of users. There are many factors affecting the migration policy. In [3],

D.Zhao et al proposed a service migration scheme based on energy consumption, cost, delay, link bandwidth and computing capacity on edge servers and apply the multi-attribute joint optimization algorithm to obtain the optimal destination server. However, this paper cannot solve the problem of service interruption. T. Taleb et al in [4] proposed two main schemes to guarantee the continuity of service. One is based on the Locator/Identifier Separation Protocol, of which IP addressing is replaced by service identification to guarantee stable migration. The other one is based on Software Defined Network (SDN) technologies which can separate the data layer from the control layer to decline the handover delay. As for user mobility, in [5], S. Wang et al employ one-dimension asymmetric random mobility model and use an improved iterative algorithm to find the optimal threshold for migration, which can decline the migration time. However, in actual scenario, the user mobility is not limited to one-dimension track. So in [6], A. Nadembega et al employ probability and Dempster-Shafer process to predict the location of user at next timeslot based on the behavior habits of users, of which the model can obtain the prediction destination and the subsequent transitions of road segments.

The traditional solutions to the migration problem require heuristic algorithm. However, in [7], Z.Gao et al designed a Reinforcement Learning-based framework for a single-user edge computing service migration system. And they take many requirements into account, but not user movement and link change. In [8], Min et al proposed a service migration mechanism in Edge Cognitive Computing, which the service are migrated based on the behavioral cognition of a mobile car. Besides the movement of cars, this paper focused on the service awareness, including emotion detection and video streaming. In [9], Cheng et al proposed a deep Q-network for task migration in MEC system, which can learn the optimal task migration policy from previous experience without necessarily acquiring the information about users' mobility pattern in advance. In [10], Florian et al formulated the service migration problem as a competitive multi-agent learning problem. Using their solution, the migration policies can be learned with respect to a large variety of optimization goals. From the above analysis, we see that reinforcement learning can be a novel solution to improve migration performance, which is expected to solve three main problems including network situation, service migration performance under breakdowns and convergence of learning rate.

III. SYSTEM MODEL

A. System Overview

As shown in Fig. 1, mobile edge cloud architecture can be divided into three layers: mobile terminals, edge servers and backend cloud [11]. There are many applications with an amount of data and computation on the terminals near users. Some of them can be offloaded to the connected edge servers to process. In MEC scenario, edge servers provide service to the terminals in a limited scale. When users move from here to there, the offloaded servers will transmit the service to guarantee the service running.

B. Edge Network Model

The user's terminal moves from here to there and sends the migration request to the offloaded edge server, which is

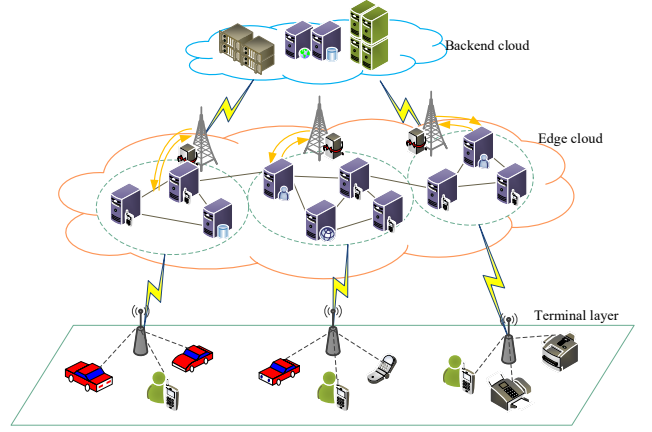


Fig. 1. Architecture of Mobile Edge Computing

defined as the **Original edge Node (ON)**. Therefore, Edge network can be modeled as an undirected connected graph $G=(V,E)$, which V means edge nodes in the edge network and E means links among servers. The model for edge nodes in edge network is relevant to the available capacity and computing ability. We use a two-tuples to define the edge nodes: $E_n=(AC,CA)$. The service components and data packets are transferred by links between edge nodes. We focus on the link stability, handling capacity and link bandwidth to model links: $E_l=(LS,HC, LB)$.

In the process of migration, the **Intermediate edge Nodes (IN)** or the **Destination Node (DN)** may be occupied by other service totally. We define this case as **Node Breakdown**. Similarly, the links between edge nodes may be out of order, so we need to find another feasible path to ensure the service migration. We define this case as **Link Breakdown**. As shown in Fig. 2, we will analyze the following situations.

