Contents lists available at ScienceDirect

Computer Networks

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

Priority-awareness VNF migration method based on deep reinforcement learning

Hua Qu [a,b], Ke Wang [a,*], Jihong Zhao [c]

[a] School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[b] School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[c] School of Communication and Information Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710061, China

A R T I C L E I N F O

Keywords:
Service function chain
Priority-awareness
Deep reinforcement learning

A B S T R A C T

By decoupling the software function on hardware devices, Network Function Virtualization (NFV) provides a new service architecture named Service Function Chain (SFC), which combines multiple Virtual Network Functions (VNFs) in a specific order. In order to improve the reliability and Quality of Service (QoS) of network services, VNF migration provides an effective solution for this requirement. However, traditional migration methods lack an appropriate VNF selection mechanism and consider a single network state. Moreover, how to determine the accurate migration location dynamically with machine learning is challenging. This paper proposed a novel PAVM algorithm for reliable and optimal VNF migration, which distinguishes two kinds of VNF migration schemes for network congestion and node failure states respectively, and a node and VNF priority-awareness mechanism is designed, which can select appropriate VNF on suitable node to migrate. Moreover, PAVM utilizes deep reinforcement learning algorithm to choose target node location of VNF migration, which jointly utilizes delay, load balancing of network as feedback factors to optimize the QoS. The experimental results indicate that compared with other three benchmark algorithms, PAVM can effectively reduce the transmission delay and improve node and link load balancing after the VNF migration.

1. Introduction

In the traditional network, the realization of network function relies on proprietary hardware. The network functions on different hardware are combined into certain network services, however, with the rapid increase of the number and diversity of network service requests, the traditional deployment of dedicated hardware by operators cannot meet the current complex and changeable network requirements. In order to solve this problem, Network Function Virtualization (NFV) technology uses general hardware such as x86 and virtualization technology to carry many functions of software processing [1]. NFV can decouple the hardware and software and abstract the functions, so that the functions of network devices no longer depend on the dedicated hardware, and the resources can be fully and flexibly shared to realize the rapid development and deployment of new services. NFV can also carry out automatic deployment, elastic scaling, fault isolation and self-healing based on the actual business needs [2]. In NFV, several Virtual Network Functions (VNF) are deployed on server nodes in a certain order to form a network function sequence that can handle specific network services, namely Service Function Chain (SFC), which uses Software Defined Network (SDN) function to create a service chain (such as firewall, network address translation, intrusion protection) connecting network services and connect it to the virtual chain [3]. Network operators can use this function to set the suite or directory of connection services, so that many services with different characteristics can use a single network connection [4].

In the 5 G network scenario, the demand for network services has increased significantly, which leads to network congestion and reduces the Quality of Service (QoS). On the other hand, the instability of the network environment is also easily to cause network failure, which results in a large number of SFCs unable to provide services normally. To this end, the emergence of VNF migration strategy solves these problems [5]. VNF migration, that is, in the network with NFV characteristics, VNF will be removed from the traditional physical machine and redeployed on better nodes, which can solve the problem of load imbalance in NFV and SDN deployment. However, there are some shortcomings that need to be addressed. Firstly, most of the methods only consider a single network situation, and do not analyze the VNF migration schemes

* Corresponding author. Xi'an Jiaotong University, China.
E-mail address: wangke12345@stu.xjtu.edu.cn (K. Wang).

https://doi.org/10.1016/j.comnet.2022.108866
Received 5 July 2021; Received in revised form 7 February 2022; Accepted 22 February 2022
Available online 6 March 2022
1389-1286/© 2022 Elsevier B.V. All rights reserved.

under different network failure situations. Secondly, the current researches lack a suitable priority mechanism, which leads to the irregular migration order of VNF, and the important SFC is difficult to be optimized or restored. Last but not least, traditional VNF migration mechanism can only deal with simple network scenarios, which is difficult to adapt to the network environment with complex network state and migration strategy.

In order to solve the above shortcomings in VNF migration problem, some literatures have carried out some research to a certain extent. Previously, some SFC deployment methods can achieve reliability results like VNF migration [6–13]. However, the change of network state is not considered in SFC deployment method, which may lead to SFC failure and unable to recover. So that several VNF migration methods optimize the related QoS indexes to a certain extent to solve this problem [14–17]. However, they ignore the network failure, and the static method is difficult to adapt to the large-scale dynamic VNF migration needs, and due to the dependency between VNFs in SFC, the current VNF decision will affect the next VNF. For that reason, more advanced and powerful reinforcement learning models have significantly achieved performance gains in SFC deployment and VNF migration problem [18, 19], which can solve sequential decision problem well. However, reinforcement learning, a classical algorithm of reinforcement learning, needs to maintain a quite large Q-table because of large state and action set, so that the computing power of the algorithm will be affected with wasting of CPU and memory. On the other hand, the search processing of reinforcement learning is very time-consuming. To this end, deep learning is proposed to simulate the Q value, and the deep reinforcement learning is proposed to apply on VNF migration problem [20,21]. Specifically, the deep reinforcement learning can choose a more suitable migration location for VNF with learning process, and it can make accurate judgements about unknown situations in the sample.

In order to effectively distinguish the importance of network node and SFC, comprehensively consider the network state, and solve the problem of dynamic VNF migration in complex networks, this paper proposes node and VNF priority-awareness strategies firstly, then designs VNF migration schemes for network congestion and failure respectively, and designs a VNF migration algorithm based on deep reinforcement learning for network congestion and node failure states finally. The main contributions of this paper are as follows.

1 Firstly, in order to effectively distinguish the importance of network node and SFC, we propose node and VNF priority-awareness strategies respectively. In view of the network failure, we measure the VNF priority according to the SFC priority, which is calculated by the CPU and bandwidth resource requirements, and the proportion of node and link resources affected by node failure. In view of the network congestion, we first calculate the node priority according to the node and link resource occupation, and then calculate the VNF priority of the highest priority node. It is the first time that determining the node and VNF priority in our work.
2 Secondly, in order to comprehensively consider the network state, we design two VNF migration schemes for network congestion and failure respectively. The former scheme selects the appropriate VNF to migrate to the appropriate node according to the node priority and VNF priority, and the latter scheme migrates VNF to the appropriate node in order according to the VNF priority.
3 Lastly, in order to solve the problem of dynamic VNF migration in complex networks, we apply deep reinforcement learning algorithm on the process of VNF migration, then we propose PAVM model to migrate VNFs with QoS requirement. Specifically, we design the feedback function with three factors including transmission delay, node balancing, and link balancing. Then, we evaluate the proposed PAVM model on the Internet2 network topology, and the extensive experimental results demonstrate that PAVM can effectively migrate VNFs to appropriate nodes and achieve QoS performance compared with other three models including two widely used VNF migration methods and random priority scheme of PAVM.

The rest of this paper is organized as follows Section 2. surveys the related works. In Section 3, the priority awareness framework used for determining migration order is described and then the proposed PAVM algorithm based on deep reinforcement learning to solve VNF migration is discussed in details. In Section 4, the experimental designs and simulation results are shown. Finally, the conclusion and discussion of future work are given in Section 5.

