Journal of Network and Computer Applications 183-184 (2021) 103058

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Service migration in multi-access edge computing: A joint state adaptation and reinforcement learning mechanism

LanLan Rui, Menglei Zhang, Zhipeng Gao, Xuesong Qiu, Zhili Wang, Ao Xiong *

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

ARTICLE INFO

Keywords:
Service migration
Multi-access edge computing
User movement
State adaptation

ABSTRACT

With the development of the internet of things (IoT), the concept of an edge network has been gradually expanding to other fields including internet of vehicles, mobile communication networks and smart grids. Because the resources of terminals are limited, the long-distance movements of users will increase the running costs of the services that are offloaded to edge servers, and even the services on terminals will stop running. Another problem is that resource shortages or hardware failures of these edge networks can affect the service migration policy. In this paper, a novel service migration method based on state adaptation and deep reinforcement learning is proposed to efficiently overcome network failures. Before migration, we define four edge network states to discuss the migration policy and adopt the two-dimensional movement around the edge servers to adapt to the applications scenarios of our work. Then, we use the satisfiability modulo theory (SMT) method to solve the candidate space of migration policies based on cost constraints, delay constraints and available resource capacity constraints to shorten the interruption time. Finally, the service migration problem can be transformed into the optimal destination server and low-cost migration path problem based on the Markov decision process by the deep Q-learning (DQN) algorithm. Moreover, we theoretically prove the rate of convergence in the learning rate function of our algorithm to improve the convergence rate. Our experimental results demonstrate that our proposed service migration mechanism can effectively shorten the delays from service interruptions, and better avoid the impact of edge network failure on the migration results and, thus, improve the users' satisfaction.

1. Introduction

With the diversification of mobile terminals and the intelligent development of terminal applications, the concept of edge computing not only can be applied to mobile communication networks, but also can be extended to wireless access networks (De and Grassi, 2019). Therefore, European Telecommunications Standards Institute (ETSI), which is a non-profit telecommunications standardization organization approved by the European Commission in 1988, introduced the concept of multi-access edge computing (MEC). This paradigm of edge computing has the following advantages: First, the edge computing servers are close to mobile terminals. Using this paradigm can save the construction cost of the edge devices and improve the calculation efficiency with the users' fragmented and dynamic service requirements; Second, MEC can only process the localized cloud service requirements near the edge terminals. Thus, large enterprises can process sensitive data locally by using MEC paradigm in order to keep the internal data of the enterprise secret (Wang et al., 2018a). Third, operators can broaden their product capabilities and types of service by the superiority of the MEC architecture, which can bring more business opportunities especially in 5G-Generation (Wang et al., 2018b). Therefore, MEC is a very meaningful paradigm in the era of IoT. As shown in Fig. 1, the architecture of MEC can be divided into three layers: the backend layer, edge node layer and terminal layer. The terminal layer closest to the user provides services to users directly, but there are some services that must to be offloaded to the edge cloud layer to be finished. Different areas keep communication with each other in way of constructing base stations linked by communication optical cable. Some of the services on the edge cloud can be uploaded to the backend cloud. Thus, other edge servers or users can download them directly to save processing costs. There are a lot of task including data processing and resource allocation that must be accomplished on the terminals for many applications. Some of these

* Corresponding author.
E-mail addresses: llrui@bupt.edu.cn (L. Rui), mlzhang@bupt.edu.cn (M. Zhang), gaozhipeng@bupt.edu.cn (Z. Gao), xsqiu@bupt.edu.cn (X. Qiu), zlwang@bupt.edu.cn (Z. Wang), xiongao@bupt.edu.cn (A. Xiong).

https://doi.org/10.1016/j.jnca.2021.103058
Received 9 March 2020; Received in revised form 8 January 2021; Accepted 22 March 2021
Available online 5 April 2021

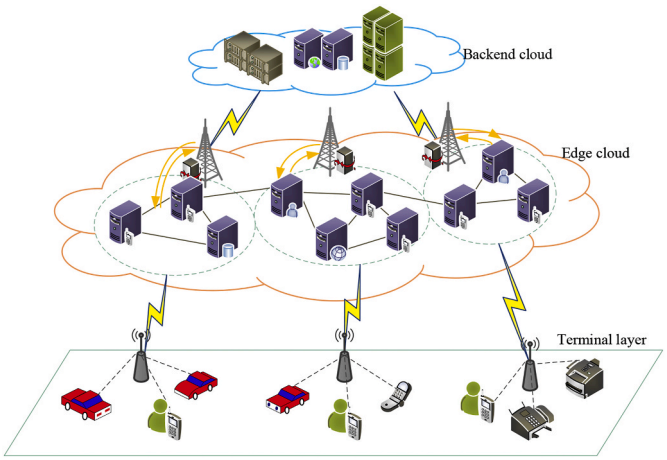


Fig. 1. The architecture of MEC.

tasks can be offloaded to the connected edge servers to be finished. However, the computational capacity and available resource of the edge servers may be limited, and thus, some of tasks should be uploaded to the backend cloud to be processed. In this way, we can decline the delay on the premise of finishing services normally, which can improve the users' experience.

Next, we introduce the meaning of service migration. Each edge server has a certain coverage, and thus, we define this area as a cell. Mobile terminals are randomly distributed in different cells. When a mobile terminal moves from this cell to that one, the previous server can no longer provide services for the terminal due to the limited coverage. To ensure that the service can be successfully finished, the previous ongoing service must be suspended and migrated to the destination server to execute, which is called service migration.

In our paper, our goal is to propose a solution for selecting the optimal destination server and obtain the migration path for services. During the service migration process, a series of problems will occur. For example, the decision of migration must consider the load balance among the servers. Service migration will bring energy consumption and service execution delay (Hossain et al., 2020). At the same time, inefficient migration will cause service interruption for a long time even stop the service. The service migration process is usually modeled as a Markov decision process. In addition to modeling the state set, action set, and reward function, the state transition probability matrix must be confirmed. Traditional migration decisions must consider multiple factors when making decision constraints, such as the link bandwidth, energy consumption, available space and computing power on the edge server. When solving the reward function, we must consider reducing the delay of the service migration and reducing the data packet loss rate.

However, based on the current research on service migration techniques, there are also two main problems to be solved. The first problem is that most migration methods are efficient only in intact edge network. If some edge nodes or links break down, they cannot execute the migration process, which means that most methods cannot adapt to the edge network states. Second, a migration controller cannot learn the migration paths in previous research. A few adopt the intelligence algorithms for migration technology to decide how to migrate the service. Hence, it is difficult to maximize effectiveness of the service migration techniques based on the current methods. In this way, we propose a **S**tate-adaptation **R**einforcement learning method for **S**ervice **M**igration (**SRSM**) in MEC. Distinguished from the existing solutions, we innovatively analyze the different states of edge network quantitatively and apply deep Q-learning (Liu et al., 2020) into migration methods, which can adjust the learning rate adaptively to implement rapid convergence in the learning process. The main contributions of this paper are shown in the following four aspects:

(1) We build the edge network model including the nodes and transmission links, the user movement model and request information model, which are used to quantify the performance of edge network. By modeling the edge nodes and links, we propose four types of states including the normal state, node failure, link failure, and multitarget failure that could occur in the edge network, which represents the dynamic changes of the edge network. This approach not only provides a normal service migration environment, but also improves the applicability and universality of the algorithm.
(2) To determine the correct migration policy candidate space, we consider certain network requirements, including the cost, delay and available resources capacity requirements based on the practical conditions. According to SMT, the above requirements are all transformed into global network-based constraints, which can guarantee uniformity and applicability in our service migration policy.
(3) We design the SRSM algorithm to complete the service migration process and obtain the optimal migration policy. We assume that there is a migration controller component in the system and the migration mechanism performs on it. The mechanism is divided into three parts, namely, information collection, migration policy space constraint and state-adaptation service migration policy. As far as is known, this mechanism can have a better effect than other service migration methods in saving costs and avoiding the influence of network failures.
(4) To evaluate the performance of the proposed algorithm, we also conduct a series of simulations. The results highlight the superior performance of our algorithm in terms of successful migration performance comparing to traditional solutions. Additionally, we have proven that the learning rate of our algorithm can converge to a stable value by considering the effect of the edge network's failures.

The remainder of this paper is organized as follows: Section 2 explains the related work. In Section 3, the overview, edge network model, user movement model and migration policy model are explained. In Section 4, our proposed algorithm is presented in detail. Section 5 shows the experimental evaluation of the SRSM algorithm. Section 6 presents the conclusions of this paper.

2. Related work

In this section, the related work on MEC, service migration methods and reinforcement learning is reviewed. By investigating the development of cloud computing and the edge computing paradigm, we can determine that the hierarchical structure of multi-access edge computing is more suitable for scenarios in which there are many types and numbers of terminals. And this structure can meet the needs of the users. By investigating previous service migration methods and their effects, we can propose innovations based on existing methods to reduce the migration overhead and interruption time. Based on the application of reinforcement learning in the field of service migration, we can understand that the convergence of learning process is exactly the problem that should be improved in this paper.

2.1. Multi-access edge computing

Currently, traditional cloud computing cannot meet the needs of some requirements of the latest applications, such as low delay, low energy and high bandwidth. To solve this problem, many edge computing technologies have been proposed in recent years. In Roman et al. (2018), Rodrigo et al. compared and analyzed the security threats and challenges from a holistic perspective, such as fog computing, mobile edge computing (MEC) and mobile cloud computing (MCC). In MacHen et al. (2016), Andrew et al. proposed a three-layer framework

for migrating running applications that are encapsulated either in virtual machines (VMs) or containers. Based on this framework, Wang et al. in Wang et al. (2018b) regarded the service migration process as a Markov decision process (MDP) model, and a Reinforcement Learning (RL) solution was proposed in this paper for the first time.

Although the above studies are about edge computing paradigms, they did not combine the software definition network (SDN) with edge computing to discuss the service migration policy. By combining SDN with multi-access edge computing, we can effectively manage resources on edge servers during the migration process.

2.2. The service migration methods

Researchers at Carnegie Mellon University (Bittencourt et al., 2015) proposed the concept of virtual machine (VM) switching and used cloudlet as the carrier. Although the dynamic migration technology of VMs is sufficiently mature, it is impossible to fully virtualize the system because of the existence of context-aware programs. In addition, VM switching must consume substantial resources on the edge servers. In this case, S. Hykes et al. (Jing et al., 2018) standardized the container component in the Linux system, which can reduce the complexity and cost of the migration process. The above research introduces the migration of different components of the services. In our paper, we adopt the migration of containers to simplify the migration process.

For service migration models, the paper (Wang et al., 2018b) defined a Markov decision process model for state transition during service migration. This paper designed proper decisions to determine whether to conduct migration by using value or policy iteration solutions, which can obtain a balance between the migration cost and the experience of users. There are many factors that affect the migration policy. In Ksentini et al. (2014), D.Zhao et al. proposed a service migration scheme that was based on energy consumption, cost, delay, link bandwidth and computing capacity on edge servers, and they applied the multi-attribute joint optimization algorithm to obtain an optimal destination server. However, this paper cannot solve the problem of service interruption. T. Taleb et al. in Taleb et al. (2019) 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 the migration process stable. The other one is based on SDN technologies which can separate the data layer from the control layer to decline the handover delay. For the influence of user mobility on migration policy, in Wang et al. (2014), S. Wang et al. employed a one-dimension asymmetric random mobility model and used an improved iterative algorithm to find the optimal threshold for migration, which can decline the migration time. However, in an actual scenario, the user mobility is not limited to the one-dimension track. As a result, in Urgaonkar et al. (2015), A. Nadembega et al. employed probability and the dempster-shafer process to predict the location of a user at the next timeslot based on the behavior habits of users, from which the model could obtain the predicted destination and the subsequent transitions of the road segments.

Although most of the current work adopts the Markov decision process method to model the service migration process, the difference in this paper is that those methods did not consider the effect of the edge network states on the migration paths. However, we consider the edge network failure situations by breaking the edge nodes or transmission links at an appropriate scale. The goal is to minimize the migration interruption time and communication cost under a failure situation.

2.3. Reinforcement learning

At present, because the service migration problem is an NP-hard problem, the multi-objective problem requires a heuristic algorithm. However, the edge environment changes all the time, and thus, the network states are affected by many elements, such as the server capacity, link bandwidth and user movement. In this case, we adopted reinforcement learning rather than the traditional heuristic algorithm to make our simulation practical and simplified. The basic idea of reinforcement learning is to learn the optimal strategy by maximizing the cumulative reward value obtained from the environment. In Saurez et al. (2016), Cheng et al. proposed a deep Q-network for task migration in a mobile edge computing system, which can learn the optimal task migration policy from previous experience without necessarily acquiring the information about the users' mobility pattern in advance. In Gao et al. (2019), Z. Gao et al. designed Q-learning and deep Q-learning algorithms for a single-user edge computing service migration system, in which they accounted for many requirements except for the user movement track and link capacity. However, in Chen et al. (2019), Min et al. proposed a service migration mechanism that was based on the behavioral cognition of a mobile car and service awareness, including emotion detection and video streaming in Edge Cognitive Computing. In Brandherm et al. (2019), Florian et al. formulated the service migration problem as a competitive multi-agent learning problem. From the above analysis, it can be seen that reinforcement learning can be a novel solution to improve migration performance, and it is expected to solve three main problems including user mobility, network states and learning convergence.

Although the above studies are all about reinforcement learning applied to a task or service migration in edge computing, in this paper, we aim to improve the convergence rate and learning rate of the algorithm for the purpose of saving the computing resources on the edge servers and reducing the migration decision time. The comparisons between our work and current service migration methods based on DQN are listed in Table 1.

3. Service migration model based on state-adaptation and edge network constraint

In this section, we introduce the workflow and the modular model of service migration first. Then, we discuss the information collection module, migration policy space constraint module and service migration policy module to elaborate the relationships among the functional modules in this paper.

3.1. Problem description

As shown in Fig. 2, the service migration is the process in which the user moves until reaching the migration threshold, and then initiates a request to the migration controller. The controller updates the migration strategy candidate space after collecting the user movement information and network state information. Then the controller calculates the optimal destination server and the migration path by using the DQN algorithm. Last, the obtained result is sent to the source server, and the server packages the data of previously running service to the destination server to complete the migration process.

In this process, the edge server initiates a migration request to the migration controller by detecting a change in the IP address of the terminal, which is not only initiated by the user's moving exceeding the migration threshold but also by insufficient capacity or computing power on the connected server.