1) Normal Migration

Each node has different ability degree, which we define as d_n . We also define AC_{max} , CA_{min} and CA_{max} respectively to represent maximal capacity value and limited computing ability of each node. d_n is considered as a random variable following uniform distribution which can be formalized as:

$$d_n = 0.5 * [AC + (AC_{max} - AC)(1 - \alpha)] + 0.5 * \{ (1 - \beta) [CA + (CA_{max} - CA)(1 - \alpha)] + \beta [CA - (CA - CA_{max})(1 - \alpha)] \} \quad (1)$$

Similarly, We can get the link ability degree as follows:

$$d_l = (1 - \beta) [HC + (LB - HC)(1 - \alpha)] + \beta [HC - (HC - HC_{min})(1 - \alpha)] \quad (2)$$

Where $\beta \in [0,1]$ is a uniform distribution parameter and $\alpha \in [0,1]$ is the node knowledge precision.

2) Node Breakdown

In this case, the available capacity or computation ability is occupied by other service. It is formalized as follows:

$$d_n = 0.5 * AC + 0.5 * \{ (1 - \beta) CA + \beta [CA - (CA - CA_{max})(1 - \alpha)] \} \quad (3)$$

where $AC=AC_{max}$, $CA=CA_{max}$, $\beta \in [0,1]$ and $\alpha \in [0,1]$.

3) Link Breakdown

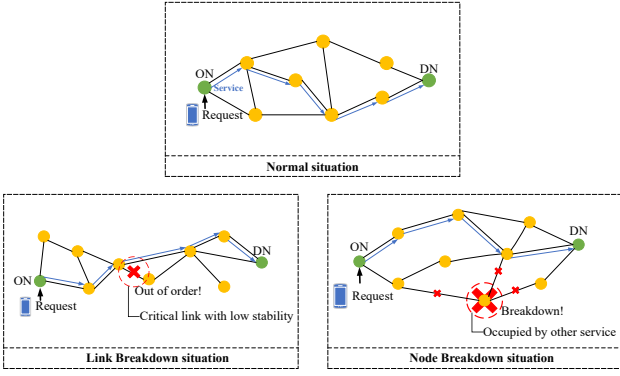


Fig. 2. Three edge network situations

Link Breakdown means the link with lower stability is likely to break down. There are two methods to describe the stability of links in the edge network. The first method is product of two nodes' ability degrees. The second method is betweenness centrality:

$$s_{ij} \triangleq \sum_{O \neq D} \frac{\sigma_{OD}(l_{ij})}{\sigma_{OD}}, \quad \forall l_{ij} \in E \quad (4)$$

where σ_{OD} is the number of shortest paths between ON and DN and $\sigma_{OD}(l_{ij})$ is the number of shortest paths passing through the edge l_{ij} respectively.

C. Migration Policy Constraints

We define requests set as $\mathcal{R}$ and use multidimensional vector $\mathbb{B}^{\tilde{r}} = \{s_i^{\tilde{r}} | \forall i \in V, \forall \tilde{r} \in \mathcal{R}\}$ to describe edge node j with the request $\tilde{r}$ is selected as s_j equals 0 or 1. Therefore, the migration policy space of specific request can be denoted as $\{\mathbb{B}_1^{\tilde{r}}, \mathbb{B}_2^{\tilde{r}}, \dots, \mathbb{B}_n^{\tilde{r}} | \forall \tilde{r} \in \mathcal{R}\}$. And in the migration policy space, we represent the DN space, the migration path space as $n^*, \mathcal{P}$ respectively. To complete the migration and communication success, there are many constraints:

1) Delay constraints

Communication delay includes transmission delay $t_t = \text{data}/BW$ and propagation delay $t_p = \text{dis}_{ij}/v$. Here data means the bite of the service and BW means the bandwidth of the transmission link. And v means the actual transmission rate depends on the architecture of the edge network. We use W_{od} to denote the delay constraints.

$$\sum_{i \in V} s_j^{\tilde{r}} t_t + \sum_{i \in V} \sum_{j \in \mathcal{L}} s_j^{\tilde{r}} s_j^{\tilde{r}} t_p \leq W_{od}, \quad \forall \tilde{r} \in \mathcal{R} \quad (5)$$

Here, $\mathcal{L}$ is the set of neighbors for node j . The hop's SMT formalization can be described as $\sum_{i \in V} s_j^{\tilde{r}} \leq D$, which means the total hop is smaller than threshold D . Besides, we define service interruption time (SDT) as the sum of t_t and t_p . $SDT = t_t + t_p$.