2. Related works

VNF migration is to improve the reliability of SFC, and in previous studies, some SFC deployment methods can achieve this goal. For example, Francisco Carpio and Admela Jukan improved service reliability using jointly replications and migrations, and then proposed a N2N algorithm based on LP to improve reliability and network load balancing [6]. However, the algorithm cannot guarantee performance of transmission delay in results. Abdelhamid Alleg et al. proposed a reliable placement solution of SFC modeled as a mixed ILP program, they designed the SFC availability level as a target and reduce the inherent cost which is affected by diversity and redundancy [7]. However, the proposed model is only applied on statical placement and lack of adaptability of machine learning techniques. Tuan-Minh Pham et al. designed the UNIT restoration model and the PAR protection algorithm based on ILP framework, the proposed algorithms protect SFC requests from network failures in terms of both resource restoration and recovery time efficiently [8]. Although the model achieves robustness of network, yet it does not guarantee the delay in a service demand, moreover, the solution needs to consider the dynamic parameters of online demands. Mohammad Karimzadeh-Farshbafan et al. proposed a polynomial time sub-optimal algorithm named VRSP which is based on mixed ILP in multi-infrastructure network provider environment [9]. However, VRSP solves the reliable SFC deployment problem only by statical method, moreover, VRSP does not consider transmission delay and network balancing in the optimization process. Qu et al. formulated the reliable SFC deployment problem as a mixed ILP and proposed a delay-aware hybrid shortest path-based heuristic algorithm [10]. Although they achieved high reliability and low latency through the model based on ILP and heuristic algorithm, yet the model is too hard to satisfy complex and dynamic network environment. Ye et al. proposed a novel heuristic algorithm to efficiently address the joint topology design and mapping of reliable SFC deployment problem [11]. However, the model does not consider the delay and load balancing as optimization targets, and heuristic algorithm is easy to fall into the local optimum unlike machine learning algorithm such as reinforcement learning and deep reinforcement learning. Mao et al. proposed a deep reinforcement learning based online SFC placement method to guarantee seamless redirection after failures and ensure service reliability [12]. Although the model automatically deploys both active and standby instances in real-time, yet it does not distinguish the priority of SFC and cannot guarantee transmission delay and network load balancing. Moreover, in the previous work, we designed a reliable SFC deployment algorithm based on deep reinforcement learning, which selects the appropriate backup scheme for each VNF by calculating priority [13]. However, compared with migration mode, the backup mode needs to occupy more node and link resources. Although the above literatures improve the reliability of SFC to a certain extent, yet they basically do not consider the optimization of multiple indicators at the same time, on the other hand, they do not consider the possible network state changing at any time, which will lead to a large number of SFC instant failure and unable to recover.

In order to solve the above problem, some researches have proposed a variety of VNF migration algorithms, by migrating specific VNF from the current node to other node, so as to achieve the relevant QoS indicators. For example, Xu et al. proposed a reliability-and-energy-balanced SFC deployment and migration algorithm for Internet of things scenario, which improved the reliability and load balance in the SFC deployment process and optimized the resource allocation in the SFC migration process respectively [14]. However, the method migrates all VNFs on SFC, which results in high cost of migration. Moreover, the method does not consider the failure of network nodes, so its applicability is not wide enough. Vincenzo Eramo et al. designed a consolidation algorithm to migrate VNF with considering energy consumption and revenue loss due to QoS degradation [15]. Although they reduce the total energy consumption, yet more optimization targets are needed, including transmission delay, load balancing and so on. Daewoong Cho et al. proposed VNF-RM algorithm to reduce delay in changing resource availability [16]. However, the method does not consider node failure and more QoS indicators, on the other hand, the algorithm is too uncomplicated to deal with large network state. Yi et al. designed a VNF migration algorithm based on node and link state awareness respectively, which migrates VNFs from poor-state nodes to good-state nodes [17]. Although the algorithm can achieve load balancing, yet it does not consider the network failure and lacks the optimization of transmission delay. Although these researches have solved the VNF migration problem effectively, yet they ignored the network failure, and difficult to adapt to large-scale dynamic VNF migration. The above methods are based on LP or heuristic algorithms. Their advantage is that they can better deal with problems in a simple network environment and can be optimized for a single objective. However, they have some shortcomings. On the one hand, they are difficult to adapt to complex and dynamic network scenarios due to algorithm performance constraints. On the other hand, they consider a single optimization goal and are difficult to meet higher QoS requirements.

More recently, in order to solve the shortcomings of the above LP and heuristic algorithms, several machine learning models have shown their superior capabilities in VNF migration due to their advanced feature extraction and data modeling abilities, they can quickly adjust the strategy according to the environment in the algorithm process to adapt to the dynamic and complex network scene. On the one hand, some literatures use reinforcement learning to solve VNF migration problem. For example, Xie et al. utilized integer linear program to model SFC migration process, and proposed a migration algorithm based on Markov chain [18]. Although they can effectively optimize the network load in a specific model. However, the Markov chain is not suitable for the case of large state and action space, furthermore, the QoS target optimized by this method is relatively single, and the priority relationship between VNFs is not considered. Li et al. formulated the VNFs scheduling problem as a mixed integer linear program and modeled it as a Markov decision process [19]. Although they leverage a reinforcement learning algorithm to achieve near-optimal performance, yet the Markov decision process needs to maintain a large state transition probability table when the state and action space is large, and the search time consumption is large, so it is not suitable for the current complex network situation. On the other hand, some deep reinforcement learning algorithms are applied to solve the VNF migration problem. For example, Tang et al. proposed a VNF migration based on deep belief network, which resolves the problem of lacking effective prediction, then they designed a topology-aware greedy algorithm to optimize system cost [20]. However, the method does not distinguish the priority of VNFs and cannot recover SFC with high priority at the first time. Chen et al. designed a deep Q-network algorithm to solve single SFC migration problem, then proposed a novel multi-agent cooperative framework to address multiple SFC migration [21]. However, the algorithm does not optimize the network load balance and the transmission delay, and does not consider the network failure situation. Although these algorithms can solve the problem of large-scale and dynamic VNF migration to a certain extent, yet they lack a suitable priority-awareness mechanism in VNF migration, which is difficult to select the VNF with the highest priority for migration. Motivated by it, this paper will for the first-time design priority-awareness model, then applies deep reinforcement learning to solve VNF migration problem in NFV framework. Moreover, the algorithm comparison of related work is shown in Table 1.

3. The proposed PAVM framework

In this section, in order to establish the mathematical model of VNF migration problem that we need to solve, we first formulate the VNF migration problem. Secondly, we discuss the priority awareness framework of network node and VNF in details to effectively distinguish the importance of network node and SFC. Next, we describe the VNF migration mechanisms under node congestion and failure respectively to comprehensively consider the network state. Finally, in order to solve the problem of dynamic VNF migration in complex networks, the proposed VNF migration algorithm based on deep reinforcement learning is explained.

3.1. VNF migration formulation

The VNF migration is aimed to find suitable VNF to migrate to appropriate node. In this work, we define the physical network topology as an undirected graph, which is represented as $G = (V,E)$, where V is the set of network nodes and $|V| = n_V$, E denotes the set of links that connect two network nodes directly and $|E| = n_E$. In addition, we define the set of SFC requests as $Req = \{sfc_1, sfc_2, \ldots, sfc_n\}$, and each SFC request consists of several different VNFs and virtual links between two adjacent VNFs.