Our method includes three parts, information collection, migration policy space constraint and state-adaptation service migration policy, shown in Fig. 3. First, we must collect the edge network states and user movement information to determine the start of the service migration. In addition, service requests must be quantitatively modeled. By using the user movement and service request information obtained in the previous step, we can use the SMT method to constrain three indexes to obtain the candidate migration policy space. In the third step, we must transform the service migration problem into an MDP problem, and define state variables, action variables, and reward. In addition, the situation matrix must be established according to the four state models, to obtain the maximum benefit. We should note that the policy and

Table 1
Comparisons between our work and current service migration methods.

Study	Environment	Objective	Performance metrics	Applications
Bittencourt et al. (2015)	Cloudlet virtual machine switching	the handoff time	Resource consumption subscription charge	interactive applications
Jing et al. (2018)	Pervasive cloud computing	the cloud resources needed by mobile applications	CPU prediction, CPU mean/ standard deviation, memory weighted mean, memory mean/ standard deviation	–
Ksentini et al. (2014)	3GPP LTE mobile network	service continuity	probability to be connected to the optimal data center and the average distance from it, average latency, cost of the service migration, service disruption time	Cloud-based services
Wang et al. (2014)	Three-layer framework in mobile edge cloud	the service downtime and overall migration time	total migration time and transferred data for LXC and KVM setups, total migration time under different RAM usage and bandwidth	Applications encapsulated either in a container or in a virtual machine
Li et al. (2019)	Fog computing in cellular networks	Improve the service continuity and QoS	Latency, reliability and migration cost	Vehicular communication
Tang et al. (2019)	Containers migration in fog computing	Reducing the cost of power consumption and delay, besides the migration cost by deep Q-learning	The comparison of CPU consumption and migration cost between container and VM, performance including delay, power consumption, migration cost, total cost with different ω.	–
Our work	Multi-access edge computing	the migration reward	The success rate of recovering different types of failures, the convergence in four network situation, state value, e2e delay	Applications in a container

Table 1 (*continued*)

Study	Environment	Objective	Performance metrics	Applications
			affected by migration threshold, percentage of successful migration, interruption time, standard deviation of available capacity on edge servers, total cost	

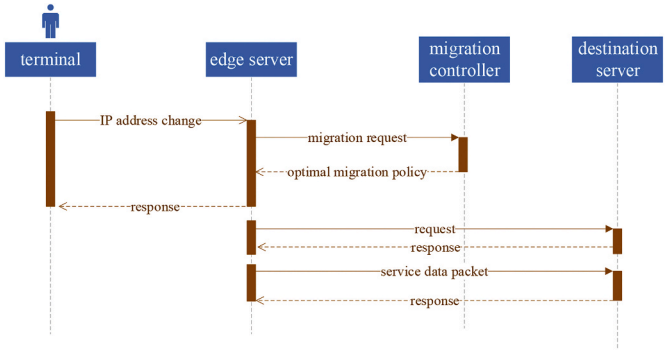


Fig. 2. Sequence diagram of service migration.

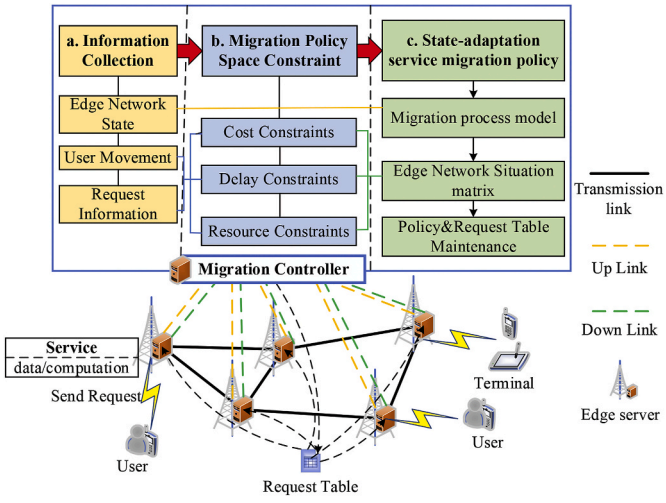


Fig. 3. Modular model of the service migration policy within SRSM.

request table maintenance is recording a table in the migration controller, and as a result we do not need to consider it in addition. We will introduce the three parts in the next sections.

3.2. Information collection model

3.2.1. Edge network state

The notations used in this work are summarized in Table 2. The user's terminal sends the migration request to the offloaded edge server, which is defined as the original edge node (ON). And the edge network can be modeled as an undirected connected graph G=(V,E) (Yuan et al., 2020), in which V means the edge nodes in the edge network and E

Table 2
The meaning of notations in our paper.

Notations	Meaning of notations
c_1	The available capacity of the edge nodes.
c_2	The computing ability of the edge nodes.
HC_e	The handling capacity of the link e between edge node m and n.
LB_e	The link bandwidth of the link e between edge node m and n.
d_n / d_e	The ability degree of edge nodes or the transmission links to represent the situation of the edge network quantitatively.
N_{OD}	The number of shortest paths between ON and DN.
$N_{OD}(e_{m,n})$	The number of shortest paths passing through the link $e_{m,n}$.
θ	The included angle between $\overrightarrow{v_t}$ and $\overrightarrow{v_{t+1}}$.
$\mathbb{S}^j$	The state set which is defined as the hops between ON and IN j.
s_i	The value of specific state.
$\mathbb{A}_{i,i+1}^{jj'}$	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'.
R_t	The reward at timeslot t.
Ex	The maximal experience the user can enjoy, and it is a decreasing function of dis_{ij}.
dis_{ij}	The distance between user i and the offloaded server j.
$cost_{com}$	The communication cost during the migrating process.
$cost_{com}$	The migration cost during the migrating process.
$\widetilde{r}$	The certain request contained within the request set $\mathscr{R}$ from the user.
$\mathbb{P}^{\widetilde{r}}$	The migration policy space including DN space n^* and the migration path space $\mathscr{P}$ for all edge nodes and request set.
$\mathbf{p}(l_{t+1}\|l_t)$	The probability that the user moves from l_t to l_{t+1}
$D(t)$	The dynamic value
$Q_t(s,a)$	The action value function of state s and action a at timeslot t by the iteration of the value function $Q_{t-1}(s,a)$

means the links among the servers.

The model for the edge nodes in an edge network is relevant to the available capacity and computing ability. We use a two-tuples to define the edge nodes: $E_n = (c_1, c_2)$. The service components and data packets are transferred by links among edge nodes. We focus on the handling capacity and link bandwidth to model the links: $E_l = (HC, LB)$.

Some edge nodes and links are not available due to insufficient energy or damage to the edge network, so the migration process cannot go on normally. Therefore, we design a normal situation and three failure conditions to analyze the service migration effects in these four cases.

In the process of migration, the intermediate edge nodes (INs) or the destination node (DN) can be occupied by other services in total. We define this case as node failure. Similarly, the links between the edge nodes could be out of order, and thus, we must find another feasible path to ensure the service migration (Talaat et al., 2020). We define this case as link failure. As shown in Fig. 4, we will analyze the following situations.

1) *Normal Migration:* Each node has a different ability degree, defined as d_n. (Dhakad and Bisen, 2016). We also define $c_{1,max}$, $c_{2,min}$ and $c_{2,max}$ to represent the maximal capacity and limited computing ability of each node. Here, d_n is considered to be a random variable that follows a uniform distribution which can be formalized according to Eq. (1) below:

$$d_n = 0.5 * \left[c_1 + (c_{1,max} - c_1)(1-\alpha)\right] + 0.5 * \left\{(1-\beta)\left[c_2 + (c_{2,max} - c_2)(1-\alpha)\right] + \beta\left[c_2 - (c_2 - c_{2,min})(1-\alpha)\right]\right\} \quad (1)$$

Similarly, we can obtain the link ability degree as shown in Eq. (2):

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

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

2) *Node Failure:* In this case, the available capacity or computation ability of some INs is occupied by other services (Jha et al., 2019). It is formalized as shown in Eq. (3):

$$d_n = 0.5 * c_1 + 0.5 * \left\{(1-\beta)c_2 + \beta\left[c_1 - (c_2 - c_{2,max})(1-\alpha)\right]\right\} \quad (3)$$

where $c_1 = c_{1,max}$, $c_2 = c_{2,max}$, $\beta \in [0,1]$ and $\alpha \in [0,1]$.

3) *Link Failure:* Link failure means that the link with a lower stability is likely to break down (Jia et al., 2018). There are two methods to describe the stability of the links in the edge network. The first method is the product of the two nodes' ability degrees. The second method is the betweenness centrality. The stability degree s_e of link $e_{m,n}$ is be defined in Eq. (4), as follows (Jia et al., 2018):

$$s_e \triangleq \sum_{O \neq D} \frac{\sigma_{OD}(e_{m,n})}{\sigma_{OD}} \quad , \quad \forall e_{m,n} \in E \quad (4)$$

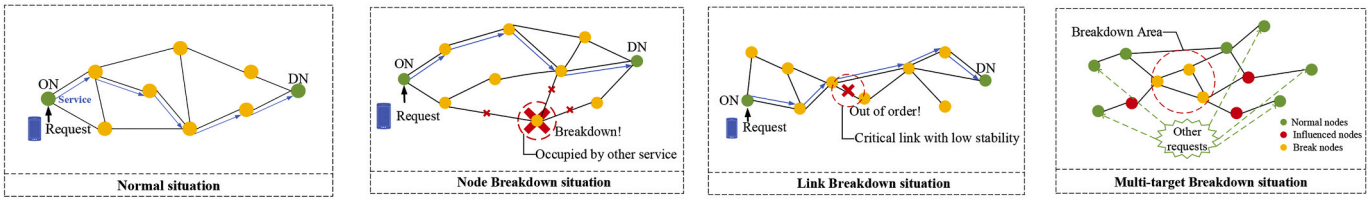
where N_{OD} is the number of shortest paths between ON and DN and $N_{OD}(e_{m,n})$ is the number of shortest paths that pass through edge $e_{m,n}$ respectively.

4) *Multitarget Coordinated Failure:* Multitarget coordi-nated failure sends enough traffic to nodes or links that surround the break area, which is defined as a set $\mathscr{S} \triangleq \{b_{n_1}, b_{n_2}, \ldots, b_{n_m} | n_m \in \mathrm{V}\}$ (Sasithong et al., 2019). There exist cut edges defined as $\mathscr{K} \triangleq (\mathscr{S}, \overline{\mathscr{S}})$ and the capacity of the cut edges is defined as cap $\mathscr{K} \triangleq \mathrm{c}(\mathscr{S}, \overline{\mathscr{S}}) = \sum_{e \in \mathscr{K}} c(e)$ (Sasithong et al., 2019). The controller identifies indirect break nodes that belong to set $\overline{\mathscr{S}}$ that are connected to the break area. For this condition, we use candidate nodes to ensure the connections of the network.

3.2.2. User movement model

Most migration mechanisms decide when to migrate by relying only on the network conditions. Few of them account for the user behavior (Rosário et al., 2018). However, deciding when to migrate according to the user's behavior and mobility has a large influence on improving the user's experience and allocating the resources on edge servers (see Fig. 5).

At timeslot t, the user's movement is defined as $\overrightarrow{v_t}$, including the direction and speed. We define the movement at the previous timeslot as



(a) (b) (c) (d)

Fig. 4. Four edge network situations.(a) Normal situation; (b) Node failure situation; (c) Link failure situation; (d) Multi-target failure situation.

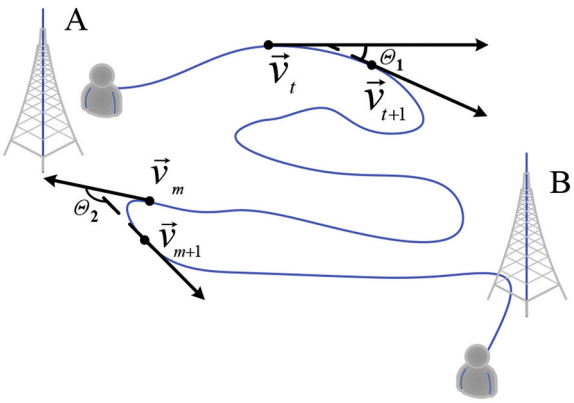


Fig. 5. User movement.

$\overrightarrow{v_{t-1}}$. To know the tendencies of the user's movements, we define $\theta = \langle \overrightarrow{v_t}, \overrightarrow{v_{t-1}} \rangle$ as the users' moving direction. Here, $\cos\theta \in [0,1]$ means that the user moves forward. In constrast, $\cos\theta \in [-1,0]$ means that the user moves backward.

3.2.3. Request information

We define the requests set as $\mathscr{R}$ and use the multidimensional vector $\mathbb{P}^{\widetilde{r}} = \{p_j^{\widetilde{r}} | \forall j \in \mathbb{V}, \forall \widetilde{r} \in \mathscr{R}\}$ to describe the edge node j, while the request $\widetilde{r}$ is selected as p_j equal to 0 or 1, which means that if the data request passes by the edge node j, p_j equals 1; otherwise, p_j equals 0. Therefore, the migration policy space of a specific request can be denoted as $\{p_1^{\widetilde{r}}, p_2^{\widetilde{r}}, \ldots, p_n^{\widetilde{r}} | \forall \widetilde{r} \in \mathscr{R}\}$. In addition, in the migration policy candidate space, we represent the DN space, and the migration path space as n^* and $\mathscr{P}$ respectively. The space of the migration path can be precalculated in the migration policy of each edge node with the assumption that the network will not oscillate violently.

3.3. Migration policy space constraint model

A request from ON to DN with specific constraints can be formalized as follows:

3.3.1. Cost constraints

Migrating the service to another edge server will bring about a migration cost for the data and a communications cost for the link. Because the cost function is more than linear to the distance, we use an exponential function of the distance to denote the cost function as Eqs. (5) and (6) show (Lee et al., 2018). This function can make it easier to learn the optimal solution and obtain an effective policy.

$$cost_{com}(dis_{oj}) = \begin{cases} \phi_c + \phi_l \theta^{dis_{oj}}, & if\ dis_{oj} > 0 \\ 0, & others \end{cases} \tag{5}$$

$$cost_{mig}(dis_{oj}) = \begin{cases} \delta_c + \delta_l \mu^{dis_{oj}}, & if\ dis_{oj} > 0 \\ 0, & others \end{cases} \tag{6}$$

where dis_{oj} means the distance between the original node and the node j. In addition, the parameters ϕ_c, ϕ_l, θ, δ_c, δ_l and μ are the real values. To guarantee that $cost_{com}$ and $cost_{mig}$ are increasing functions of dis_{oj}, we set the cons-traints as follows:

$$0 \le \theta \le 1, \quad \phi_l \le 0 \quad \phi_c \ge -\phi_l$$

$$\mu \ge 1, \quad \delta_l \ge 0, \quad \delta_c \in R$$

The demand for the cost varies for the different services. We define $q = cost_{com}/cost_{mig}$ to represent the service type. Here, $q \in (0,1)$ means that the request $\widetilde{r}$ is a data-sensitive service, and $q \in (1,\infty)$ means that the request $\widetilde{r}$ is a delay-sensitive service.