2) Available resource of DN constraints

Migration paths should not include those edge nodes that have unavailable resource for extra requests:

$$C_j^{\max} - \sum_{\tilde{r} \in \mathcal{R}} s_j^{\tilde{r}} c_j^{\tilde{r}} \geq C_j^{\text{th}}, \quad \forall j \in V \quad (6)$$

where $C_j^{\max}$ is the max capacity for node j , $c_j^{\tilde{r}}$ denotes the cost at node j for request $\tilde{r}$ and C_j^{th} is the minimum threshold for node j . $c_j^{\tilde{r}}$ can be described in (7).

$$c_j^{\tilde{r}} = C_j' \left(\xi \frac{c_j^{\tilde{r}}(j')}{C_j'} - 1 \right) \quad \forall j \in V, \quad \forall \tilde{r} \in \mathcal{R} \quad (7)$$

Where C_j' is the available capacity and ξ is a parameter that is set as $2n$, n is the total number of edge nodes.

3) Link bandwidth constraints

Edge servers connect with each other through the transmission links. To avoid the situation that the link capacity is not enough to transmit the data packet, we define $E_{e(j,j')}^{\tilde{r}}$ to denote whether link $e(j,j')$ is adopted to be migrated in (8).

$$E_{e(j,j')}^{\tilde{r}} = \begin{cases} 1, & \text{if link } e(j,j') \text{ is adopted to be migrated} \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

When the optimal migration path $p \in \mathcal{P}$ does not include link $e(j,j')$, $E_{e(j,j')}^{\tilde{r}}$ equals 0. We use $N_{jj'}$ to denote the capacity of link $e(j,j')$. The demand of transmission rate must be less than the capacity of the link as described in (9).

$$v \times (1 - E_{e(j,j')}^{\tilde{r}}) \leq N_{jj'}, \quad j \in V, \quad e \in E \quad (9)$$

D. Service Migration Policy Model

In our paper, we adopt a deep Q-learning based method to learn the optimal migration policy. There are three elements in reinforcement learning: a) State set, b) Action set, c) Reward obtained by action a in state s .

a) *State*: In the practical scenario, the edge server has a certain coverage. We define hops between ON and $IN j$ as state function $\mathcal{S}(t)$. To represent the state set, we suppose that the edge server connected with the user initially as the observation point O . In this way, we obtain state set $\mathbb{S}^j = \{0, 1, 2, \dots, d | d \leq D, j \in V\}$, in which we suppose $\{s_i | 1 \leq i \leq d+1\}$ as each state.

b) *Action*: We define $\mathbb{A} = \{A_{i,i+1}^{j,j'} | \text{edge server } j \rightarrow j', \text{ state } s_i \rightarrow s_{i+1}\}$ to represent the action set, which means at state s_i , the migration controller takes the action a_i^j and the service is migrated from edge server j to j' .

c) *Reward*: We describe the reward at timeslot t as $R_t(s, a)$, which is the function of state s_i and action a_i^j at timeslot t . If the service cannot be migrated to the edge server j' , $R_t(s, a)$ will be set negative values that are linearly related to costs. On the other hand, if the service can be migrated to server j' uninterruptedly, R_t will be set fixed positive value.

The reward is defined as the difference between the experience of user and the migration cost. The experience of user can be replaced by the response delay, which is determined by $d_i^{jj'}$, the distance between user i to the offloaded server j .

$$e^{ij}(s_t) = E - \lambda d_i^{jj'} \quad (10)$$

Where E is the maximal experience the user can enjoy, and it is a decreasing function of $d_i^{jj'}$, which means that $e^{ij}(s_t)$ declines when the user moves far away from the offloaded edge server. As for the cost during the migration process, we define it as $c_t(s_t, a_t) = \text{cost}_{\text{com}}(\text{dis}_{oj}) + \text{cost}_{\text{mig}}(\text{dis}_{oj})$, which is explained in Section IV.A. Therefore, the reward $r_t(s_t, a_t) = e^{ij}(s_t) - c_t(s_t, a_t)$.