Because network hosts VNF and virtual link via CPU and bandwidth resources respectively, therefore we mainly consider CPU resources on network nodes and bandwidth resources on physical links. Due to the difference of computing power requirements, we assume that the number of CPUs occupied by different VNF is not same. Similarly, we assume that the amount of bandwidth resources occupied by different virtual link is different too. Moreover, in order to quantify the network load balancing, we define variance of CPU usage percentage on all network nodes as an indicator. Specifically, the load variance represents the discrete level of node resource usage, mathematically, it is the square of the standard deviation. The formula is as follows,

$$bal_n = \sum_{i=1}^{n_V} \left(\frac{u_i}{U_i} - \sum_{j=1}^{n_V} \frac{u_j}{U_j} \Big/ n_V \right)^2 \Big/ n_V, \tag{1}$$

where u_i and U_i indicate remaining and total CPU resources on node i respectively. And then, we define variance of bandwidth usage percentage on all physical links as an indicator too,

$$bal_l = \sum_{i=1}^{n_E} \left(\frac{w_i}{W_i} - \sum_{j=1}^{n_E} \frac{w_j}{W_j} \Big/ n_E \right)^2 \Big/ n_E, \tag{2}$$

where w_i and W_i indicate remaining and total bandwidth resources on link i respectively. In addition, SFCs consist of several different VNFs, so the length of SFCs may has a difference, as a result the delay of different SFCs will be quite different. For that reason, we use average delay of adjacent VNFs to measure the performance of delay optimization,

$$del = \sum_{i=1}^{n} \sum_{j=2}^{l_i} d(f_{j-1}, f_j) \Big/ \sum_{i=1}^{n} (l_i - 1), \tag{3}$$

where l_i denotes the length of SFC_i, and $d(f_{j-1}, f_j)$ represents the delay between VNF f_{j-1} and f_j. Besides, there are limits of the network resources, it means that the resource on each node and link is non-negative and no more than the maximum resource limit, which is expressed as,

$$\begin{cases} 0 \le u_n \le U_n, \ \forall n \in V \\ 0 \le w_l \le W_l, \ \forall l \in E \end{cases}. \tag{4}$$

3.2. Priority awareness of network node and VNF

Priority awareness is aimed to determine the important level of node and VNF through network status and resource requirements. Firstly, the model perceives the current abnormal network state and confirms whether it is network congestion or node failure. Then, in view of the network congestion, the algorithm calculates the node priority according to the node and link resource occupation, and then calculates the VNF priority on the highest priority node, the algorithm arranges the priority of VNFs in descending order and migrates the highest priority VNF. In view of the network failure, the algorithm calculates the VNF priority according to the SFC priority, which is calculated by the CPU and bandwidth resource requirements, and the proportion of node and link resources affected by failure node. Finally, the algorithm arranges the priority of nodes and VNFs in descending order, and migrates VNFs in order.

3.2.1. Node priority

In view of the network congestion, we use the CPU resource usage of node and bandwidth resource usage of adjacent links to measure a node's priority, which is defined as,

$$p_{node} = \frac{p_1 + p_2}{2} \tag{5}$$

$$p_1 = (U_m - u_m)/U_m \tag{6}$$

$$p_2 = \sum_{n=1}^{n_V}(W_{mn} - w_{mn}) \bigg/ \sum_{n=1}^{n_V} W_{mn} \tag{7}$$

where w_{mn} and W_{mn} denote the remaining and total bandwidth resources of direct link between node m and node n. Formula (6) shows the ratio of occupied CPU resources to total resources of node m, Formula (7) shows the ratio of occupied bandwidth resources to total resources of direct link between node m and node n, so that Formula (5) shows that if the CPU resources of a node and the bandwidth resources of adjacent links are occupied more, the load on the node is greater, and the VNFs on the node need to be migrated first. Specifically, we calculate the priority of all VNFs on the highest priority node in the following.

3.2.2. VNF priority

In view of the network failure, we assume that node m fails, so we need to migrate all VNFs on node m. On the other hand, in view of the network congestion, we assume that node m has the highest priority, so we need to migrate the VNF with highest priority. At first, since VNF depends on each SFC, we measure the priority of VNF by the priority of SFC. Specifically, we measure the priority by the CPU and bandwidth resources required of SFC, which is expressed as,

$$p_{sfc} = \frac{p_3 + p_4 + p_5 + p_6}{4} \tag{8}$$

$$p_3 = \sum_{i=1}^{l} rc_i \bigg/ m_{rc} \tag{9}$$

$$p_4 = \sum_{i=2}^{l} rb(f_{i-1}, f_i) \bigg/ m_{rb} \tag{10}$$

$$p_5 = \sum_{i=1}^{l} rc_i^m \bigg/ \sum_{i=1}^{l} rc_i \tag{11}$$

$$p_6 = \sum_{i=1}^{l} rb\left(f_{i-1}, f_i^m\right) \bigg/ \sum_{i=2}^{l} rb(f_{i-1}, f_i) \tag{12}$$

where rc_i denotes the required CPU resources of VNF f_i, $rb(f_{i-1}, f_i)$ denotes the required bandwidth resources of virtual link between f_{i-1} and f_i, m_{rc} and m_{rb} denote the maximum required CPU and bandwidth resources of all SFCs respectively, f_i^m denotes VNF f_i which is deployed on node m, rc_i^m indicates the required CPU resources of VNF f_i^m. Formula (9) shows the ratio between CPU resource requirements and maximum CPU resource requirements of all VNFs in an SFC, Formula (10) shows the ratio between bandwidth resource requirements and maximum bandwidth resource requirements of all virtual links in an SFC, Formula (11) shows the ratio between CPU resource requirements of VNFs deployed on node m and total CPU resource requirements in an SFC, Formula (12) shows the ratio between bandwidth resource requirements of links passing node m and total bandwidth resource requirements in an SFC, so that Formula (8) shows that the higher the sum of the total resource requirements of an SFC and the resource requirements related to the node m is, the higher the priority of the SFC is. Then, all SFCs are sorted in descending order according to the priority, and the SFCs without passing node m are removed. Finally, the VNFs deployed on node m in each SFC are sorted, specifically, if VNF i is prior to VNF j, the former VNF has higher priority than the latter.

3.3. VNF migration scheme

The VNF migration scheme is to decide how to select the appropriate node and choose the appropriate VNF for migration. In this work, we consider network congestion and node failure respectively.

3.3.1. Network congestion

In the state of network congestion, we mainly consider the congestion of nodes. At first, we calculate the priority of each node according to Formulas (5)-(7). Then, we select the node with the highest priority as the pending node m. Thirdly, we calculate the priority of each VNF on node m according to Formulas (8)-(12). Fourthly, the model selects the VNF with the highest priority to migrate according to the migration algorithm. Finally, repeat the above process until the number of iterations reaches the upper limit.

3.3.2. Node failure

In the state of node failure, we mainly consider the case of single node failure. Firstly, we calculate the priority of all SFCs according to Formulas (8)-(12). Then, we remove the SFCs which do not pass through the failed node m. Thirdly, the model arranges the remaining SFCs in descending order according to the priority. Fourthly, the VNFs of each SFC deployed on node m are arranged in the order of dependence. Finally, the model migrates the arranged VNFs according to the algorithm.

3.4. Deep reinforcement learning migration algorithm

Deep reinforcement learning migration algorithm aims to solve the VNF migration problem which has large-scale network state and action set.

3.4.1. State set

In network topology, the network state is mainly represented by the state of nodes and links and the selected node, so in this work, we define the state set as,

$$S(t) = (cpu_t, bw_t, m) \tag{13}$$

$$cpu_t = (n_1, n_2, \ldots, n_{n_V}) \tag{14}$$

$$bw_t = (l_1, l_2, \ldots, l_{n_E}) \tag{15}$$

where node m represents the node with the highest priority or the failure node in the two network states respectively, where n_i denotes the remaining CPU resources of node i, l_i denotes the remaining bandwidth resources of link i. Specifically, Formula (14) and (15) represent the remaining resources of all nodes and links at time t respectively, Formula (13) shows the network state at time t. In particular, if node m is a failure node, then $cpu_m = 0$.

3.4.2. Action set

Action set represents a set of all optional actions. In this work, we define the action set in the case of network congestion and node failure respectively.

In the case of network congestion, we need to migrate the VNF with highest priority on the highest priority node to another node. In the case of node failure, we need to migrate all VNFs on the failed node to other nodes, so we define the action set as,

$$A(t) = n_m \tag{16}$$

so, the number of action set A is $n_V - 1$.