3.3.2. Delay constraints

Communication delay includes the transmission delay $t_t = data/v_t$ (Koyasako et al., 2020) and propagation delay $t_p = dis_{ij}/v_p$ (Koyasako et al., 2020). Here, $data$ means the length of the data frame in bits, and v_t means the transmission rate in the network adapter. Additionally, v_p means the propagation rate in fiber (Koyasako et al., 2020), and dis_{ij} means the distance between the edge servers i and j. The difference between t_t and t_p is that t_t occurs in the network adapter when the host sends the data frame, but t_p occurs in the transmission channels which is relevant to the distance and the transmission medium. We use W_{od} to denote the delay constraints.

$$\sum_{i \in V} s_j^{\widetilde{r}} t_t + \sum_{i \in V} \sum_{j \in \mathscr{L}} s_j^{\widetilde{r}} s_{j'}^{\widetilde{r}} t_p \le W_{od}, \quad \forall \widetilde{r} \in \mathscr{R} \tag{7}$$

Here, $\mathscr{L}$ is the set of neighbors close to node j. The hop SMT formalization can be described as $\sum_{i \in V} s_j^{\widetilde{r}} \le H$, which means that the total hop is smaller than threshold H as described in Eq. (12).

3.3.3. Available resources capacity constraints

The resource capacity includes the node and link capacities, and we discuss them separately.

1) DN capacity constraints

Migration paths should not include those edge nodes that have unavailable resources for extra requests, as shown in Eq. (8) (Nawrocki and Sniezynski, 2018).

$$c_{1,j}^{max} - \sum_{\widetilde{r} \in \mathscr{R}} s_j^{\widetilde{r}} c_j^{\widetilde{r}} \ge c_{1,j}^{th} \quad , \quad \forall \mathrm{j} \in \mathrm{V} \tag{8}$$

where AC_j^{max} is the max capacity for node j, $c_j^{\widetilde{r}}$ denotes the cost at node j for request $\widetilde{r}$, and AC_j^{th} is the minimum threshold for node j. Here, $c_j^{\widetilde{r}}$ can be described as:

$$c_j^{\widetilde{r}} = c_{1,j} \left(\xi^{1 - \frac{c_{1,j}(j')}{c_{1,j}}} - 1 \right) \forall \mathrm{j} \in \mathrm{V} \quad , \quad \forall \widetilde{r} \in \mathscr{R} \tag{9}$$

where $c_{1,j}$ is the available capacity, and ξ is a parameter that is set to $2n$, where n is the total number of edge nodes.

2) Link capacity constraints

Edge servers connect with each other through transmission links. To avoid a situation in which the link capacity is not sufficient to transmit the data packet, we define $E_{e(j,j')}^{\widetilde{r}}$ to denote whether link e(j, j′) is adopted to be migrated as shown in Eq. (10):

$$E_{e(j,j')}^{\widetilde{r}} = \begin{cases} 1, & if\ link\ e(\mathrm{j}, \mathrm{j}')\ is\ adopted\ to\ be\ migrated \\ 0, & otherwise \end{cases} \tag{10}$$

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

$$v \times \left(1 - E_{e(j,j')}^{\widetilde{r}}\right) \le N_{jj'}, \quad j \in V, \quad e \in E \tag{11}$$

3.4. State adaptation-based service migration policy model

3.4.1. Service migration process model

1) Whether to migrate the service:

As the user moves away from ON, the communication delay and loading of the links will increase. It will cause interruption of the service, which affects the user's experience and the performance of the edge server. Therefore, we define a variable H to describe the maximum hops within the area of one cell as shown in Eq. (12):

$$\begin{cases} h_{mn} > H, & must\ migrate\ service \\ h_{mn} \le H, & migration\ decision\ by\ SRSM \end{cases} \tag{12}$$

where h_{mn} means the hops between the user and the offloaded server.

2) How to migrate the service

We propose a reinforcement learning-based service migration 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 (see Fig. 6).where E is the maximal experience that the user can enjoy, and it is a decreasing function of dis_{ij}^{t}, which means that $e^{ij}(s_t)$ declines when the user moves far away from the offloaded edge server. For the cost during the migration process, we define it as $c_t(s_t, a_t) = cost_{com}(dis_{oj}) + cost_{mig}(dis_{oj})$, which is explained in Eqs. (5) and (6). Therefore, the reward is shown in Eq. (14).

$$r_t(s_t, a_t) = e^{ij}(s_t) - c_t(s_t, a_t) \tag{14}$$

3.4.2. Edge network situation matrix

The migration controller needs to obtain the cost $C_t = \{c_t(s_t, a_t) | \forall j \in V, t \in N^+\}$ of migrating the service, which is described in Eqs. (5) and (6). If node j breaks down, the negative effect is defined as $G_t^b = \{g_j^b(t) \,|\, \forall j \in V, t \in N^+\}$. If node j is available, then the reward is defined as $R_t^e = \{r_j^e(t) \,|\, \forall j \in V, t \in N^+\}$, which is s hown in Eq. (14). Thus, the edge network state matrix is represented as follows:

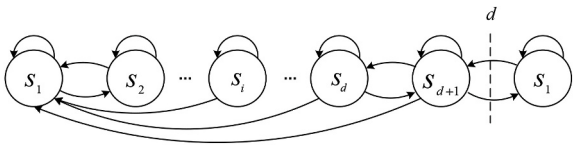


Fig. 6. State Transmission diagram.

Definition 1 State. In a practical scenario, the edge server has a certain coverage (Li et al., 2019). We define the hops between the ON and IN j as state function $S^j(t)$. To represent the state set, we assume that the edge server connected to the user initially is the observation point **O**. In this way, we obtain state set $\mathbb{S}^j = \{s_i = 0, 1, 2, \ldots, d | d \le D, j \in V, i \in [1, d+1]\}$.

Definition 2 Action. We define state s_i; the migration controller takes the action a_i^j, and the service is migrated from edge server j to j', which can be represented as $\mathbb{A} = \{A_{i,i+1}^{j,j'} \,|\, \text{edge server} : j \to j',\ \text{state} : s_i \to s_{i+1}\}$. All of the act-ions should be connected to the edge server that is serving for the user at state s_i. (Tang et al., 2019)

Definition 3 Reward. We described 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', then $R_t(s, a)$ will be given negative values that are linearly related to the costs. On the other hand, if the service can be migrated to server j' uninterruptedly, R_t will be given a fixed positive value. In this way, the reward is defined as the difference between the experience of the user and the migration cost. (Chu et al., 2020) The experience of the user can be replaced by the response delay determined by dis_{ij}^{t}, which is the distance from user i to the offloaded server j.

$$e^{ij}(s_t) = Ex - \lambda dis_{ij}^{t} \tag{13}$$

$$X_{\mathscr{L}\times\mathscr{J}} \triangleq \begin{cases} x_{ij} = -g_j^b(t), & if\ node\ j\ breaks\ down \\ x_{ij} = r_j^e(t), & if\ node\ j\ is\ avaliable \\ x_{ij} = 0, & otherwise \end{cases} \tag{15}$$

where $\mathscr{L}$ denotes the number of time slots and $\mathscr{J}$ denotes the number of edge nodes. The state value of timeslot t_o is described in Eq. (16):

$$V(t_o) = \sum_{t=1}^{t_0} \left[\sum_{i \in \mathscr{L}, j \in \mathscr{J}} x_{ij} + \sum_{m \in \mathscr{M}} c_t^m(t) - c_t \right] \tag{16}$$

Here, $\mathscr{M}$ is the set of broken nodes. We use the negative derivative of V to represent the network states in Eq. (17):

$$K = -V'(t_0) = -\lim_{\Delta t \to 0} \frac{V(t_0 + \Delta t) - V(t_0)}{\Delta t} \tag{17}$$

We call the trade-off weight between the new return value and old return value as the learning rate α shown in Eq. (18). When the edge network breaks down, α should be closer to 1, which focuses more on the new return value. Otherwise, α should be closer to 0, which focuses more on the old return value when the edge network is normal.

$$\alpha(K, \tau) \triangleq \frac{1}{(1 + e^{-K})\tau} \tag{18}$$

Here τ is called the time factor, which means that its value will increase when a set number of time slots pass.

4. Service migration policy based on state-adaptation deep Q-learning algorithm

In this section, we introduce the learning process based on the value iteration method to find the optimal migration destination server and path, to reduce the migration delay and cost. In addition, the convergence and the computing complexity of the proposed algorithm should also be improved to optimize the proposed solution.

4.1. State adaptation-based deep Q-learning algorithm

The aim of the 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. Policy is defined as the action sequences through migrating at each state, and the details are shown as follows.

Definition 4 The migration controller's policy (Dhakad and Bisen, 2016) is about taking actions at state s_t from the timeslot $t \in [0, \infty]$, which can be defined in Eq. (19) as follows:

$$\pi = \{\varphi_t(s_t, a_t) | \forall t \in [1, \infty)\} \tag{19}$$

where $\varphi_t(s_t, a_t)$ is a mapping function from one state s_t to one action a_t. The goal of the migration controller is to maximize the total expected reward with an optimal policy π^*.

$$\pi^* = \arg\left\{ \max_{\pi \in \Pi} E_{s_t}^{\pi} \left[\sum_{t=1}^{\infty} \gamma r_t(s_t, a_t) \right] \right\} \tag{20}$$

During the state iteration process, the system updates the state value at each step of the exploration. The update formula uses the Bellman equation, given in Eq. (21) (Yoshida et al., 2013).

$$Q_t^*(s_t, a_t) = E_{s_{t+1}} \left[r_t(s_t, a_t) + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1}) \right] \tag{21}$$

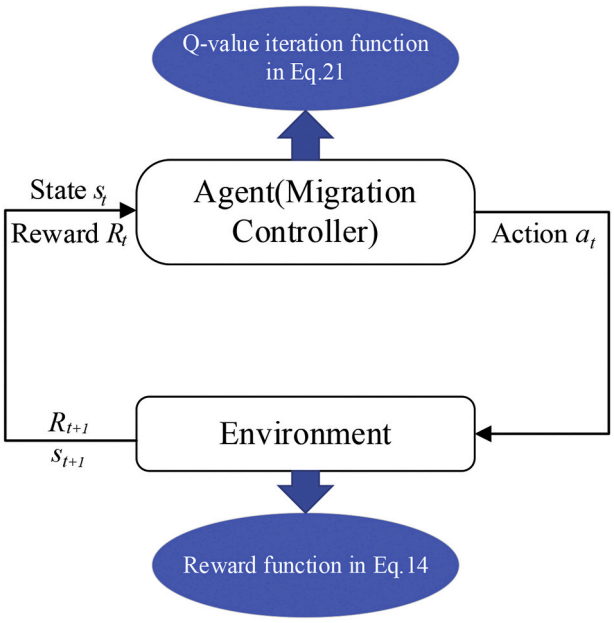


Fig. 7. The process of Q-value iteration.

where γ represents a discount factor in the interval (0,1). Q-value Q(s, a) is a function that predicts the expected long-term reward of taking action a in state s by using a deep function approximator. Additionally, the optimal policy corresponds to the $Q_t^*(s_t, a_t, \theta)$ value which is defined in Eq. (22).

$$\varphi_t^* = \underset{a_t \in \mathbb{A}}{\operatorname{argmax}} Q_t^*(s_t, a_t, \theta) \tag{22}$$

As shown in Fig. 7, firstly, the environment will give the original state (os). The agent will obtain all Q (s, a) in Eq. (21) of os according to the value function network, and then, we use ε−greedy to select the action and make a decision. The environment will give a reward and the next state after receiving this action. After one step, we update the parameters of the value function network according to the reward function in Eq. (14) 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 that corresponds to the maximum Q-value from the calculated Q-values in the next timeslot with a probability of 1- ε, as shown in Eq. (18). Here, p is a random number that belongs to [0,1]. In our experiment, we set ε as 0.1 (Saurez et al., 2016).

$$\varphi_t^* = \begin{cases} \underset{a_t \in \mathbb{A}}{\operatorname{argmax}} Q_t^*(s_t, a_t, \theta), & p = 1 - \varepsilon \\ a \sim U(a), & p = \varepsilon \end{cases} \tag{23}$$

Our algorithm is based on DQN to address the problem defined in Eq. (20); it can help the migration controller generalize its past experience to predict the Q-value about the unexplored states. In addition, to analyze the effect on the convergence rate of the different network states, we redefine the learning rate function $\alpha(t)$, which is shown in Eq. (18) and is based on the awareness of the network state.

Algorithm 1
State-Adaptation DQN Algorithm for Service Migration Policy

Input: State/reward Matrix (R) is a $n \times m$, 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

(*continued on next column*)

Algorithm 1 (*continued*)

2. Initialize Q function with random θ
3. Initialize $\widehat{\mathrm{Q}}$ function with parameters θ^-
4. **for** each episode **do**
5. $t \leftarrow 1$; $s_1 \leftarrow s_i \in \mathbb{S}$.
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 Eq. (23)
9. $s_{t+1} \leftarrow s_t$
10. Calculate $r_t(s_t, a_t)$ by Eq. (14).
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 $y_i = \begin{cases} r_j, & \text{if } j+1 \text{ is the last state} \\ r_j + \gamma \max_{a_t \in \mathbb{A}} Q_t(s_{j+1}, a_{j+1}, \theta_j^-), & \text{otherwise} \end{cases}$
14. Execute a gradient descent step on $(y_i - Q_t(s_t, a_t, \theta))^2$ by θ.
15. Reset $\widehat{\mathrm{Q}} = Q$ in every C steps
16. $t \leftarrow t + 1$
17. **end for**
18. **end for**
19. **return** P

Algorithm 1 (Table Algorithm 1) shows the pseudo code of the state-awareness DQN. When the first service migration request is sent to the offloaded edge server, the migration controller explores different server states without prior knowledge. By using the above ε-greedy policy, the exploration starts until the Q-value converge to the optimal value, which is called an episode. The input matrix R is called the immediate reward matrix to represent the action reward value from s_t to t he next s_{t+1}. The output matrix P is initialized to zeros. On each episode, the migration controller adopts the ε-greedy policy to select the migration actions and observes the reward to update the matrix P. As indicated from line 12, the experience of the migration controller's (s_{t+1}, a_t, r_t, s_t) is reserved in the memory, thus without needing the transition probabilities (YE and Zhang, 2019).