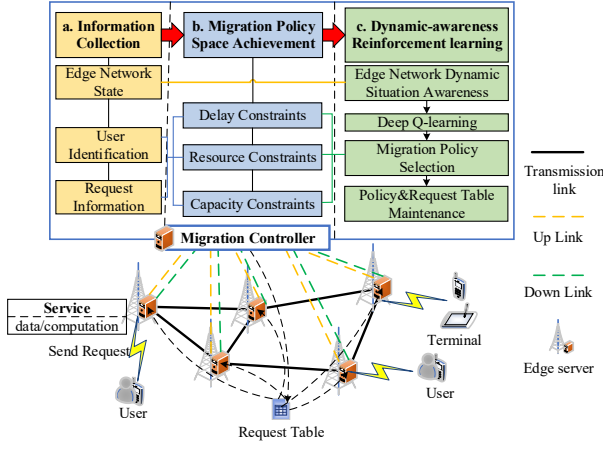


Fig. 3. The framework of our proposed algorithm

IV. DYNAMIC AWARENESS BASED SERVICE MIGRATION POLICY

The components of our method are shown in Fig. 3. The agent is the migration controller. In Section III, we define the models in Information Collection module. In this section, we define the policy space and propose the learning algorithm. The role of the user terminal is that the change of its location will cause its IP address to change. MEC receives the information of the IP change, which is applied to the start of service migration and is used to determine the optimal destination MEC, that is, the user identification in [12].

A. The deep Q-learning algorithm for service migration

The aim of migration controller is to maximize the total sum reward of these timeslots through deciding the actions of each timeslot from the first to the last one. *Policy* is defined as the action sequences through migrating at each state, the details shown as the following.

Definition 1. The migration controller's *policy* is about taking actions from time $t=1$ to $t=\infty$, defined in (11).

$$\pi = \{\phi_t(s_t), \forall t \in \{1, \dots, \infty\}\} \quad (11)$$

here $\phi_t(s_t)$ is a function at time t about mapping from one state s_t to an action. The goal of the controller is to maximize the total expected reward with an optimal *policy* π^* in (12).

$$\pi^* = \arg \left\{ \max_{\pi \in \Pi} E_{s_t}^{\pi} [\sum_{t=1}^{\infty} \gamma r_t(s_t, a_t)] \right\} \quad (12)$$

here γ represents a discount factor between (0,1).

First, the environment will give observation(*obs*). The agent will get all $Q(s, a)$ of *obs* according to the value function network, and then use ϵ -greedy to select the action and make a decision. The environment will give a reward and the next *obs* after receiving this action. After one step, we update the parameters of the value function network according to reward and proceed to the next step until a satisfactory value function network has been trained.

The ϵ -greedy strategy refers to the guidance that the agent obtains an action from the action set randomly with a possibility of ϵ or selects the optimal action corresponding to the maximum Q value from the calculated Q-values in the next timeslot with a probability of $1 - \epsilon$ in (13). p is a random number belonging to $[0,1]$. In our experiment, we set ϵ as 0.1.

$$\phi_t^* = \begin{cases} \arg \max_{a_t \in A} Q_t^*(s_t, a_t, \theta), & p = 1 - \epsilon \\ a \sim U(a), & p = \epsilon \end{cases} \quad (13)$$

So that we can avoid the situation that the state and action that are not collected during the sampling process can no longer be collected as well as the combination of exploration and exploitation is realized.

During the state iteration process, the system updates the state value at each step of the exploration. The update formula is using the Bellman equation in (14).

$$Q_t^*(s_t, a_t) = E_{s_{t+1}} [r_t(s_t, a_t) + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1})] \quad (14)$$

Where γ represents a discount factor between (0,1). For DQN algorithm, we design the target Q network to calculate the target, which is different from the predicted value. Target is defined in (15).

$$target = r_t + 1 + \gamma \max_{a_t \in A} Q_t(s_{t+1}, a_t, \theta_t^-) \quad (15)$$

Algorithm 1: Dynamic-awareness DQN Algorithm for Service Migration Policy

Input: State/reward Matrix (R) is a $n \times m$ matrix, which n is the number of network states and m is the number of potential paths.

Output: Optimal migration policy to adapt dynamic situation (Matrix P).