3.4.3. Feedback function

In this work, we aim to optimize the transmission delay of SFC and network load balancing, including load balancing of nodes and links. To this end, we optimize all optimization targets by minimizing the value of delay, maximum node load ratio, and maximum link load ratio. Specifically, we define the delay as the transmission delay sum between the migration node m and deployment nodes of adjacent VNF, which is expressed as,

$$D = d\left(f_{i-1}, f_i^m\right) + d\left(f_i^m, f_{i+1}\right) \tag{17}$$

Then, we define the maximum node load ratio as,

$$U = \max \frac{U_i - u_i}{U_i} \tag{18}$$

And we define the maximum link load ratio as,

$$W = \max \frac{W_i - w_i}{W_i} \tag{19}$$

So, we define the feedback function as,

$$R(t) = \frac{\alpha}{D} + \frac{\beta}{U} + \frac{\gamma}{W} \tag{20}$$

where α, β, γ are the weight coefficients, which represent the importance of delay, node load ratio and link load ratio in the feedback function respectively. For the convenience of calculation, we set the sum of the three weight coefficients as 1, so they need to meet this constraint,

$$\begin{cases} \alpha > 0, \beta > 0, \gamma > 0 \\ \alpha + \beta + \gamma = 1 \end{cases} \tag{21}$$

3.4.4. VNF migration algorithm

VNF migration algorithm is aimed to select the appropriate scheme for different network states, including network congestion and node failure. At first, we determine the network state. Then, we choose the corresponding priority calculation scheme according the network state. Thirdly, all the parameters of neural network are initialized. Next, we choose action through $\varepsilon - greedy$ method, which is defined as,

$$\varepsilon = \frac{K}{100} \tag{22}$$

where K denotes the exploration times. Specifically, in the $\varepsilon - greedy$ method, the larger the value of ε, the smaller the probability of using random action, and the greater the probability of using the action with the largest value of current Q function, and the number 100 is the empirical value of our previous experiments. Finally, we store a 5-tuple $(S(t), A(t), S(t+1), R(t), is_{end})$ in replay memory buffer after every action selection, where is_{end} is a binary indicates whether the state is terminated. Specifically, when the migrated node or link resources are insufficient, or the last VNF of an SFC has been processed, the state is terminated, conversely, the state is not terminated, which is expressed as,

$$\begin{cases} is_{end} = 1, \ state\ is\ terminated \\ is_{end} = 0, \ state\ is\ not\ terminated \end{cases} \tag{23}$$

The network randomly selects several samples from D in each episode, then the Q value will be calculated by,

$$Q(t) = \begin{cases} R(t), \ while\ is_{end} = 1 \\ R(t) + \\ \delta * \max(Q(S(t+1), A(t+1), \omega)) \\ , \ while\ is_{end} = 0 \end{cases} \tag{24}$$

where ω is the parameter of neural network, which will be updated in each episode by backward gradient propagation, and δ represents the learning rate. The training algorithm for our model is as Algorithm 1.

4. Simulation results and analysis

In this section, the network topology is introduced firstly, and then the parameters are described, and finally the optimization performance and analysis are presented and discussed.

4.1. Network topology

To formulate the deployment process of SFC and VNF migration process, we use Internet2 network topology to simulate, which has 12 network nodes and 15 physical links. In practice, the link delay is mainly related to the transmission distance, and we cannot accurately measure it due to the limitations of experimental conditions. Therefore, in order to measure the proportional relationship between different link delays, we randomly generate the transmission delay of each link in (0,1), generate the CPU resources of each node and bandwidth resources and of each link in [75,100]. The network topology diagram is showed in Fig. 1. The black number represents CPU capacity of each network node, the red and blue numbers next by solid line represent bandwidth capacity and transmission delay of each physical link respectively.

4.2. Experimental settings

In this subsection we introduce the formulation environment firstly, including hardware and software platform, then, several state-of-the-art

Algorithm 1
Pseudo code of PAVM algorithm.

```
Algorithm 1 PAVM training algorithm
1:  Input: SFC request with the requirement of VNFs and virtual links
2:  Output: Deployment node and migration location of VNFs to be migrated
3:  Initialize the neural network N with the parameter $\omega$, and clear the replay
    memory buffer
4:  while $is_{end} \neq 0$ do
5:  Determine network state in network congestion and node failure
6:  if in network congestion state
7:  Calculate node priority level $p_{node}$ as (5)-(7)
8:  Calculate SFC priority level $p_{sfc}$ as (8)-(12)
9:  Determine the highest priority VNF on the node with highest priority
10: else in node failure state
11: Calculate SFC priority level $p_{sfc}$ as (8)-(12)
12: Determine the VNF in order of priority from high to low
13: end if
14: Select action $A(t)$ by $\varepsilon - greedy$ method
15: Calculate reward $R(t)$ as (17)-(21)
16: Store the sample $(S(t), A(t), S(t+1), R(t), is_{end})$ in replay memory buffer
17: Randomly select samples from replay memory buffer then calculate $Q$ as (24)
18: Train the network N and update $\omega$ by using back propagation of the loss
    function
19: end while
```

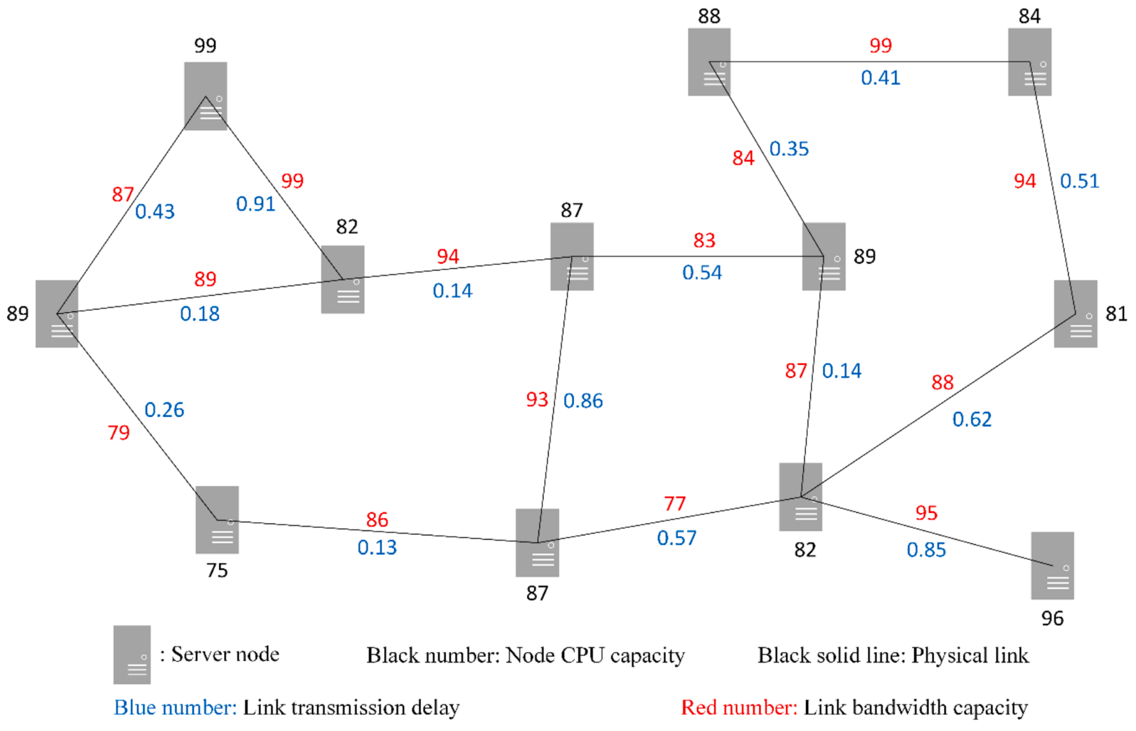

99
82
87
89
88
84
81
75
87
82
96
89
87
0.43
99
0.91
89
0.18
94
0.14
83
0.54
99
0.41
84
0.35
94
0.51
0.26
79
93
0.86
87
0.14
88
0.62
86
0.13
77
0.57
95
0.85
: Server node
Black number: Node CPU capacity
Black solid line: Physical link
Blue number: Link transmission delay
Red number: Link bandwidth capacity

Fig. 1. Internet2 topology with random resources and transmission delays.

comparison algorithms are described in detail, and finally we set related parameters of experiment and algorithm respectively.