4.2. Analysis

In this section, we will analyze the convergence and the computing complexity of the proposed algorithm. Our proposed algorithm is based on Q-learning, which is a type of value iteration algorithm. Thus, we focus on the iteration process of our algorithm, with the consideration of the edge network situation matrix in Eq. (15).

First, we will prove the convergence of the proposed algorithm. In value iteration, the action value function Q_t in Eq. (21) and the update process after the state change is shown in Eq. (24).

$$TD = R_{t+1} + \gamma \max_{a_{t+1}} Q_{t+1}(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \tag{24}$$

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha * TD \tag{25}$$

Here, *TD* means Temporal Difference-error. In Jaakkola et al. (1994), the writers have proven that Q_t converges to the optimal value Q^* with probability 1. Considering that the migration controller does not know which failure situation it is, we propose a state-adaptation Q-learning algorithm, in which the learning rate α changes according to the current edge network failure situation. Thus, the objective function is described in Eq. (26) as follows:

$$Q^{\pi}(s_t, a_t) = E_{\pi}\left[R_{t+1} + \sum_{k=1}^{\infty}\left[\prod_{i=0}^{k-1}\gamma\right] R_{t+k+1}\right] \tag{26}$$

Here, $\gamma \in [0, 1]$ is the discounting factor. Variate i means a symbol for recording the times of γ, and it ranges from 0 to $k-1$. In this way, Eq. (25) can be rewritten in Eqs. (27) and (28), below:

$$TD_t = R_t + \gamma \max_{a' \in \mathbb{A}} Q_t(s_t, a') - Q_t(s_t, a_t) \tag{27}$$

$$Q_{t+1}(s_t,a_t)=Q_t(s_t,a_t)+\alpha * TD=(1-\alpha_t(s_t,a_t))Q_t(s_t,a_t) + \alpha_t(s_t,a_t)\left[R_t+\gamma\max_{a'\in\mathbb{A}}Q_t(s_t,a')\right] \tag{28}$$

where $\alpha_t(s_t,a_t)$ is the learning rate which is dependent on the current network states. To show the convergence of the optimal action value function $Q^{\pi}(s_t,a_t)$, we adopt the convergence theorem of the stochastic sequence from a previous paper (Rosário et al., 2018).

Theorem 1. The proposed SRSM algorithm can converge to the optimal action value with probability 1 (Rosário et al., 2018).

Proof. The random iteration process I_t takes the values in R_n and is defined as follows:

$$I_{t+1}(s)=(1-\alpha_t(s))I_t(s)+\alpha_t G_t(s) \tag{29}$$

which converges to 0 with a probability under the following assumptions:

(1) $0\le\alpha_t\le 1,\sum_{t=0}^{\infty}\alpha_t(s)=\infty,\sum_{t=0}^{\infty}\alpha_t^2(s)<\infty$
(2) $\|E[F(s)|F_t]\|_W\le\gamma\|I_t\|_{W'}\ \gamma<1$
(3) $\overline{[F(s)|F_t]}\le C(1+\|I_t\|_W^2)$, for $C>0$

From Eq. (21), we can obtain $Q^*(s_t,a_t)$ which is connected to the optimal migration policy. Additionally, we define $I_{t+1}(s,a)=Q_t(s_t,a_t)-Q^*(s_t,a_t)$, $G_t(s,a)=R_t+\gamma\max_{a'\in\mathbb{A}}Q_t(s_t,a')-Q^*(s_t,a_t)$. Then we can have another formulation to represent the random process in Theorem 1.

$$I_{t+1}(s)=(1-\alpha_t(s_t,a_t))I_t(s,a)+\alpha_t(s_t,a_t)G_t(s,a) \tag{30}$$

Next, we will prove that SRSM algorithm satisfies the above three assumptions. In our algorithm, $r_t(s_t,a_t)$ is bonded, and all of the state-action spaces are visited infinitely. The value of the sigmoid function $\frac{1}{(1+e^{-K^*})}$ is a constant Ψ that is between 0 and 1. From Eq. (23), we can obtain the expansion equation (Rosário et al., 2018) of $\sum_{t=0}^{\infty}\alpha_t(s)$ as follows:

$$\sum_{t=0}^{\infty}\alpha_t(s)=\frac{1}{(1+e^{-K_1})}+\frac{1}{2(1+e^{-K_2})}+\frac{1}{3(1+e^{-K_3})} +\ldots+\frac{1}{\tau^*(1+e^{-K^*})}+\frac{1}{(\tau^*+1)(1+e^{-K^*})} +\ldots \tag{31}$$

Here, τ^* means the equilibrium beginning time. From Eq. (22), we can obtain that the sum of the 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 Eq. (31) as $\sum_{t=0}^{\infty}\alpha_t(s)=\Omega_1+F_1$. F_1 can be expanded as follows:

$$F_1=\frac{1}{(1+e^{-K^*})}\left(\frac{1}{\tau^*}+\frac{1}{\tau^*+1}+\ldots\right)=\frac{1}{(1+e^{-K^*})}\left(1+\frac{1}{2}+\frac{1}{3}+\ldots\right)-\frac{1}{(1+e^{-K^*})}\left(1+\frac{1}{2}+\ldots+\frac{1}{\tau^*-1}\right)=\frac{1}{(1+e^{-K^*})}\sum_{\tau=1}^{\infty}\frac{1}{\tau}-\Omega_2=\Psi\sum_{\tau=1}^{\infty}\frac{1}{\tau}-\Omega_2 \tag{32}$$

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. Next, we need to prove the divergency of the harmonic series $\sum_{\tau=1}^{\infty}\frac{1}{\tau}$ using a hypothetic Proof.

First, we assume that the harmonic series converges, and thus we have

$$\lim_{n\to\infty}(S_{2n}-S_n)=\lim_{n\to\infty}\left(\sum_{\tau=1}^{2n}\frac{1}{\tau}-\sum_{\tau=1}^{n}\frac{1}{\tau}\right)=0 \tag{33}$$

However, we also have

$$S_{2\tau}-S_{\tau}=\frac{1}{\tau+1}+\frac{1}{\tau+2}+\frac{1}{\tau+3}+\ldots+\frac{1}{2\tau}>\frac{\tau}{2\tau}=\frac{1}{2}$$

It is obvious that the assumption in Eq. (32) does not work, which means that $\lim_{n\to\infty}\left(\sum_{\tau=1}^{2n}\frac{1}{\tau}-\sum_{\tau=1}^{n}\frac{1}{\tau}\right)=\infty$, more specifically, $\sum_{t=0}^{\infty}\alpha_t(s)=\Omega_1+\Psi\sum_{\tau=1}^{\infty}\frac{1}{\tau}-\Omega_2=\infty$.

We can obtain the expansion function of $\sum_{t=0}^{\infty}\alpha_t^2(s)$ which is similar to the above Proof process:

$$\begin{aligned}\sum_{t=0}^{\infty}\alpha_t^2(s)&=\frac{1}{(1+e^{-K_1})^2}+\ldots+\frac{1}{\tau^{*2}(1+e^{-K^*})^2}+\ldots\\&=\Omega_3+\frac{1}{(1+e^{-K^*})^2}\left(\frac{1}{\tau^{*2}}+\frac{1}{(\tau^*+1)^2}+\ldots\right)\\&=\Omega_3+\frac{1}{(1+e^{-K^*})^2}\left(1+\frac{1}{4}+\frac{1}{9}+\ldots\right)\\&\quad-\frac{1}{(1+e^{-K^*})^2}\left(1+\frac{1}{4}+\ldots+\frac{1}{(\tau^*-1)^2}\right)\\&=\Omega_3-\Omega_4+\Psi^2\sum_{\tau=1}^{\infty}\frac{1}{\tau^2}\end{aligned} \tag{34}$$

The series $\sum_{\tau=1}^{\infty}\frac{1}{\tau^2}$ obviously converges from our advanced mathematics knowledge. Thus, $\sum_{t=0}^{\infty}\alpha_t^2(s)<\infty$.

Next, we can know that the learning rate function satisfies the following conditions:

$$0<\alpha_t(s)\le\alpha_{max}\le 1,\sum_{t=0}^{\infty}\alpha_t(s)=\infty,\sum_{t=0}^{\infty}\alpha_t^2(s)<\infty \tag{35}$$

Then, we need to prove that the random process satisfies assumptions (2) and (3):

$$
\begin{aligned}
E[F(s)|F_t] &= \sum_{s' \in \mathbb{S}} P^a_{s,s'} \left[R(s,a) + \gamma \max_{a' \in \mathbb{A}} Q_t(s_t, a') - Q^*(s_t, a_t) \right] \\
&= \sum_{s' \in \mathbb{S}} P^a_{s,s'} \left[R(s,a) + \gamma \max_{a' \in \mathbb{A}} Q_t(s_t, a') \right] - \sum_{s' \in \mathbb{S}} P^a_{s,s'} Q^*(s_t, a_t) \\
&= \sum_{s' \in \mathbb{S}} P^a_{s,s'} \left[R(s,a) + \gamma \max_{a' \in \mathbb{A}} Q_t(s_t, a') \right] - Q^*(s_t, a_t) \\
&= \sum_{s' \in \mathbb{S}} P^a_{s,s'} \left[R(s,a) + \gamma \max_{a' \in \mathbb{A}} Q_t(s_t, a') \right] - \sum_{s' \in \mathbb{S}} P^a_{s,s'} \left[R(s,a) + \gamma \max_{a' \in \mathbb{A}} Q_t^*(s_t, a) \right] \\
&\leq \gamma Q_t - Q_t^*{}_{\infty} = \gamma I_{t\infty}
\end{aligned}
\tag{36}
$$

where we have the condition that $\sum_{s' \in \mathbb{S}} P^a_{s,s'} = 1$.

$$
\overline{[F(s)|F_t]} = \overline{\left[R(s,a) + \gamma \max_{a' \in \mathbb{A}} Q_t(s_{t+1}, a') \middle| F_t \right]} \leq C\left(1 + I_t^2{}_{\infty}\right) \tag{37}
$$

Here, $R(s,a)$ is bounded and $\gamma \in [0,1]$. C is a constant and more details can be seen in Dhakad and Bisen (2016)). Thus, $I_{t+1}(s,a) = Q_t(s_t, a_t) - Q^*(s_t, a_t)$ will converge to zero which implies that Q^* will be the optimal and stable value.

Next, we will apply the method in Pelamatti et al. (2020) to improve the rate of convergence. At each SOMVSP iteration, the relative performance of each subproblem q with respect to the others must be determined Pelamatti et al. (2020). To do so, three different predicted optima are computed for each sub-problem (in other words, the cost subproblem, delay problem and available resources subproblem) by considering different confidence interval scenarios: the best case (BC), the worst case (WC) and the nominal case (NC). In practice, these scenarios correspond to the predicted feasible optimum value of the subproblems by considering an optimistic, pessimistic and null value of the predicted variance respectively, and they are defined as follows:

$$
\begin{aligned}
&BC = min \widehat{y_q}(x, z, w_q) - a \cdot \widehat{s_q}(x, z, w_q) \\
&s.t. EV\left(g_c(x, z, w_q)\right) < t_c \ \ for \ \ c = 1, \ldots, n_{g_i}(w_q)
\end{aligned}
\tag{38}
$$

$$
\begin{aligned}
&WC = min \widehat{y_q}(x, z, w_q) + a \cdot \widehat{s_q}(x, z, w_q) \\
&s.t. EV\left(g_c(x, z, w_q)\right) < t_c \ \ for \ \ c = 1, \ldots, n_{g_i}(w_q)
\end{aligned}
\tag{39}
$$

$$
\begin{aligned}
&NC = min \widehat{y_q}(x, z, w_q) \\
&s.t. EV\left(g_c(x, z, w_q)\right) < t_c \ \ for \ \ c = 1, \ldots, n_{g_i}(w_q)
\end{aligned}
\tag{40}
$$

where $a \in \mathbb{R}^+$ is a tunable parameter that represents how conservative are the definitions of the BC and WC scenarios. Additionally, x means the continuous variables such as the delay in the subproblem; z means the discrete variables, such as the resource capacity; and w means the dimensional variables, such as the request information.

At every SOMVSP iteration, the discarding of nonoptimal subproblems is followed by the allocation of a different computational budget to each remaining subproblem. This action means that at every iteration, a budget of B_q data samples to be infilled is allocated to each remaining subproblem q. B_q is computed by accounting for both the predicted performance of a given subproblem as well as its dimension, and it is defined as shown in Eq. (41):

$$
B_q = d_q \left(\frac{1 + \Delta_q}{2} \right) \tag{41}
$$

where d_q is the total dimension of the subproblem q, while Δq is a term that represents the relative performance of the considered subproblem with respect to the remaining subproblems. It is computed as:

$$
\Delta_q = \frac{NC_{max} - NC_q}{NC_{max} - NC_{min}} \tag{42}
$$

where NC max and NC min are respectively the largest and lowest NC values among the remaining subproblems.

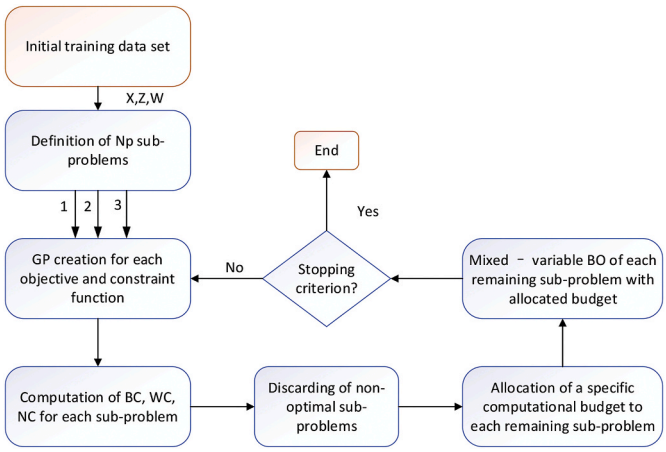


Fig. 8. The overview of the optimization process.