- 1: Initialize Replay memory H
 - 2: Initialize Q function with random θ
 - 3: Initialize $\widehat{Q}$ function with parameters θ^-
 - 4: **for** each episode **do**
 - 5: $t \leftarrow 1; s_1 \leftarrow s_i \in \mathcal{S}$.
 - 6: **for** $t \in [1, T]$ **do**
 - 7: Generate a random number p in $[0,1]$
 - 8: select the action ϕ_t^* by ϵ -greedy policy according to (14)
 - 9: $s_{t+1} \leftarrow s_t$
 - 10: Calculate $r_t(s_t, a_t)$.
 - 11: Store transition (s_{t+1}, a_t, r_t, s_t) in H
 - 12: Sample random minibatch of (s_{j+1}, a_j, r_j, s_j) from H
 - 13: Set $y_i = \begin{cases} r_j, & \text{if } j+1 \text{ is the last state} \\ r_j + \gamma \max_{a_{t+1} \in A} Q_t(s_{j+1}, a_{j+1}, \theta_j^-), & \text{otherwise} \end{cases}$
 - 14: Execute a gradient descent step on $(y_i - Q_t(s_t, a_t, \theta))^2$ by θ .
 - 15: Reset $\widehat{Q} = Q$ in every C steps
 - 16: $t \leftarrow t+1$
 - 17: **end for**
 - 18: **end for**
 - 19: **return** P
-

The mean squared error (MSE) of loss function is defined in (16).

$$\overline{L_t^2(\theta_t)} = E_{s_{t+1}, a_t, r_t, s_t} \left[\left(y_j - Q_t(s_t, a_t, \theta_t) \right)^2 \right] \quad (16)$$

Using differentiating to get the loss function's gradients in (17).

$$\nabla_{\theta_t} \overline{L_t^2(\theta_t)} = E_{s_{t+1}, a_t, r_t, s_t} \left[y_j - Q_t(s_t, a_t, \theta_t) \nabla_{\theta_t} Q_t(s_t, a_t, \theta_t) \right] \quad (17)$$

The gradient $\nabla_{\theta_i} Q_i(s_i, a_i, \theta_i)$ guides loss function to be reduced in a feasible direction. Therefore, update the parameter as shown in (18).

$$\theta_{i+1} = \theta_i + \alpha \nabla_{\theta_i} \overline{L_i^2}(\theta_i) \quad (18)$$

here, α denotes the learning rate. In our project, in order to identify the consequence of our algorithm, we should redefine α by considering the network situation as follows.

The migration controller needs to get the cost $C_t = \{c_t(s_t, a_t) | \forall j \in V, t \in N^+\}$ of migrating the service. If the node j breaks down, the affect gain is defined as $G_t^b = \{g_j^b(t) | \forall j \in V, t \in N^+\}$. If the node j is available, the engine reward is defined as $R_t^e = \{r_j^e(t) | \forall j \in V, t \in N^+\}$. So, the edge network dynamic matrix is represented in (19).

$$X_{L \times J} \triangleq \begin{cases} x_{ij} = -g_j^b(t), & \text{if node } j \text{ breaks down} \\ x_{ij} = r_j^e(t), & \text{if node } j \text{ is available} \\ x_{ij} = 0, & \text{otherwise} \end{cases} \quad (19)$$

where L denotes the number of time slots and J denotes the number of edge nodes. The dynamic value of timeslot t_0 is described in (20).

$$V(t_0) = \sum_{t=1}^{t_0} [\sum_{i \in L, j \in J} x_{ij} + \sum_{m \in M} c_t^m(t) - c_t] \quad (20)$$

here M is the set of broken nodes. We use the negative derivative of V to represent the dynamic situation in (21).

$$K = -V'(t_0) = -\lim_{\Delta t \rightarrow 0} \frac{V(t_0 + \Delta t) - V(t_0)}{\Delta t} \quad (21)$$

$$\alpha(K, \tau) \triangleq \frac{1}{(1 + e^{-K})^\tau} \quad (22)$$

B. Numerical analysis of our algorithm

Learning rate $\alpha_t(s_t, a_t)$ is depended on the current network situation. To show the convergence of the optimal action value function $Q^*(s_t, a_t)$, we adopt the convergence theorem of the stochastic sequence from [13].

Theorem 1. The proposed DASM algorithm can converge to the optimal action value with probability 1.