4.2.1. *Formulation environment*

The experiments are implemented with MATLAB platform on a 64-bit computer with Intel Core CPU i5–9400F @ 2.9 GHz, 16-GB RAM. In order to simulate the SFC deployment process and ensure a reasonable number of resources, we set two cases of SFC numbers. Specifically, case 1 generates 50 SFCs randomly, case 2 generates 80 SFCs randomly, then deploy SFC by randomly selecting locations firstly. In the training process, in order to avoid linear results due to linear episode numbers, we set the training episode numbers as 100, 200, 300, 400, 500, 1000, 1500, 2000, 2500, 3000, 4000, 6000, 8000, 10,000 respectively in the network congestion state, and set episode numbers as 10, 20, 30, 40, 50, 100, 150, 200, 250, 300, 400, 600, 800, 1000 respectively in the node failure state, which can ensure that the training time will not be too long and performance will achieve the goal.

4.2.2. *Comparison algorithms*

In order to measure the performance of PAVM algorithm which is proposed in this work, two state-of-the-art algorithms including PA-RL, NLAVMM are selected as the baselines. Specifically, the first comparison

Table 1
Algorithm comparison of related works.

Type of problem	Reference	Basic algorithm	Advantage	Disadvantage
SFC deployment	[6]	linear programming	improves reliability and network load balancing	cannot guarantee performance of transmission delay
	[7]		reduces the inherent cost	cannot to adapt to dynamic network environment
	[8]		protects SFC requests from network failures	does not guarantee the delay
	[9]		improves reliability	does not consider transmission delay and network balancing
	[10]		achieves high reliability and low latency	hard to satisfy complex and dynamic network environment
	[11]	heuristic algorithm	addresses the joint topology design and mapping of reliable SFC deployment problem	does not consider the delay and load balancing
	[12]	deep reinforcement learning	guarantees seamless redirection after failures and ensure service reliability	does not distinguish the priority of SFC and cannot guarantee transmission delay and network load balancing
	[13]		distinguishes priority and improves reliability	does not consider the possible network state changing
VNF migration	[14]	linear programming	improves the reliability and load balance and optimizes the resource allocation	does not consider the failure of network nodes
	[15]		reduces the total energy consumption	does not consider the delay and load balancing
	[16]		reduces delay in changing resource availability	does not consider node failure
	[17]		achieves load balancing	does not consider the network failure and lacks the optimization of transmission delay
	[18]	reinforcement learning	optimizes the network load	search time consumption is large, optimization objective is single, priority is not considered
	[19]		achieves near-optimal performance	search time consumption is large
	[20]	deep learning	resolves the problem of lacking effective prediction	does not distinguish the priority
	[21]	deep reinforcement learning	addresses multiple SFC migration	does not optimize the network load balance and the transmission delay and does not consider the network failure situation

algorithm is PA-RL, which is the result of applying priority-awareness model to RL proposed in [19]. The second comparison algorithm is NLAVMM, which is designed for node-aware and link-aware situations in [17]. In addition, we implement two versions of our solution. One is RNDVM, which randomly determines VNF to migrate with deep reinforcement learning algorithm. The other is PAVM, which determines VNF according to priority-awareness model to migrate with deep reinforcement learning as mentioned before.

4.2.3. Related parameters

We assume there are five varieties of VNFs, including Firewall (FW), Net Address Transition (NAT), Deep Packet Inspection (DPI), Domain Name System (DNS), and Load Balancing (LB). The required resources of each VNFs and virtual links are presented asTables 2 and 3. For example, the SFC FW-NAT-DPI has 4 units CPU and 3 units bandwidth resources requirement respectively. So that, the SFC NAT-DPI-DNS-LB-FW has most resources requirement, the required CPU and bandwidth resources of this SFC are 10 units CPUs and 10 units bandwidths respectively.

4.2.4. Parameters of algorithm

We set the parameters of feedback function, neural network and replay buffer through the experience obtained from some previous experiments, so that the model can converge successfully without consuming too long. On one hand, through several times of adjusting according to the experimental performance, we set the parameters of feedback function (22) as $\alpha = 0.8, \beta = 0.1$, and $\gamma = 0.1$ respectively, and then, we set up two hidden layers with 100 neurons for the neural network. On the other hand, in the training process of algorithm, the learning rate of deep reinforcement learning is usually set as [0.9, 0.999], so that the learning effect will be better than other value the learning rate is. In this work, we set the learning rate δ of proposed deep reinforcement learning process as $\delta = 0.99$.

Moreover, we set the size of replay memory buffer is 10^6, and the number of samples for each training episode is 100, which is selected from replay memory buffer randomly, so as to ensure that the impaction between samples is small and the distribution of samples is uniform.

4.3. Experimental results

We get the results of runtime, delay optimization rate, nodes' load balancing variance, and links' load balancing variance under the formulation environment as mentioned before.

4.3.1. Runtime

Fig. 2 shows the runtime of PAVM and PARL algorithms in two network states respectively. Specifically, Fig. 2(a) shows the runtime of two algorithms in two cases in the state of network congestion. It's obvious that the runtime of PAVM is much less than PARL in both cases, this is because PARL uses reinforcement learning algorithm, which needs to maintain a huge Q table. The method needs to query the Q table every time it selects an action, querying the Q table needs to traverse the Q values of all actions in a specific state, the larger the Q table is, the longer the traversal time is. Therefore, it will consume a lot of time. On the contrary, PAVM uses deep reinforcement learning algorithm, which simulates Q value through neural network and does not need to maintain and query huge Q table, which greatly shortens the runtime of the algorithm. It only needs to fit the Q value through the representation ability of neural network, and the time consumption is basically irrelevant to the size of state and action space. On the other hand, we can clearly see that the runtime of case 1 is less than that of case 2 for two algorithms. This is because case 1 has fewer SFCS and consumes less network resources, so the processing time is less than that of case 2. For the same reason, in Fig. 2(b), the runtime of each algorithm is similar to that in Fig. 2(a) in the node failure status.

Table 2
All VNFs and their respective CPU resource requirements.

Varieties of VNF	CPU resource requirements
FW	1 unit
NAT	1 unit
DPI	2 units
DNS	3 units
LB	3 units

Table 3
All virtual links and their respective bandwidth resource requirements.