Following the computational budget allocation, each remaining subproblem is independently optimized by infilling a number of data samples proportional to the allocated budget. The newly infilled data sample for each subproblem is defined as follows:

$$
\begin{aligned}
&\{x^n, z^n\} = argmax(EI(x, z)) \\
&s.t. EV_i(x, z) \leq t_i \ \ for \ \ i = 1, .., n_g \\
&w.r.t. \quad x \in F_x, z \in F_z
\end{aligned}
\tag{43}
$$

Please note that in this case, x and z refer to continuous and discrete variables that the considered subproblem depends on. The overview of the optimization process is shown in Fig. 8.

Assuming that $|\mathbb{S}|$ is the number of network states, $|\mathbb{A}|$ is the number of migration engine's actions and $|T|$ is the number of time slots. Thus, the space complexity is $O(|\mathbb{S}||\mathbb{A}|)$.

Q-learning is a type of value iteration algorithm, for which the max time complexity is $O(|\mathbb{S}|^2|\mathbb{A}|)$. Considering the dynamic awareness method, the max time complexity of SRSM is $O(|\mathbb{S}|^2|\mathbb{A}||T|)$.

5. Performance evaluation

In this section, numeric analysis is conducted to validate the effect of the proposed SRSM algorithm. We compare the proposed algorithm with the following three other methods in the previous literature: (i) the method in which service migration based on multi-attribute (MADA) is applied (Zhao et al., 2017); (ii) the method in which the migration policy is based on deep Q-learning (DQN) is applied (Gao et al., 2019); (iii) the case when no service migration is applied (NSM). Please note that the MADA algorithm requires a transition probability. This method can

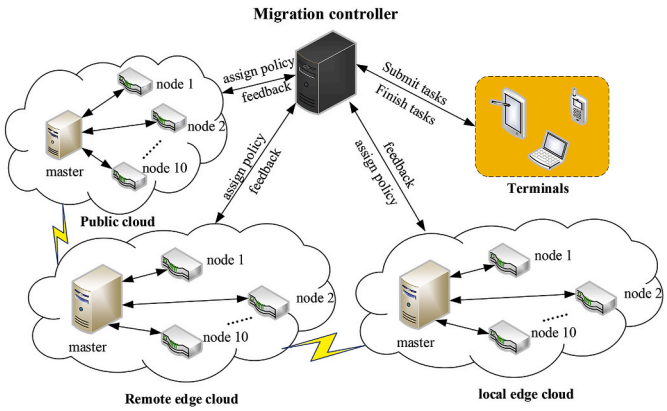


Fig. 9. The experimental environment of a mobile multi-access environment.

obtain the optimal policies if the transition probability is correct and limited. However, it is difficult to obtain the ground truth transition probability. We would use several "incorrect" transition probabilities because the user's ground truth transition probability usually contains noise. On the other hand, our deep Q-learning based algorithm has no requirement for the transition probability. In addition, our method can process the migration with a large scale for the state and action space.

In our algorithm, the observable state for the migration policy of the service request $\tilde{r}$ at edge server j comprises the location l_t, the hops between ON and user, the available resource capacity c_1 of the edge nodes and the currently used memory of all edge nodes, as well as the user's connected base station position, the previous edge node position, and the memory requirements of all service requests at the current edge node j.

5.1. Configurations

For the programming environment, we use the toolkit inside the TensorFlow learning framework based on python to finish the training process of our proposed algorithm and obtain the optimal destination edge server. In addition, we use the Z3 Solver (Singh et al., 2010) to solve the SMT constraints problem.

The simulation experiment is performed on Mininet-WiFi, which can simulate terminals, edge servers and the migration controller scenarios. The experiment environment of the edge network is shown in Fig. 9, which shows the logic connection of the components. The migration controller is the core component, and it has connections among the public cloud, local edge cloud, remote edge cloud and terminals. The terminals component and the controller exchange binary strings are to serve the users' demands. In the requesting frame, if the IP address of the terminals changes, the controller will start the service migration module and send resource requests to the edge cloud servers to obtain information on the available resources. Edge cloud and public cloud are composed of master servers and node servers, to realize the distribution of resources.

Edge servers are distributed in a 4×4 grid area according to a Poisson Distribution and the number of edge servers is 20. In the grid, each edge represents a road, which means that the terminal must move along these edges. To make our simulation results general, we follow the method in Hart et al. (1968); Sadilek and Krumm (2012) to plan a motion path for the purpose of traversing as many edge servers as possible ("traversal" refers to the fact that the user passes the coverage of an edge server). The position of the user's motion track is shown in Fig. 10.

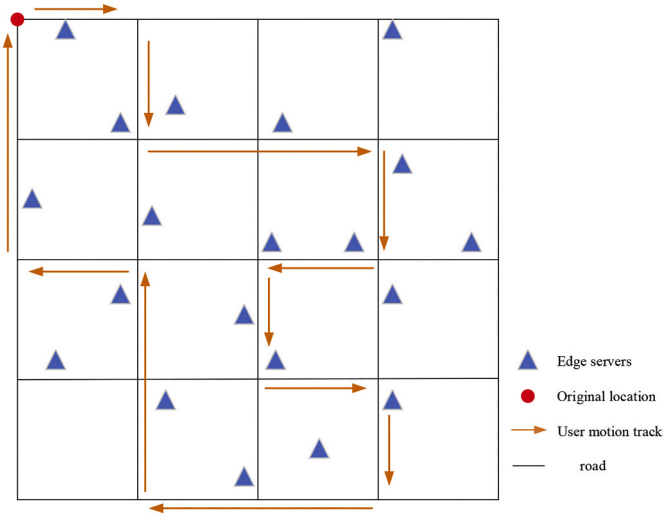


Fig. 10. The position and motion track of our simulation.

Table 3
Other parameters in the experiment.

Parameter meanings	Parameter values
Threshold that service migrates D	Hossain et al. (2020); Liu et al. (2020); MacHen et al. (2016); Roman et al. (2018); Wang et al. (2018a,b)
Communication cost ϕ_c, ϕ_l, θ	1, −1, 0.8
Migration cost δ_c, δ_l, μ	0.4, 0.1, 1.1
Requesting data packet data	1–64 M
Channel bandwidth BW	[100,300]M
Rate of data transmission v_t	100Mbit/s
Propagation rate in fiber v_p	2.0×10^5km/s
Discount factor γ	0.9

The timeslot of the recording data is set to 1s. Whenever a user connects to a new edge node, the placement of the connected service is re-evaluated and a migration decision is triggered. The link capacity $N_{j,j'}$ (in Bytes) between two MECs obeys a uniform distribution in the range (50, 300). We set delay constraints W_{od} for each session that obey a uniform distribution in the range (3,6). The MEC's capacity for each resource C_j^{th} (in MIPS) obeys a uniform distribution in the range (100, 400). Table 3 states the other parameters.

5.2. Experimental comparison

5.2.1. The success rate of recovering failures

Avoiding the effects of the node or link failures is one of the important elements during the service migration process. We simulated 2.3×10^5 time slots for every round and calculate the success rate of recovering the failures in every time slot for different schemes, including random route mutation (RRM)(Duan et al., 2013)、end-point route mutation (EPRM) (Rauf et al., 2016) and SRSM algorithms. The RRM method is to find an available migrating path randomly under different failures. The EPRM method can minimize the overlay among multiple streams to save the resources of links and nodes. As shown in Fig. 11(a), it can be seen that the RRM algorithm has the highest success rate, 23.3%, for recovering the link failure. Since the RRM algorithm is mainly aimed at the route mutation strategy of link failure in the network layer. The success rate of the other two schemes are approximately 19.6% and 18.5% respectively. The success rate of recovering multitarget failure is the lowest, because there are two types of failures, and thus, it is more complex to solve the problem with more uncontrollable factors. For the RRM algorithm, the success rate retain almost the same value over time and does not converge to a certain value, which indicates that the algorithm cannot overcome the influence of failures during the service migration process. Migration has no effect on the success rate of failure recovery. Similarly, as shown in Fig. 11(b), the change trends of the success rate of the different failures by EPRM algorithm are the same as by RRM algorithm.

As shown in Fig. 11(c), for the failure recovery of a link, the success rate increases from 5.5% to 23% with an increase of 17.5%. For node failure recovery, the success rate increases from 6.6% to 20.8% with an increase of 14.2%. For failure recovery of multitargets, the success rate increases from 8% to 18%. Thus, we know that the failure recovery effect for the link and node work well at the same time, because we account for the available resource capacity of the edge server and the bandwidth of the transmission link for the constraint elements. In this way, we can ensure that the algorithm does not cause failures again when it is executed, which can improve the success rate of the fault recovery. As shown in Fig. 11(d), in conclusion, with the comparison of the RRM and EPRM algorithms, the success rate can converge to a certain value in our proposed algorithm for three types of failures. Additionally, it can increase the success rate of the failure recovery by approximately 5%, effectively reducing the impact of the node and link

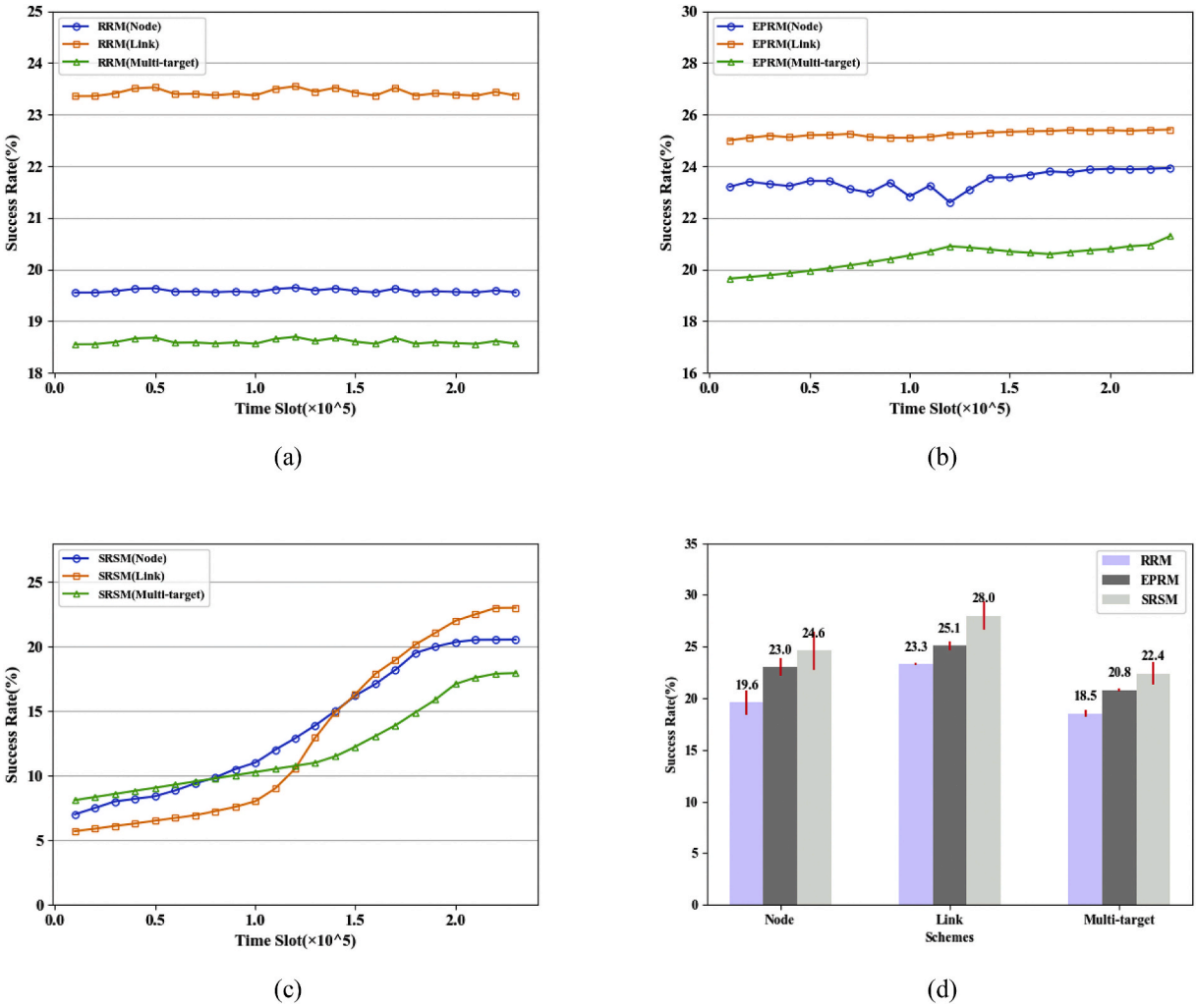


Fig. 11. The success rate of recovering different types of failures. (a) RRM algorithm; (b) EPRM algorithm; (c) SRSM algorithm; (d) comparison between three algorithms.

failures in the process of service migration.

5.2.2. *Convergence in the SRSM algorithm*

Fitness (Rosário et al., 2018) is incremental or cumulative and thus, it can be considered to be the cumulative update value of partial history. The definition of fitness (Rosário et al., 2018) (Lee et al., 2018) can be described as follows:

$$Fitness(r_a) = E_{e\sim\varepsilon}E_{h\sim\langle A(r_a,e)\rangle}\{F(h)\}$$

where E denotes the expectation operator and e is the environment which is randomly sampled from the whole possible environment ε. Here, $h = \{(s_0, a_0), (s_1, a_1), \ldots\}$ is the learning history generated by the reinforcement learning algorithm in the environment, and $F(h)$ is the evaluation measure function of the history data. However, in a practical scenario, the fitness can be obtained by using the average of the evaluation of the history data approximately, in other words, $Fitness(r_a) \approx \sum_{n=1}^{N}\{F(h_n)\}$.