Proof. The random iteration process I_t takes values in R_n which is defined in (23).

$$I_{t+1}(s) = (1 - \alpha_t(s))I_t(s) + \alpha_t G_t(s) \quad (23)$$

The learning process converges to 0 with probability under the following assumptions in (24).

$$0 \leq \alpha_t \leq 1, \sum_{t=0}^{\infty} \alpha_t(s) = \infty, \sum_{t=0}^{\infty} \alpha_t^2(s) < \infty \quad (24)$$

In our algorithm, we define $I_{t+1}(s, a) = Q_t(s_t, a_t) - Q^*(s_t, a_t)$, $G_t(s, a) = R_t + \gamma \max_{a \in A} Q_t(s_t, a') - Q^*(s_t, a_t)$.

Next, we will prove DASM algorithm satisfies the above three assumptions. In our algorithm, $r_t(s_t, a_t)$ is bonded and all the state-action spaces are visited infinitely. The value of sigmoid function $\frac{1}{(1 + e^{-K})^\tau}$ is a constant Ψ which is between 0 and 1. From (22), we can get the expansion equation of $\sum_{t=0}^{\infty} \alpha_t(s)$ in (25).

$$\sum_{t=0}^{\infty} \alpha_t(s) = \frac{1}{(1 + e^{-K_1})} + \frac{1}{2(1 + e^{-K_2})} + \dots + \frac{1}{\tau^*(1 + e^{-K^*})} + \dots \quad (25)$$

Here, τ^* means the equilibrium begin time. From (21), we can get that the sum of series items before τ^* can be shown as a constant defined as Ω_1 , and the left part can be replaced with F_1 , then we can rewrite (25) as $\sum_{t=0}^{\infty} \alpha_t(s) = \Omega_1 + F_1$. F_1 can be expanded in (26) as follows.

$$F_1 = \frac{1}{(1 + e^{-K^*})} \left(\frac{1}{\tau^*} + \frac{1}{\tau^* + 1} + \dots \right) = \frac{1}{(1 + e^{-K^*})} \left[\left(1 + \frac{1}{2} + \dots \right) - \left(1 + \frac{1}{2} + \dots + \frac{1}{\tau^* - 1} \right) \right] = \frac{1}{(1 + e^{-K^*})} \sum_{\tau=1}^{\infty} \frac{1}{\tau} - \Omega_2 = \Psi \sum_{\tau=1}^{\infty} \frac{1}{\tau} - \Omega_2 \quad (26)$$

In this way, we can learn that $\sum_{t=0}^{\infty} \alpha_t(s) = \Omega_1 + \Psi \sum_{\tau=1}^{\infty} \frac{1}{\tau} - \Omega_2$, in which Ω_1 , Ω_2 and Ψ are all constants. And we can learn that series $\sum_{\tau=1}^{\infty} \frac{1}{\tau}$ is a divergent series. So, $\sum_{t=0}^{\infty} \alpha_t(s) = \infty$.

Similarly, we can also prove that $\sum_{t=0}^{\infty} \alpha_t^2(s) = \Omega_3 - \Omega_4 + \Psi^2 \sum_{\tau=1}^{\infty} \frac{1}{\tau^2}$. And $\sum_{\tau=1}^{\infty} \frac{1}{\tau^2}$ is a convergent series. So, $\sum_{t=0}^{\infty} \alpha_t^2(s) < \infty$. Above all, we have proved that $\alpha_t(s)$ can converge to 0, and our algorithm can converge to the optimal action value with probability 1.

Assuming that $|S|$ is the number of network states, $|A|$ is the number of migration engine's actions and $|T|$ is the number of time slots. So, the space complexity is $O(|S||A|)$. Q-learning is a kind of value iteration algorithm, which max time complexity is $O(|S|^2|A|)$. Considering the dynamic awareness method, the max time complexity of DASM is $O(|S|^2|A||T|)$.

V. PERFORMANCE EVALUATION

In this section, we compare the proposed algorithm with the following three other methods: (1) the method which service migration based on multi-attribute (MADA) is applied [3], (2) the method which migration policy based on deep Q-learning (DQN) is applied [8], (3) never migrate service (NSM). We would use several "incorrect" transition probability because user's ground truth transition probability usually contains noise. And our proposed algorithm has no requirement for transition probability.

We use Mininet-WiFi to simulate our scenario with a 4×4 grid distribution area. The user moves following the planned track in [5] in the area with assumption that the length of timeslot is 2s and the number of terminals and MEC servers is set to 50 and 5 respectively. To implement the experiment, we use the toolkit inside the Python and we use the toolkit inside Tensorflow to finish the training process of our proposed algorithm. The link capacity $N_{jj'}$ (in Bytes) between two MECs obeys uniform distribution between (50, 300). We set delay constraints W_{od} for each session obeys uniform distribution between (3, 6). MEC's capacity for each resource C_j^{th} (in MIPS) obeys uniform distribution between (100, 400).