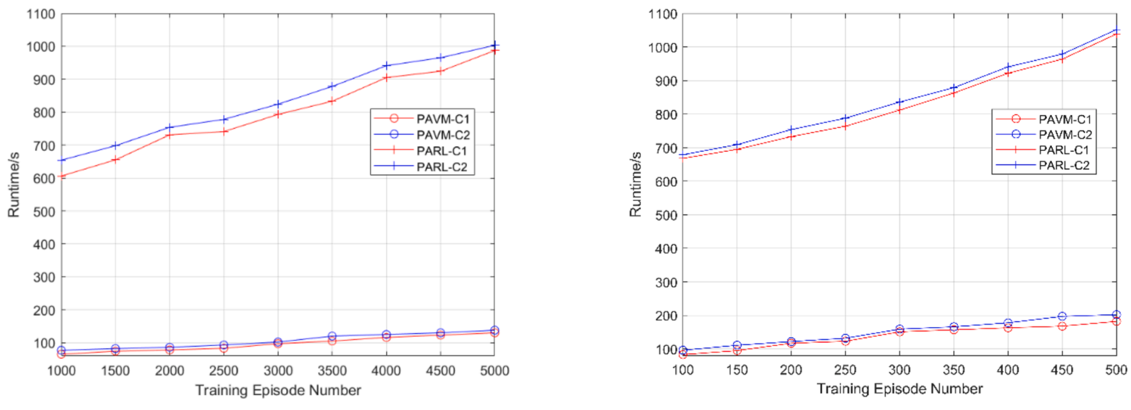
Varieties of virtual links between adjacent VNFs	Bandwidth resource requirements
FW-NAT (NAT-FW)	1 unit
FW-DPI (DPI-FW)	1 unit
FW-DNS (DNS-FW)	2 units
FW-LB (LB-FW)	2 units
NAT-DPI (DPI-NAT)	2 units
NAT-DNS (DNS-NAT)	2 units
NAT-LB (LB-NAT)	2 units
DPI-DNS (DNS-DPI)	3 units
DPI-LB (LB-DPI)	2 units
DNS-LB (LB-DNS)	3 units

4.3.2. Average delay between adjacent VNFs

Fig. 3 shows the average transmission delay between adjacent VNFs of four algorithms in two cases and two states respectively. For Fig. 3(a), it's easy to observe four facts. At first, the average delay of NLAVMM is the largest, this is because NLAVMM's main optimization goal is network load balancing, it does not consider the delay factor, and NLAVMM is a static algorithm, it cannot improve the optimization performance gradually like deep reinforcement learning and reinforcement learning. Therefore, the performance is the worst under each episode number. Secondly, the average delay of RNDVM is higher than PARL and PAVM, because the latter two algorithms use the priority-awareness model, which can migrate the more important VNF preferentially, and they define transmission delay as one feedback factor of feedback function, so the two algorithms are more inclined to migrate VNF to approaching node. Thirdly, this result agrees with the theoretical analysis above that the proposed PAVM algorithm is better than PARL, this is because we use deep reinforcement learning algorithm which is more advanced than reinforcement learning, if reinforcement learning method is used, many states will not appear, and it will be difficult for the algorithm to select the optimal action when encountering these states. However, deep reinforcement learning method can simulate the possible states through neural network and make effective action decisions for unknown states, so the performance of deep reinforcement learning is better than reinforcement learning. Last but not least, the average delay of PAVM becomes smaller and smaller at first, and then tends to be stable, because the training data is too small to make the algorithm converge in the early stage of training, the performance gradually improves. When the training data reaches a certain degree, the algorithm completes convergence and the performance basically tends to be stable. Similarly, observing the results in Fig. 3(b), (c) and (d), it is easy to discover that the relationship between the delay of four algorithms is the same as that of Fig. 3(a).

4.3.3. Load balancing of network nodes

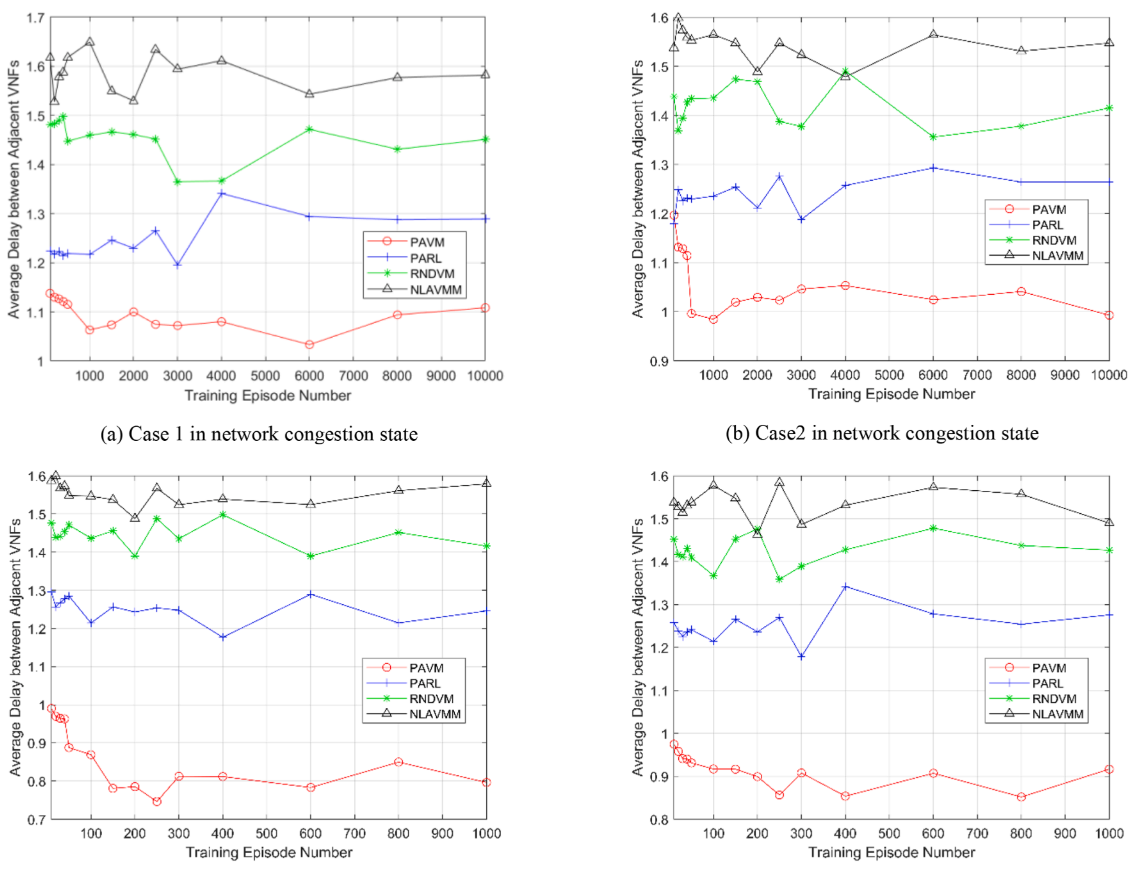
Fig. 4 shows the network node load variance of the four algorithms in two cases and two states respectively. For Fig. 4(a), it is obvious that there are five facts. First of all, the node load variance of PAVM is the lowest, because PAVM uses priority-awareness model, it can migrate VNF which takes more resources on nodes with large load, so PAVM is easier to achieve node load balancing. Secondly, the performance of PARL is worse than PAVM, this is because compared with reinforcement learning algorithm, deep reinforcement learning algorithm can predict



(a) Network congestion state

(b) Node failure state

Fig. 2. Runtime of PAVM and PARL in two network states.