First, as shown in Fig. 12, the blue curve represents the influence of the learning rate based on network dynamic awareness of the fitness in our proposed algorithm. The purple curve represents the case in which the learning rate is set to 0.9. In view of the four edge network situations, the learning rate based on the network dynamic awareness gradually converges to a certain value as the time slot goes by, but the purple curve represents that the fitness gradually increases as time passes by. This relationship indicates that the SRSM algorithm can make better use of the historical state-action (s, a) data set. The reason is that the learning rate in this solution is continuously adjusted according to the network states. Failures in the edge network will cause the matrix $X_{\mathcal{J}\times\mathcal{J}}$ to change, and the learning rate will be adjusted accordingly. In addition, it can be seen that under normal circumstances, at time slot 1.0×10^5, the fitness value has begun to be higher than 0.9. In the case of failures occurring, the occurrence of a stable fitness value takes place later, especially in the case where the failure type is multitarget. When the timeslot gradually reaches 1.5×10^5, the fitness gradually converges and the whole process is stabilized. In addition, with an upgrade in the failure type, the range of fitness values gradually increases. Under normal circumstances, the value of the fitness decreases from 0 to -1.55, while with the multitarget failure, the value of the fitness decreases from 0 to -3.7; the decline in the value is more than twice as large as in a normal situation. In conclusion, the occurrence of failures has a large impact on the convergence of the algorithm in the migration process, and the more complicated the failure is, the greater the impact. In this algorithm, the function of the network state is regarded as the learning rate, which can significantly improve the convergence effect of the algorithm; these findings and relationships indicate that the algorithm is very meaningful.

5.2.3. *State-adaptation analysis*

In Section 4.2, we have discussed the definition of a state value,

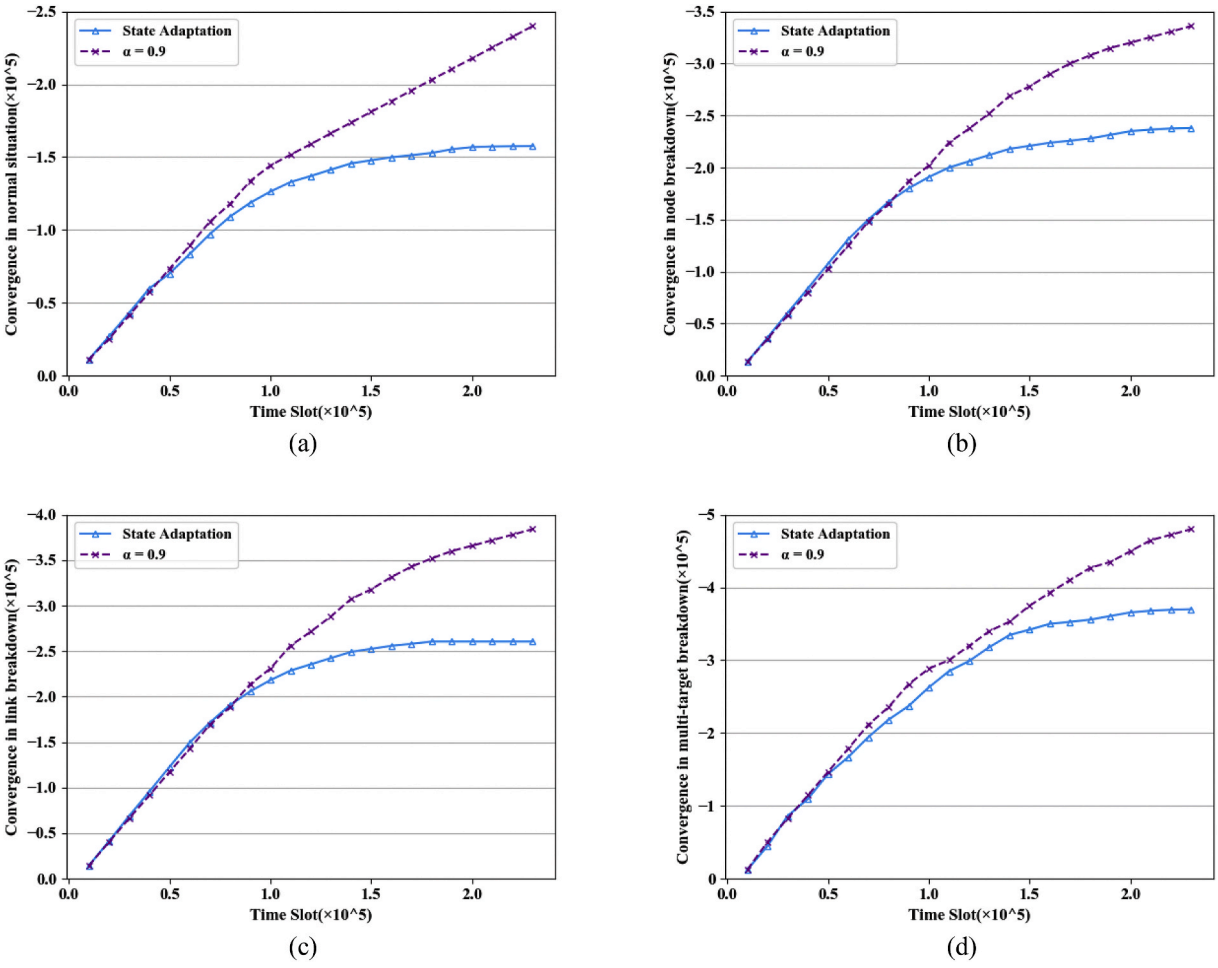


Fig. 12. The convergence in four network situation. (a) Narmal situation; (b) Node failure; (c) Link failure; (d) Multi-target failure.

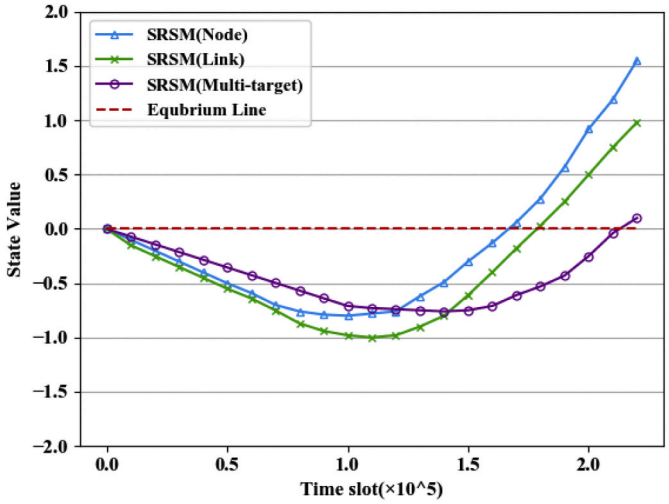


Fig. 13. State value in SRSM algorithm.

which represents whether the edge network breaks down and whether the algorithm adjusts the migration paths at time slot t_o. When the state value is 0, no failure has occurred or the fault has been recovered. The service migration can be performed normally. When the state value is less than 0, the edge network breaks down. In addition, the failure has not been recovered all the time when V becomes smaller and smaller. When the state value starts to rise, the failure starts to recover until the dynamic value is increased again to 0, which indicates that the failure has been recovered. As shown in Fig. 13, the red dotted line represents the normal situation of the edge network. The node failure recovery is the fastest, and it starts at 1.0×10^5 time slots and progresses until 1.7×10^5. The reason for the recovery being fast is that an edge node failure is very simple; it is usually due to insufficient capacity preventing the service from being migrated to the edge server. The recovery time of the link failure is 1.8×10^5, while the recovery time of the multitarget is the longest, which is 2.1×10^5. Since the multitarget refers to the failure of nodes and links in a certain area, it is difficult to recover the fault. In conclusion, the dynamic value function can represent the speed of recovery of different types of faults, and thus, we can define the negative derivative of the dynamic value function as the dynamic change function K of the network. When K = 0, the fault starts to recover. Additionally, the K function is a nonincremental function. Through this function, we can further obtain the learning rate function $\alpha(K,t)$ to improve the convergence speed of the algorithm.

5.2.4. *Performance of the SRSM algorithm*

In this part, we compare our proposed algorithm with three other service migration mechanisms, including NSM, MADA and DQN, in five respects: the e2e delay affected by the migration threshold, the percentage of successful service migration, the service interruption time, the loading balance among the edge servers and the total cost during the migration process.

a) Delay affected by the migration threshold

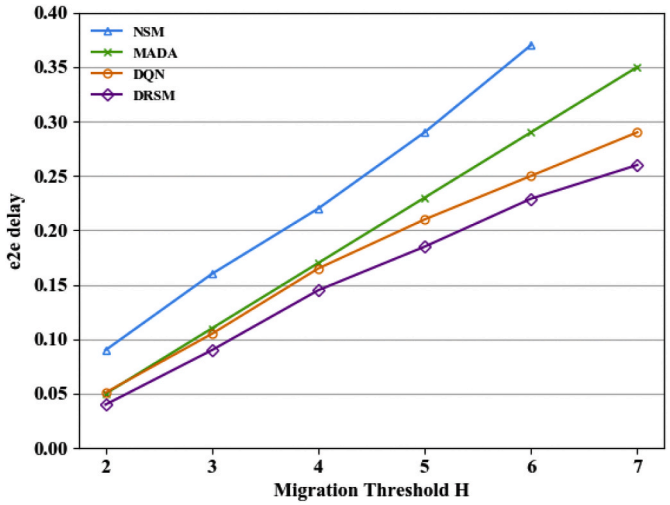


Fig. 14. Performance of the algorithm: e2e delay affected by migration threshold.

During the process of service migration, we set the migration threshold H, which means that when the hops between the terminal and the connected edge server is shorter than H, the service will not be migrated, and this circumstance causes the delay to become longer and longer along with the moving of the user. In this way, we use the e2e delay between the terminal and the edge server to represent the effect of the threshold. As shown in Fig. 14, for four solutions, the delay increases when the threshold hops from 2 to 7. Among these solutions, it can be seen that the delay in our proposed algorithm increases by five times from 0.05 ms to 0.25 ms. However, the DQN solution increases by more than six times, since this algorithm does not consider the user's movement. Additionally, the MADA solution performs worse than DQN, because this solution uses multi-objective optimization. It can also be seen that the NSM solution has the worst performance, and when the threshold equals 7, the service cannot go on normally, and thus, we cannot obtain the e2e delay. In conclusion, the results prove that our proposed method can decline the delay from the terminal to the connected edge server during the migration process and it is better than the other methods in avoiding interruptions.

b) Migration performance affected by user numbers

The number of users affects the success rate of the service migration. The resources and computing power on the edge servers are limited, and thus, when the number of users increases, the service request will compete for resources. In this case, when all of the users are moving in the area, in other words, $p(l_{t+1}|l_t) \in (0,1)$, some of the services cannot be migrated to the available edge servers. To analyze the performance of the successful migration, we set 50, 100, 150, 200 and 250 users during the migration process in normal situations, and for every number of user, we calculate the successful percentage of migration. As shown in Fig. 15, for 50 users, the successful percentage of our proposed method is the highest value, which is 91%, but the NSM method is only 49%.

Other methods have a success rate of approximately 87% and 79%, respectively, for DQN and MADA. In addition, when the number of users increases, the percentage declines from 91% to 60.5% in our SRSM method. The other methods decline substantially, especially the MADA method, which occurs because MADA considers only the distance between the users and the edge servers. In conclusion, our method can guarantee that the service migration goes on successfully in most cases.

c) Service continuity 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 (Slamnik. et al., 2020). However, service migration does not guarantee that the service is completely continuous, and there will be a long or short interruption time. Therefore, in this part, we use SDT to represent the service interruption time generated during the migration process. At every 50 rounds, we calculate the SDT value. Then, we calculate the average of SDT values of these methods. As shown in Fig. 16, the dot line represents the average of SDT values. We can learn that the average SDT of our SRSM method is the lowest, which is 2.1 ms. Other methods' SDT values are 2.7 ms, 3.3 ms and 6.3 ms respectively for DQN, MADA and NSM. The average SDT value of DQN is almost the same as SRSM, which is very close. This is because both of them use deep reinforcement learning algorithm, but the proposed scheme has better consideration of the user's migration probability and movement when setting constraints. However, the average interruption time of the MADA algorithm is longer. This is because the MADA algorithm does not use the deep learning algorithm, but uses the multi-objective optimization algorithm, so the computational complexity is high, and the calculation time of the service migration decision is slow, which has an impact on the continuity of the service. And the NSM method is the highest, because this method doesn't migrate the service with the movement of users, which will cause the service stopping at last. Among them, some of SDT values are too large to guarantee the service going on normally. In conclusion, under the premise of considering the user's mobile situation, SRSM adopts the deep Q-learning algorithm, which

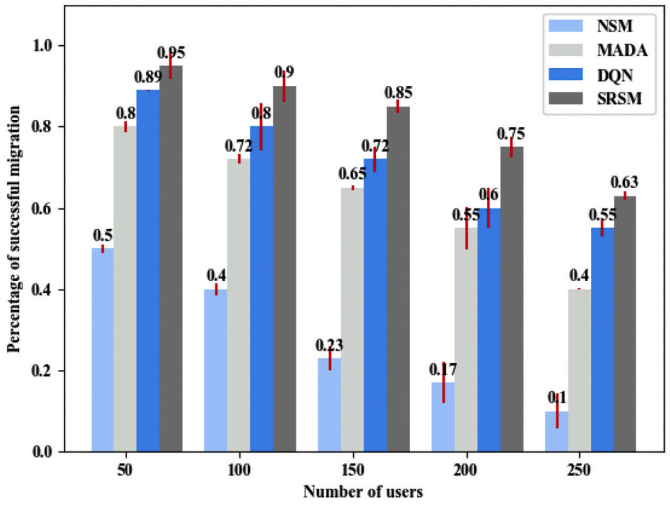


Fig. 15. Performance of the algorithm: Percentage of successful migration.

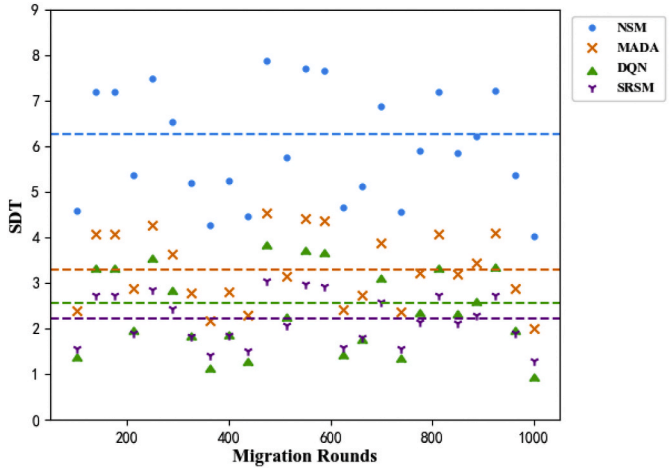


Fig. 16. Performance of the algorithm: interruption time analysis.