1) The success rate of overcoming breakdowns

Avoiding the effect of node or link breakdowns is one of the important elements in the service migration process. We simulate 2.3×10^5 time slots for every round and calculate the success rate of recovering the breakdowns in every time slot. As shown in Fig. 4, for the breakdown recovery of link, the success rate increases from 5.5% to 23% with increase of 17.5%. For node breakdown recovery, the success rate increases from 6.6% to 20.8% with an increase of 14.2%. So, we know that the breakdown recovery effect for the link and

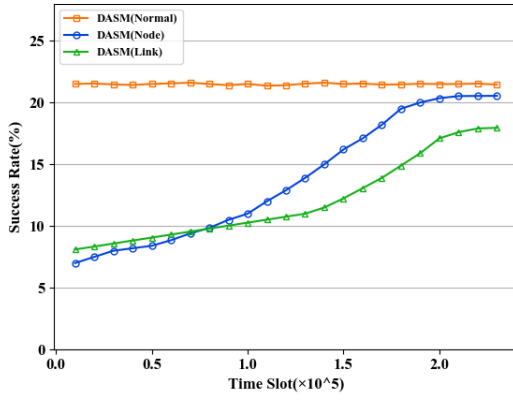


Fig. 4. The success rate of overcoming the breakdowns

available resource capacity of the edge server and the bandwidth of the transmission link into account for the constraint elements. In this way, we can ensure that the algorithm does not cause breakdowns again when it is executed, which can improve the success rate of fault recovery.

2) The convergence in DRSM algorithm

As shown in Fig. 5. The blue curve represents the influence of the learning rate based on network dynamic awareness on the convergence speed of our proposed algorithm [14]. The purple curve represents the case that the learning rate is set as 0.9. The learning rate based on the network dynamic awareness gradually converges to a certain value at 1.6×10^5 timeslots, but the purple curve represents that the fitness gradually increases with the time passing by. This indicates that the DASM algorithm based on network dynamic awareness can make better use of the historical state-action (s, a) data set. Because the learning rate in this solution is continuously adjusted according to the network situations.

3) Service Interruption Time Analysis

We know that the user's movement may cause the service interruption if the terminal does not send the migration request to the edge server. But the service migration does not guarantee the continuous service totally. In Fig. 6, at every 40 rounds, we calculate the SDT value. Then, we calculate the average value of SDT values of these methods. As shown in Fig. 8, we learn that the average SDT of our DASM method is the lowest. Other methods' SDT values are 2.9ms, 3.1ms and 3.5ms respectively. And the NSM method is the highest, and some of SDT values are too large to guarantee the service going on. Besides of taking the users' movement, we also consider the edge network's dynamic changes. So DASM method performs better than MADA and DQN methods which do not care about the dynamic adaption.

VI. CONCLUSION

In this paper, we have proposed a novel service migration policy method based on reinforcement learning. First, we investigate user movement, edge network situations and traditional migration policies. Then, we obtain migration policy space based on system constraints. We further propose a dynamic-awareness deep Q-learning algorithm to select paths from the policy space iteratively and conduct dynamic awareness to adjust learning rate adaptively. Finally, we