(a) Case 1 in network congestion state

(b) Case2 in network congestion state

(c) Case 1 in node failure state

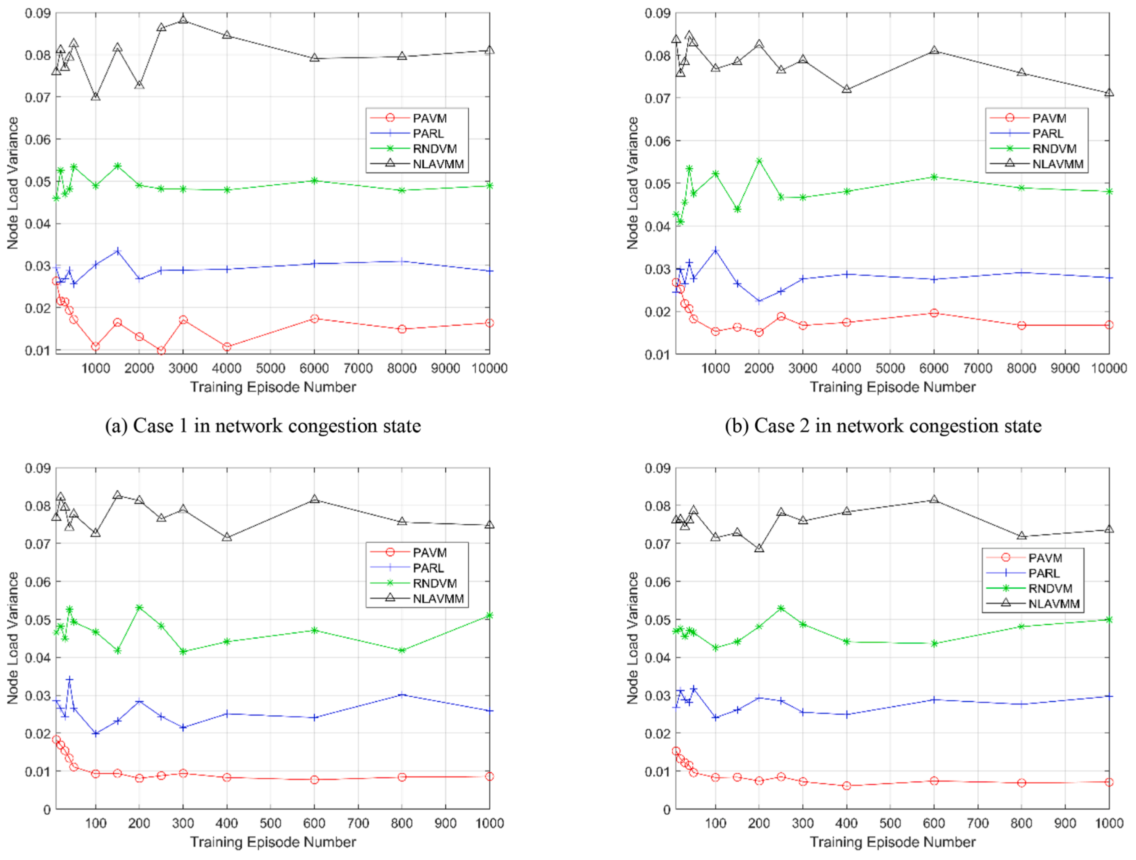
(d) Case 2 in node failure state

Fig. 3. Average delay between adjacent VNFS in two cases under two network states.

the action selection of unknown state with higher accuracy, therefore the node load variance of PARL is higher than PAVM. Thirdly, the node load variance of RNDVM is higher than PARL and PAVM, because RNDVM does not use the priority-awareness model, the algorithm may choose the node with less load and VNF to migrate in a random way, so the performance is worse than the two algorithms with priority-awareness model. Fourthly, the node load variance of NLAVMM is the highest, this is because NLAVMM does not use the priority-awareness model, and it operates on nodes and links separately, it is difficult to ensure that nodes and links achieve load balancing effect at the same time. Finally, the performance of PAVM if getting better at first, and then tends to be stable, because the training data is too small to make the algorithm converge in the early stage of training, the performance gradually improves. Then the algorithm completes convergence and the performance basically tends to be stable when the training data reaches a certain degree. Similarly, observing the results in Fig. 4(b), (c) and (d), it is easy to discover that the relationship between the node load variance of four algorithms is the same as that of Fig. 4(a).

4.3.4. Load balancing of network links

Fig. 5 shows the network link load variance of the four algorithms in two cases and two states respectively. For Fig. 5(a), it is clear to observe



(a) Case 1 in network congestion state (b) Case 2 in network congestion state

(c) Case 1 in node failure state (d) Case 2 in node failure state

Fig. 4. Node load variance of four algorithms in two cases under two network states.

five facts. Firstly, the link load variance of NLAVMM is the highest, this is because NLAVMM does not use the priority-awareness model proposed in this work, it is difficult to select the most suitable VNF for migration, moreover, the algorithm optimizes the node and link load separately, and the comprehensive performance is difficult to guarantee, so the performance of NLAVMM is the worst. Next, the link load variance of RNDVM is higher than PAVM, because RNDVM randomly selects VNF for migration, although it uses link load as a feedback index, yet it cannot achieve the optimal effect. Thirdly, the performance of PARL is worse than PAVM, this is because compared with deep reinforcement learning algorithm, reinforcement learning algorithm is difficult to predict the unknown state and select the appropriate action for it, so the link load variance of PARL is higher. Then, the link load variance of PAVM is the lowest, because PAVM utilizes the priority-awareness model, which can select the VNF with the highest priority on the node with the highest load priority to migrate, so that the virtual link on the high loaded link can be migrated to the low loaded link at the same time. Last but not least, the link load variance of PAVM becomes smaller and smaller at first, and then tends to be stable, this is because the training data is too small to make the algorithm converge in the early stage of training, so the performance gradually improves. When the training data becomes bigger, the algorithm completes convergence and the performance is basically optimal. Similarly, observing the results in Fig. 5(b), (c) and (d), it is easy to discover that the relationship between the link load variance of four algorithms is the same as that of Fig. 5(a).