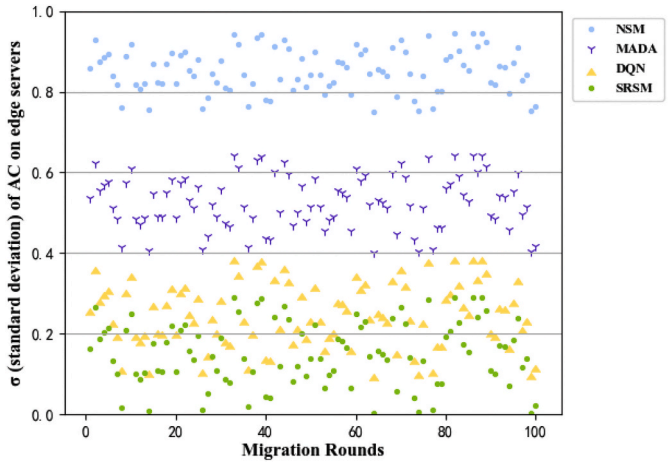


Fig. 17. Performance of the algorithm: Standard deviation of available capacity on edge servers.

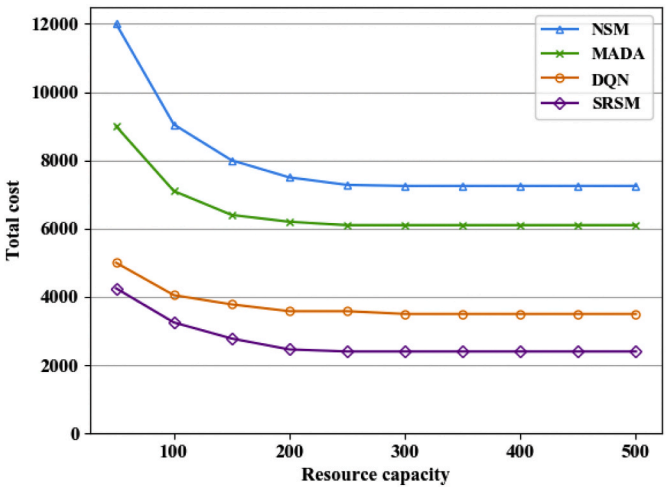


Fig. 18. Total cost analysis.

can shorten the service interruption time and ensure the continuity of service migration. So SRSM method performs better than the MADA and DQN methods which do not care about the dynamic adaption.

d) Loading balance analysis

Load balancing on edge servers is one of the important factors to ensure service migration process (Li et al., 2020). When the service migration process is completed, the load on the edge nodes is balanced, which can indicate the superiority of the mechanism. In this section, we have a total of 100 rounds of migration in the case where the edge network is normal without failure. We calculate the residual energy RE_j on each edge node of each round and record it. Then we calculate the standard deviation of RE_j on all edge nodes of each round, which is $\sigma = \sqrt{\frac{1}{n} \times \sum_{j \in V} (RE_j - \overline{RE})^2}$. The smaller the vari-ance value, the more balanced the load on the edge nodes. As shown in Fig. 17, the standard deviation σ in SRSM method is lower than other methods for most rounds. And the value of σ in NSM is the higher than other methods, which indicates that the balanced performance of SRSM algorithm is the best in this part. We can see that σ in DQN and SRSM methods is at the same level nearly, some values of the standard deviation in DQN is smaller than the standard deviation of the proposed scheme, but the overall is higher. This is because the commonality of the two schemes is that the resource capacity of the node and the bandwidth of the transmission link are considered when establishing the constraints, so the equalization effect of the two schemes is almost the same. However, the MADA algorithm has a poor load balancing effect because it does not comprehensively consider the resources of all edge nodes. In conclusion, because the SRSM method apply the deep Q-learning to solve the migration policy problem, in this way, making migration decisions can consider the available energy of each edge node to guarantee using the energy of edge servers evenly.

e) Total cost analysis

The migration cost is also one of the most important elements in evaluating the service migration algorithm. The smaller the migration cost, the less energy is consumed, and the life cycle of the entire network is prolonged. Therefore, we need to analyze the impact of resource capacity on the edge node on the migration effect. We use $cost = cost_{mig} + cost_{com}$ to represent the whole cost of the migration process and set the resource of edge nodes as [50,500]. As shown in Fig. 18, we can see that the trend of the SRSM, DQN and MADA schemes is that as the resource capacity increases, the migration cost gradually decreases to a certain value and is stable at this value. Since even if the capacity is sufficient, the migration cost is subject to other factors, such as link bandwidth limitations and data packet size. This suggests that the reduction in overall migration cost requires a combination of factors. Next, we can see that at the beginning, the cost of the SRSM algorithm is the lowest, and the stable cost is also the lowest, which is about 2300. Then, the declining rate of the migration cost in the MADA algorithm is the fastest, because this scheme uses a multi-objective optimization algorithm with taking bandwidth, delay, and resource capacity into account when calculating the weighting factors of each element, and the weight of resource capacity is the largest. As for the stable value, we can learn that DQN and SRSM algorithms are almost identical and similar. The cost of MADA algorithm is a bit higher, because MADA is not as comprehensive as SRSM, and does not take into account the user's mobility. However, for NSM scheme, the expansion of resource capacity does not lead to a reduction in migration cost. We can see that as the distance between the user and the edge server increases, the total cost increases, which can indicate the need for service migration.

6. Conclusion and future work

In this article, we propose a novel service migration policy method based on deep reinforcement learning and dynamic adaptation in MEC. In a scenario of moving users and recovering failures in the edge cloud, we first establish four failure models for the condition of the network. The user moves away from the location and the connected edge server receives the user's migration service request. When the migration controller collects the network state information, we propose the determination of the constraints including the cost, delay, resource capacity and bandwidth. After the information collection, we can obtain the candidate space for the migration strategy under different network states. Next, we use the DQN algorithm to obtain the optimal result for a certain failure type through the deformation of the value iterative method. Finally, the migration controller sends the calculated migration schemes to complete the service migration process from ON to DN. At the same time, it proves that the convergence rate of our algorithm's learning rate is fast. Finally, we implement a series of simulation experiments to confirm that our proposed migration mechanism can perform better than the other methods and can overcome the negative impact of faults during the service migration. However, there are still limitations in our work. Our work cannot predict the moving track in the next timeslot according to the user's action habits. And, we cannot predict certain failure occurrence, but this aspect is more important in the 5G generation. Additionally, the prediction mechanism will be included in our work for research in the future. We can design a solution to decrease the computational complexity and use a failure prediction

method to avoid the impact of network failures on service migration decisions on the basis of a mobility prediction mechanism.

Credit author statement

LanLan Rui: Conceptualization, Methodology. Menglei Zhang: Data curation, Writing – original draft preparation, Investigation. Zhipeng Gao: Visualization, Investigation. Xuesong Qiu: Supervision. Zhili Wang: Software, Validation. Ao Xiong: Visualization, Writing- Reviewing and Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The work was supported by National Key Research and Development Program of China (2020YFB1807802). Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jnca.2021.103058.

References

Bittencourt, L., Lopes, M., Petri, I., Rana, O., 2015. Towards virtual machine migration in fog computing. In: Proceeding of Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput., 3PGCIC. IEEE, pp. 1–8.
Brandherm, F., Wang, L., Mühlhäuser, M., 2019. A learning based framework for optimizing service migration in mobile edge clouds. In: Proceeding of ACM Int. Workshop Edge Syst., Anal. Netw., Part EuroSys. ACM, pp. 12–17.
Chen, M., Li, W., Fortino, G., et al., 2019. A dynamic service migration mechanism in edge cognitive computing. ACM Trans. Internet Technol. ACM 19 (2), 1–15.
Chu, T.S., Wang, J., Codeca, L., et al., 2020. Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE T Intell Transp 21 (3), 1086–1095.
De, V., Grassi, V., 2019. Architectural issues for self-adaptive service migration management in mobile edge computing scenarios. In: Proc. - IEEE Int. Conf. Edge Comput., EDGE - Part IEEE World Congr. Serv. IEEE, pp. 27–29.
Dhakad, C., Bisen, A., 2016. Efficient route selection by using link failure factor in MANET. In: Proceedings of Int. Conf. Electr., Electron., Optim. Techniques. ICEEOT. IEEE, pp. 3740–3743.
Duan, Q., Al-Shaer, E., Jafarian, H., 2013. Efficient Random Route Mutation considering flow and network constraints. In: Proceeding of CNS, National Harbor. IEEE, MD, pp. 260–268.
Gao, Z., Jiao, Q., Xiao, K., et al., 2019. Deep reinforcement learning based service migration strategy for edge computing. In: Proceeding of IEEE Int. Conf. Service-Oriented Syst. Eng., SOSE. IEEE, pp. 116–121.
Hart, P., Nilsson, N., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4, 100–107.
Hossain, M.D., Huynh, L.N.T., Sultana, T., et al., 2020. Collaborative task offloading for overloaded mobile edge computing in small-cell networks. In: Proceeding of ICOIN. IEEE, pp. 717–722.
Jaakkola, T., Jordan, M., Singh, S., 1994. On the convergence of stochastic iterative dynamic programming algorithms. Neural Comput. 6 (6), 1185.
Jha, V., Prakash, N., Mohapatra, A., 2019. Energy efficient model for recovery from multiple nodes failure in wireless sensor networks. Wireless Pers. Commun. 108, 1459–1479.
Jia, H., Gai, Y., Zheng, H., 2018. Network Recovery for large-scale failures in smart grids by reinforcement learning. In: Procedding of IEEE Int. Conf. Comput. Commun. IEEE, pp. 2658–2663.
Jing, H., Zhang, Y., Zhou, J., et al., 2018. LSTM-based service migration for pervasive cloud computing. In: Proceedings of IEEE Int. Congr. Cybermatics: IEEE Conf. Internet Things, Green Comput. Commun., Cyber, Phys. Soc. Comput., Smart Data, Blockchain, Comput. Inf. Technol., iThings/GreenCom/CPSCom/SmartData/Blockchain/CIT. IEEE, pp. 1835–1840.
Koyasako, Y., Suzuki, T., Kim, S.Y., et al., 2020. Real-time motion control method using measured delay information on access edge computing. In: Proceeding of IEEE Annu. Consum. Commun. Netw. Conf. IEEE, pp. 1–4.
Ksentini, A., Taleb, T., Chen, M., et al., 2014. A Markov Decision Process-based service migration procedure for follow me cloud. In: Proceeding of IEEE Int. Conf. Commun., ICC. IEEE, pp. 1350–1354.
Lee, J., Kim, J., Tae, Y., Pack, S., 2018. QoS-aware service migration in edge cloud networks. In: Proceeding of IEEE Int. Conf. Consum. Electron. - Asia. IEEE, pp. 1–3.
Li, J., Shen, X., Chen, L., et al., 2019. Service migration in fog computing enabled cellular networks to support real-time vehicular communications. IEEE Access 7, 13704–13714.
Li, C., Tang, J., Luo, Y., et al., 2020. Service Cost-Based Resource Optimization and Load Balancing for Edge and Cloud Environment. Knowl Inf Syst. Springer, pp. 1–21.
Liu, J.J., Guo, H.Z., Xiong, J.Y., et al., 2020. Smart and resilient EV charging in SDN-enhanced vehicular edge computing networks. IEEE J. Sel. Areas Commun. IEEE 38 (1), 217–228.
MacHen, A., Wang, S., Leung, K., Ko, B., Salonidis, T., 2016. Migrating running applications across mobile edge clouds: poster. In: Proceeding of Annu Int Conf Mobile Comput Networking. ACM, pp. 435–436.
Nawrocki, P., Sniezynski, B., 2018. Adaptive service management in mobile cloud computing by means of supervised and reinforcement learning. J. Netw. Syst. Manag. 26 (1), 1–22.
Pelamatti, J., Brevault, L., Balesdent, M., et al., 2020. Bayesian optimization of variable-size design space problems. Optim. Eng. 1–49.
Rauf, U., Gillani, F., Al-Shaer, E., et al., 2016. Formal approach for resilient reachability based on end-system route agility. In: Proceedings of the 2016 ACM Workshop on Moving Defense. ACM, pp. 117–127.
Roman, R., Lopez, J., Mambo, M., et al., 2018. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. Future Generation Computer Systems, vol. 78. Elsevier B.V., pp. 680–698
Rosário, D., Schimuneck, M., Camargo, J., et al., 2018. Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support. Sensors 18 (2), 1–17.
Sadilek, A., Krumm, J., 2012. Far out: predicting long-term human mobility. In: Proceeding of Natl Conf Artif Intell. AI Journal, pp. 814–820.
Sasithong, P., Quynh, L., Saengudomlert, P., et al., 2019. Maximizing double-link failure recovery of over-dimensioned optical mesh networks. Opt. Switch. Netw. 36, 1–10.
Saurez, E., Hong, K., Lillethun, D., et al., 2016. Incremental deployment and migration of geo-distributed situation awareness applications in the fog. In: Proceeding of ACM Int. Conf. Distrib. Event-Based Syst. ACM, pp. 258–269.
Singh, S., Lewis, R., Barto, A., et al., 2010. Intrinsically motivated reinforcement learning: an evolutionary perspective. IEEE Trans. Auton. Mental Dev. 2, 70–82.
Slamnik, K.N., de Resende, H.C.C., Donato, C., et al., 2020. Leveraging mobile edge computing to improve vehicular communications. In: Proceeding of IEEE 17th Annual Consumer Communications and Networking Conference. IEEE, pp. 1–4.
Talaat, F.M., Saraya, M.S., Saleh, A.I., et al., 2020. A Load Balancing and Optimization Strategy (LBOS) Using Reinforcement Learning in Fog Computing Environment. Journal of Amb Intel Hum Comp. Springer, pp. 1–16.
Taleb, T., Ksentini, A., Frangoudis, P., et al., 2019. Follow-me cloud: when cloud services follow mobile users. IEEE Trans. Cloud Comput. 7, 369–382.
Tang, Z., Zhou, X., Zhang, F., et al., 2019. Migration modeling and learning algorithms for containers in fog computing. IEEE Trans. Serv. Comput. 12, 712–725.
Urgaonkar, R., Wang, S., He, T., et al., 2015. Dynamic service migration and workload scheduling in edge clouds. Perform. Eval 91, 205–228.
Wang, S., Urgaonkar, R., He, T., et al., 2014. Mobility-induced service migration in mobile micro-clouds. In: Proceeding of IEEE Mil Commun Conf MILCOM. IEEE, pp. 835–840.
Wang, Z., Gu, R., Zhang, G., Zhao, T., et al., 2018a. Demonstration of network slicing in mobile edge computing service migration. In: Proceeding of Asia Commun. Photonics Conf. IEEE, pp. 1–3.
Wang, S., Xu, J.L., Zhang, N., Liu, Y., 2018b. A survey on service migration in mobile edge computing. IEEE Access 6, 23511–23528.
Ye, J., Zhang, Y., 2019. DRAG: deep reinforcement learning based base station activation in heterogeneous networks. IEEE Trans. Mobile Comput. 1, 1.
Yoshida, N., Uchibe, E., Doya, K., 2013. Reinforcement learning with state-dependent discount factor. In: Proceedings of IEEE Jt. Int. Conf. Dev. Learn. Epigenetic Rob. ICDL - Electron. Conf. IEEE, pp. 1–6.
Yuan, Q., Li, J.L., Zhou, H.B., et al., 2020. A joint service migration and mobility optimization approach for vehicular edge computing. IEEE Trans. Veh. Technol. 69 (8), 9041–9052.
Zhao, D., Yang, T., Jin, Y., et al., 2017. A service migration strategy based on multiple attribute decision in mobile edge computing. In: Proceeding of ICCT. IEEE, pp. 986–990.