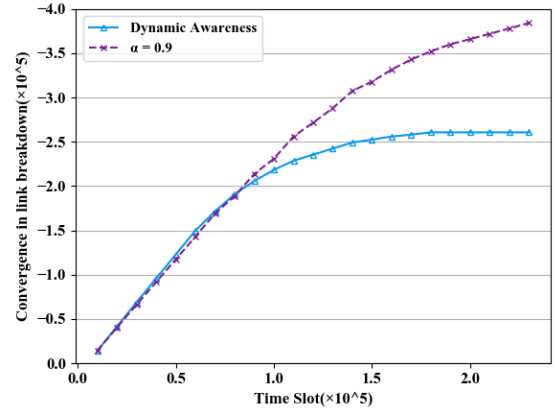


Fig. 5. The convergence of DASM under normal situation

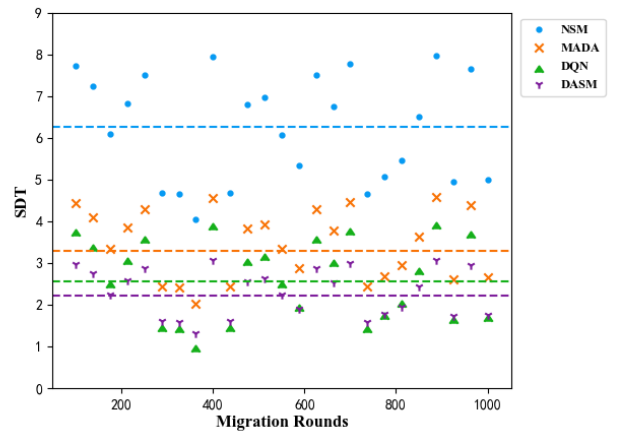


Fig. 6. SDT of the residual energy of edge nodes

execute series of simulation experiments to confirm the dynamic and universality of our method. For future work, we can design a solution to decline the computation complexity and simplify the state space by studying the reinforcement learning algorithm.

REFERENCES

- [1] A. Ksentini, T. Taleb, and M. Chen, "A markov decision processbased service migration procedure for follow me cloud," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1350–1354, IEEE, 2014. doi:10.1109/icc.2014.6883509.
- [2] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *Proceedings of the IEEE Global communications Conference (GLOBECOM)*, pp. 1291–1296, IEEE, 2013. doi: 10.1109/glocom.2013.6831252.
- [3] D. Zhao, T. Yang, Y. Jin and Y. Xu, "A service migration strategy based on multiple attribute decision in mobile edge computing," *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, Chengdu, 2017, pp. 986-990. doi: 10.1109/ICCT.2017.8359782.
- [4] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013. doi: 10.1109/mnet.2013.6616110.
- [5] P. Nurmi, "Long-term monitoring of human mobility on smartphones," *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, Budapest, 2014, pp. 236-236. doi: 10.1109/PerComW.2014.6815209.
- [6] A. Nadembega, A. Hafid and T. Taleb, "A Destination and Mobility Path Prediction Scheme for Mobile Networks," in *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2577-2590, June 2015. doi: 10.1109/TVT.2014.2345263.

- [7] Z. Gao, Q. Jiao, K. Xiao, Q. Wang, Z. Mo and Y. Yang, "Deep Reinforcement Learning Based Service Migration Strategy for Edge Computing," *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, San Francisco East Bay, CA, USA, 2019, pp. 116-1165.doi: 10.1109/SOSE.2019.00025.
- [8] Cheng Zhang, Zixuan Zheng, "Task migration for mobile edge computing using deep reinforcement learning" *Future Generation Computer Systems*, Volume 96, 2019, Pages 111-118, ISSN 0167-739X. doi:10.1016/j.future.2019.01.059.
- [9] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental deployment and migration of geodistributed situation awareness applications in the fog," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems(DEBS)*, pp. 258–269, ACM, 2016. doi: 10.1145/2933267.2933317.
- [10] Florian Brandherm, Lin Wang, and Max Mühlhäuser. 2019. "A Learning based Framework for Optimizing Service Migration in Mobile Edge Clouds" in *2nd International Workshop on Edge Systems, Analytics and Networking (EdgeSys '19)*, March 25, 2019, Dresden, Germany.ACM,NewYork,NY,USA.doi: 10.1145/3301418.3313939.
- [11] Urgaonkar, R., Wang, S., He, T., Zafer, M., Chan, K., & Leung, K. K. (2015). Dynamic service migration and workload scheduling in edge-clouds. *Performance Evaluation*, 91, 205-228. doi: 10.1016/j.peva.2015.06.013.
- [12] S. Wang, J. Xu, N. Zhang and Y. Liu, "A Survey on Service Migration in Mobile Edge Computing," in *IEEE Access*, vol. 6, pp. 23511-23528, 2018.doi: 10.1109/ACCESS.2018.2828102.
- [13] S. Singh, R. L. Lewis, A. G. Barto and J. Sorg, "Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective," *IEEE Transactions on Autonomous Mental Development*, 2010, 2(2): 70-82. doi: 10.1109/TAMD.2010.2051031.doi: 10.1109/tamd.2010.2051031.
- [14] T. Jaakkola, M. I. Jordan, et al., "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Computation*, 1994, 6: 1185-1201.doi: 10.21236/ada276517.