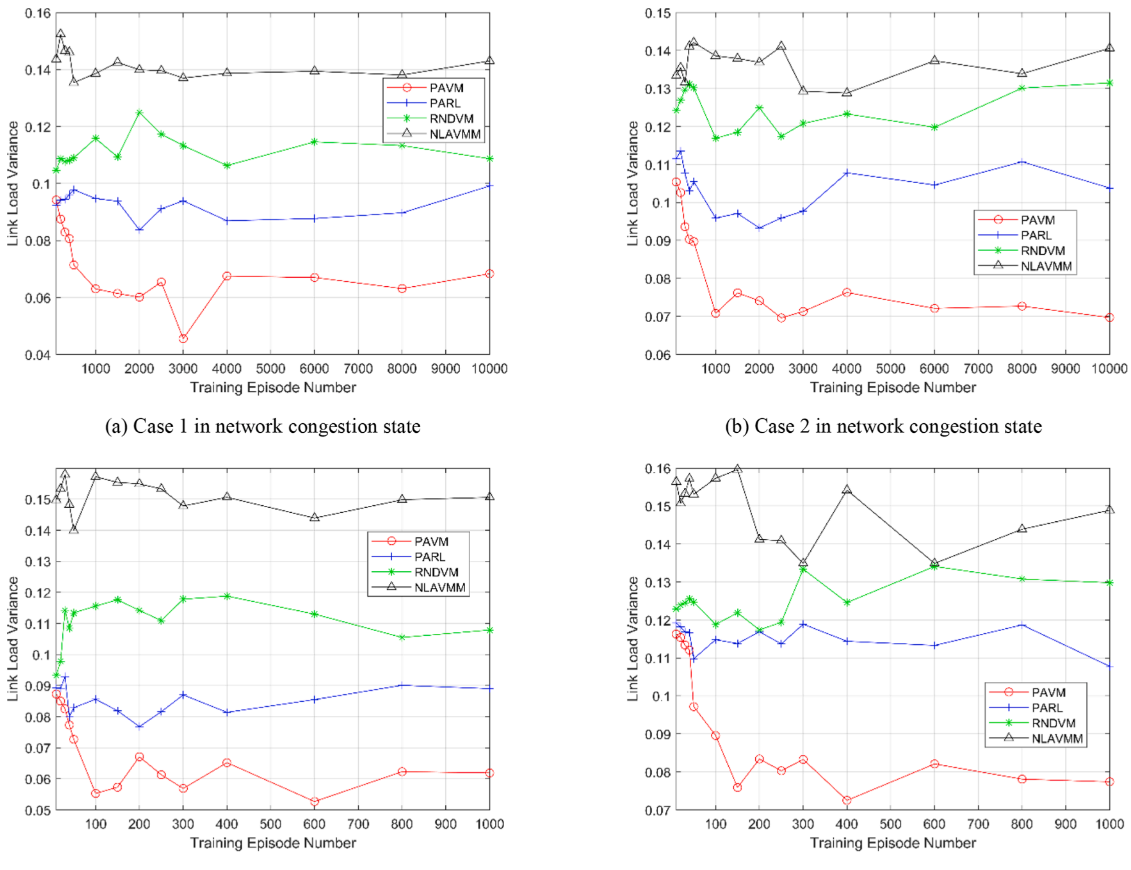
5. Conclusion

In this paper, we proposed a novel PAVM algorithm for reliable and optimal VNF migration. We distinguished two kinds of VNF migration schemes for network congestion and node failure states respectively, and designed node and VNF priority-awareness mechanism, which could select appropriate VNF on suitable node to migrate. Moreover, PAVM utilized deep reinforcement learning algorithm to choose target node location of VNF migration, and updated the neural network dynamically. The experimental results showed that compared with the other three algorithms (NLAVMM, PARL, and RNDVM), PAVM used transmission delay and network load balancing as feedback factors, so that it could reduce the transmission delay, and improved the load balancing of nodes and links to a certain extent. In addition, compared with PARL algorithm, PAVM could reduce the running time of the algorithm by a wide margin.

Although our proposed PAVM model can achieve reliable and optimal VNF migration, yet there are still some shortcomings that need to be addressed. Firstly, the parameters of neural network and deep reinforcement learning, including hidden layers, neurons number, and coefficients were designed through minor adjustments, so we could not guarantee the performance is the best. Therefore, we need to adjust the parameters through more experimental comparison in the future work. Next, the single VNF migration mechanism did not consider the possible VNF sharing. When part of VNFs work, the other part of VNFs are idle at the same time, which is easily to cause a waste of network resources. Therefore, we will further consider the VNF sharing mechanism in the follow-up work.

CRediT authorship contribution statement

Hua Qu: Funding acquisition, Writing – review & editing. **Ke Wang:** Conceptualization, Investigation, Methodology, Writing – original draft. **Jihong Zhao:** Funding acquisition, Writing – review & editing.



(a) Case 1 in network congestion state

(b) Case 2 in network congestion state

(c) Case 1 in node failure state

(d) Case 2 in node failure state

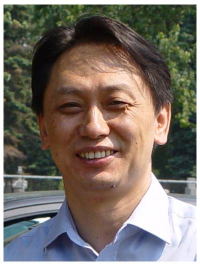
Fig. 5. Link load variance of four algorithms in two cases under two network states.

Declaration of Competing Interest

The authors declare no conflict of interest.
All authors read and contributed to the manuscript.

References

[1] B. Yi, X. Wang, K. Li, S.K. Das, M. Huang, A comprehensive survey of network function virtualization, Comput. Netw. 133 (2018) 212–262.
[2] R. Mijumbi, J. Serrat, G. Juan-Luis, N. Bouten, F.D. Turck, R. Boutaba, Network function virtualization: state-of-the-art and research challenges, IEEE Commun. Surv. Tutor. 18 (1) (2017) 236–262.
[3] J.M. Halpern, C. Pignataro, Service Function Chaining (SFC) Architecture, RFC 7665, RFC Editor, 2015.
[4] Y. Li, M. Chen, Software-defined network function virtualization: a survey, IEEE Access 3 (2017) 2542–2553.
[5] X. Wang, X. Chen, C. Yuen, W. Wu, M. Zhang, C. Zhan, Delay-cost tradeoff for virtual machine migration in cloud data centers, J. Netw. Comput. Appl. 78 (2017) 62–72.
[6] F. Carpio, A. Jukan, Improving reliability of service function chains with combined VNF migrations and replications. (2017).
[7] A. Abdelhamid, A. Toufik, M. Mohamed, B. Raouf, Joint diversity and redundancy for resilient service function chain provisioning, IEEE J. Sel. Areas Commun. 38 (2020) 1490–1504.
[8] P. Tuan-Minh, F. Serge, N. Thi-Thuy-Lien, C. Hoai-Nam, Modeling and analysis of robust service composition for network functions virtualization, Comput. Netw. 166 (2020).
[9] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, D. Niyat o, Reliability aware service placement using a viterbi-based algorithm, IEEE Trans. Netw. Serv. Manag. 17 (2020) 622–636.
[10] L. Qu, C. Assi, M.J. Khabbaz, Y. Ye, Reliability-aware service function chaining with function decomposition and multipath routing, IEEE Trans. Netw. Serv. Manag. 17 (2019) 835–848.
[11] Z. long Ye, X. jun Cao, J. ping Wang, H. fang Yu, C. ming Qiao, Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization, IEEE Netw. 30 (2016) 81–87.
[12] W. Mao, L. Wang, J. Zhao, Y. Xu, Online fault-tolerant VNF chain placement: a deep reinforcement learning approach, in: 2020 IFIP Networking Conference (Networking), Paris, France, 2020.
[13] H. Qu, K. Wang, J.H. Zhao, Reliable service function chain deployment method based on deep reinforcement learning, Sensors 21 (8) (2021) 2733.
[14] S. Xu, B. Liao, B. Hu, C. Han, A. Xiong, A reliability-and-energy-balanced service function chain mapping and migration method for internet of things, IEEE Access 8 (2020) 168196–168209.
[15] V. Eramo, E. Miucci, M. Ammar, F.G. Lavacca, An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures, IEEE/ACM Trans. Network. 25 (4) (2017) 2008–2025.
[16] D. Cho, J. Taheri, A.Y. Zomaya, P. Bouvry, Real-time virtual network function (VNF) migration toward low network latency in cloud environments, in: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), IEEE Computer Society, 2017.
[17] B. Yi, X. Wang, M. Huang, K. Li, Design and implementation of network-aware vnf migration mechanism, IEEE Access 8 (2020) 44346–44358.
[18] L.J. Xie, Q. Zhou, J.Y. Huang, Dynamic service chain migration for network functions virtualization, in: 2017 International Conference on Network and Information Systems for Computers (ICNISC, 2017.
[19] J. Li, W. Shi, N. Zhang, X.S. Shen, Reinforcement learning based VNF scheduling with end-to-end delay guarantee, in: 2019 IEEE/CIC International Conference on Communications in China (ICCC), IEEE, 2019.
[20] L. Tang, X. He, P. Zhao, G. Zhao, Y. Zhou, Q. Chen, Virtual network function migration based on dynamic resource requirements prediction, IEEE Access (2019) 99.
[21] R. Chen, H. Lu, Y. Lu, J. Liu, MSDF: a deep reinforcement learning framework for service function chain migration, in: 2020 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2020.



Qu Hua (1961-), received the B.Eng. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, and the Ph.D. degree from Xi'an Jiaotong University, Xi'an, China. He is currently a Professor with Xi'an Jiaotong University. His research interests include mobile Internet, IP-based network, network management and control, and radio resource management in LTE-A system. He is a Senior Member of the China Institute of Communications and also an Editor of China Communications Magazine.

Wang Ke (1992 -), received the B.Eng. degree in software engineering from Xi'an Jiaotong University, Xi'an, China, in 2014, where he is currently pursuing the Ph.D. degree with the School of Software Engineering. His current research interests include analysis of the software defined network, network function virtualization, service function chain.

Zhao Jihong (1963-), received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, and the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China. She is currently a Professor with Xi'an Jiaotong University. She also researches broadband communication network, management and control of new generation network and machine learning for network management, and device-to-device communications.