Lanlan Rui is an associate professor of Computer Science in Beijing University of Posts and Telecommunications. Her research interests include communication software and network management.

Menglei Zhang received her BS degree in Electronic Information Engineering from Xidian University in 2018. She is a master in Beijing University of Posts and Telecommunications. Her research interests include edge computing and network management.

Zhipeng Gao is a professor of Computer Science in Beijing University of Posts and Telecommunications. His research interests include Blockchain cross-chain technology and intelligent applications, edge data analysis and edge intelligent systems, big data intelligent management and analysis technologies, etc.

Xuesong Qiu (Senior Member, IEEE) received the Ph.D. degree in communication and information systems from the Beijing University of Posts and Telecommunications, Beijing, China, in 2000.He is currently a professor with BUPT, and the deputy director of the

State Key Laboratory of Networking and Switching Technology, and vice president of the school of computer, Beijing University of Posts and Telecommunications. He has authored/co-authored more than 200 academic papers. His major research interests include network and service management, information and communication technology of smart grid. He was editor for the two is International Telecommunications Union Telecommunication Standardization Sector (ITU-T) standards and four industry standards of China. He was twice recipient of the National Scientific and Technological Progress Prize of China.

Zhili Wang is an associate professor of Computer Science in Beijing University of Posts and Telecommunications. He received the ME from Xidian University in 1998, and PhD from BUPT in 2005. His research interests include Network management and communication software.

Ao Xiong is an associate professor of Computer Science in Beijing University of Posts and Telecommunications. His research interests include Network management and communication software.

Update

Journal of Network and Computer Applications

Volume 186, Issue , 15 July 2021, Page

DOI: https://doi.org/10.1016/j.jnca.2021.103085

Journal of Network and Computer Applications 186 (2021) 103085

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Corrigendum to "Service migration in multi-access edge computing: A joint state adaptation and reinforcement learning mechanism" [J. Netw. Comput. Appl. 183–184 (2021) 103058]

LanLan Rui, Menglei Zhang, Zhipeng Gao, Xuesong Qiu, Zhili Wang, Ao Xiong *

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

The authors regret to inform that the authors forgot to mark two references and some figures need to be changed due to the authors' operational reasons.

1) Page 4, Fig. 3 of Section 3.1.:

Fig. 3 from the original paper can't clearly show the process of the service migration, so the authors change it to the flow diagram as follows:

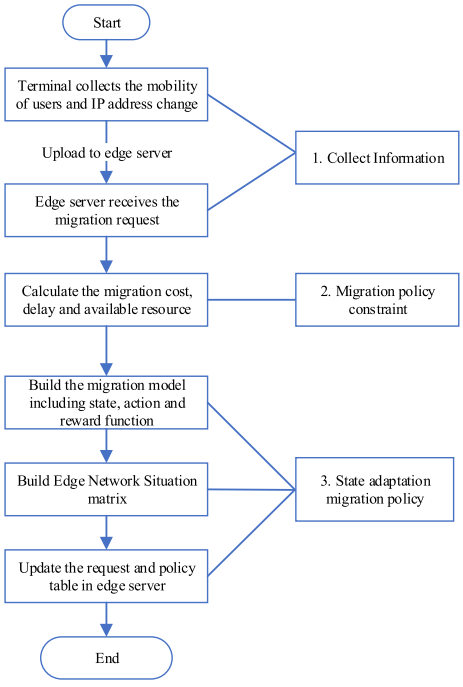


Fig.3. Flow diagram of the service migration policy within SRSM

DOI of original article: https://doi.org/10.1016/j.jnca.2021.103058.
* Corresponding author.
E-mail address: xiongao@bupt.edu.cn (A. Xiong).

https://doi.org/10.1016/j.jnca.2021.103085

Available online 20 May 2021

2) Page 5, Fig. 4 of Section 3.2.1.:

The authors replaced the circle dots with the edge servers to make the edge network model specific and understood easily. The corrected Fig. 4 is shown as follows:

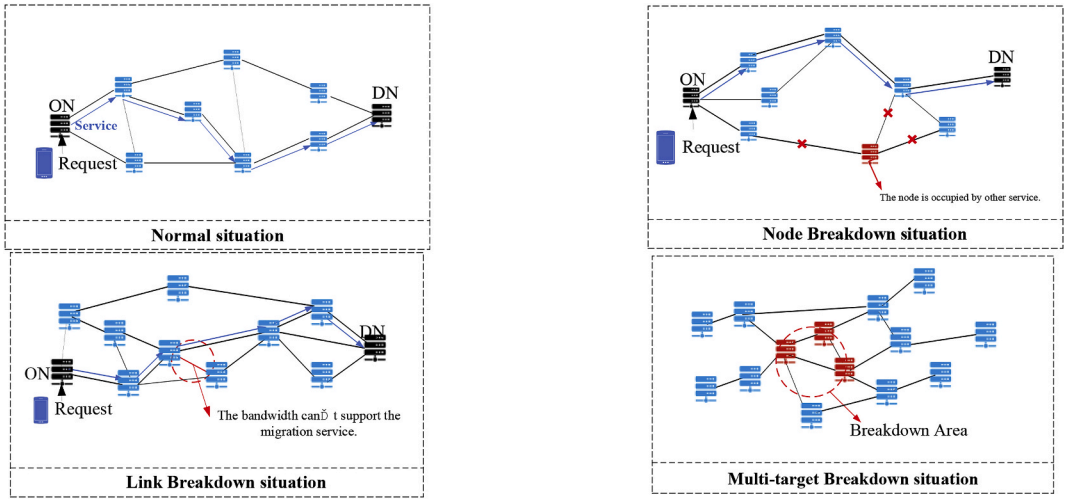


Fig.4. Four edge network situations (a) Normal situation; (b) Node failure situation; (c) Link failure situation; (d) Multi-target failure situation

3) Page 13, Fig. 12 of Section 5.2.2.:

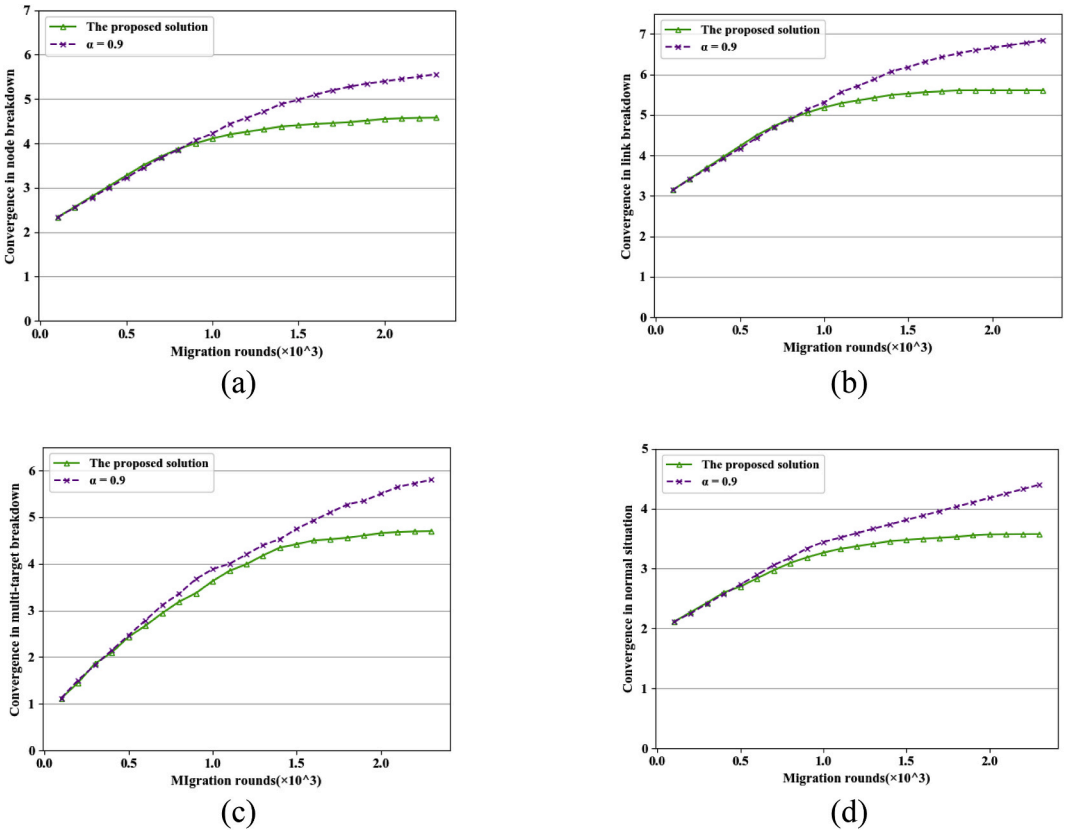


Fig.12. The convergence in four network situation. (a) Narmal situation; (b) Node failure; (c) Link failure; (d) Multi-target failure.

4) Page 13, Section 5.2.3.:

Delete all contents of Section 5.2.3, and replace them with the contents of the following paragraph and Fig. 13:

In Section 4.2, we have discussed the definition of a state value, which represents how the proposed solution can avoid different types of breakdown to obtain the better service migration's effect. At first, the value is below 0, it indicates that the migration policy is not good because of the breakdown. As the migration round goes by, the value grows up because our migration policy considers the cost, delay and available resources for different network states. At last, the value can be stable at a certain value, which represents the final migration effect of our solution. As shown in Fig. 13, the node failure recovery is the fastest, and it starts at 350 rounds and progresses until 41.3. The reason for the recovery being fast is that an edge node failure is very simple; it is usually due to insufficient capacity preventing the service from being migrated to the edge server. The recovery time of the link failure is at 400 migration rounds, while the recovery time of the multitarget is the longest, which is 450 rounds. Since the multitarget refers to the failure of nodes and links in a certain area, it is difficult to recover the fault. In conclusion, our solution can adapt to different network states and obtain the optimal service migration.

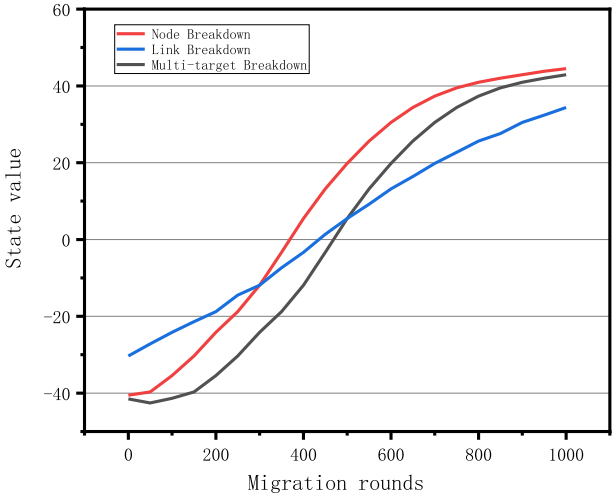


Fig.13. State value in SRSM algorithm.

5) Page 5, the fourth paragraph of Section 3.2.1

The sentence "As shown in Fig. 4, we will analyze the following situations." was replaced with "As shown in Fig. 4, we will analyze the following situations with the reference in (Tao et al., 2019)". And the reference (Tao et al., 2019) is "Tao. Z., Xiaohui. K., Zan. Z., et al., 2019. An Intelligent Route Mutation Mechanism against Mixed Attack Based on Security Awareness. In: Proceeding of IEEE Global Communications Conference. IEEE, pp. 1–6."

6) Page 6, the second paragraph of Section 3.3.3

Add the reference (Tao et al., 2019) at the end of the sentence "Migration paths should not include those edge nodes that have unavailable resources for extra requests, as shown in Eq. 8 (Nawrocki and Sniezynski, 2018)."

7) Page 7, the first paragraph of Section 3.4.2

Add the reference (Tao et al., 2019) at the end of the sentence "Thus, the edge network state matrix is represented as follows:"

8) Page 8, the first paragraph of Section 4.2

Add the sentence "We use the proof method in (Changqiao et al., 2021) for reference." at the end of the first paragraph of Section 4.2. And the reference (Changqiao et al., 2021) is "Changqiao X., Tao Z., Xiaohui K., et al, 2021. Context-aware Adaptive Route Mutation Scheme: A Reinforcement Learning Approach. IEEE Internet of Things Journal. IEEE, pp 1-14. https://doi.org/10.1109/JIOT.2021.3065680"

9) Page 7, the definitions of Section 3.4.1

The locations of "Definition 1", "Definition 2" and "Definition 3" are behind the location of the equation (14).

10) Page 11, table 3 of Section 5.1

In the first row of table 3, the service migration threshold in the original article is polished wrongly. The correct value of migration threshold is set at 2 to 7 hops.

The authors would like to apologize for any inconvenience caused.

References

Changqiao, X., Tao, Z., Xiaohui, K., et al., 2021. Context-aware Adaptive Route Mutation Scheme: A Reinforcement Learning Approach. In: IEEE Internet of Things Journal. IEEE, pp. 1–14. https://doi.org/10.1109/JIOT.2021.3065680.

Tao, Z., Xiaohui, K., Zan, Z., 2019. An Intelligent Route Mutation Mechanism against Mixed Attack Based on Security Awareness. In: Proceeding of IEEE Global Communications Conference. IEEE, pp. 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9014119.