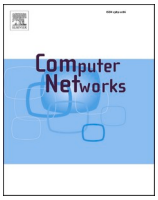
Computer Networks 246 (2024) 110418

Contents lists available at ScienceDirect

Computer Networks

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



Low-latency Virtual Network function Scheduling Algorithm Based on Deep Reinforcement Learning*

Zhiwei Liu, Zhaogang Shu *, Shuwu Chen, Yiwen Zhong, Jiaxiang Lin

Computer and Information College, Fujian Agriculture and Forestry University, Fuzhou, China

ARTICLE INFO

Key words:
Service function chain
Virtual network functions
Delay-aware
VNF scheduling
Deep reinforcement learning

ABSTRACT

This paper addresses the problem of mapping, scheduling, and routing of virtual network functions (VNF) on a service function chain (SFC) that is sensitive to latency in a virtual network. A scheduling algorithm for VNF is proposed, which aims to minimize the SFC rejection rate while taking into account VNF mapping, scheduling, and traffic routing during the scheduling process. To achieve this goal, a Markov decision process (MDP)-based VNF scheduling model is established that guarantees SFC resource requirements are met. The model uses the D3QN (Dueling Double DQN) algorithm based on composite rules to select the SFC at each scheduling time point, and selects virtual nodes and routes using a routing optimization algorithm to minimize the SFC rejection rate. We compare our algorithm with the single rule, DQN and genetic algorithm, and the simulation results show that the proposed algorithm can reduce the rejection rate of SFC by approximately 8% compared to genetic algorithms.

1. Introduction

With the development of network technology, 5G networks have been further upgraded compared to traditional networks, resulting in diversified network services. At the same time, the number of low-latency networks is growing exponentially. Meeting the growing and diverse needs of users for the network is currently the focus of the communication industry.

In the traditional static network architecture, there are mainly two problems. Firstly, the services provided by the network, such as firewalls and WAN optimizers, are tightly coupled with hardware called middleboxes [1]. Different network functions require different hardware, resulting in inflexible network functions and difficult maintenance, which requires a significant amount of operational and capital expenditures [2,3]. Secondly, the static network mode cannot meet the differentiated performance requirements of new applications. To address these problems, Network Function Virtualization (NFV) has been introduced in 5G networks. The main function of NFV is to decouple hardware and software from proprietary devices, making software independent of any proprietary hardware. The decoupled software is abstracted into independent network modules, called Virtualized Network Functions (VNF) [4]. These VNF can be adaptively placed on physical resources to provide the network node with the corresponding VNF function. Therefore, based on a reliable VNF architecture in the network, it not only improves the flexibility of the network, meets QOS requirements, makes reasonable use of network resources, but also offsets dedicated hardware devices, thereby reducing operators' operational and capital expenditures [5,6].

In the architecture of NFV, a Service Function Chain (SFC) is formed by several VNF instances arranged in a certain order to provide network services on the network infrastructure [7]. However, configuring SFC on NFV-supported network infrastructure is not a simple task, especially for delay-sensitive SFC (e.g., tactile internet services), as these SFC need to be combined in a specific order and completed within strict service deadlines [8]. To meet such strict timing requirements, service providers must effectively perform VNF placement and scheduling as well as traffic routing for these SFC, a challenge also known as NFV resource allocation (NFV-RA) [2,9]. Generally, the NFV-RA problem can be divided into three main sub-problems: (a) VNF composition, (b) VNF placement, and (c) VNF scheduling. The first sub-problem involves the composition of SFC, the second sub-problem, aims to place the VNF in the SFC onto nodes that support NFV and map the virtual links between

* This research work was supported in part by the Fujian Province Natural Science Foundation of China under Agreement 2020J01574 and Industry-university-research Innovation Fund of China under Agreement 2021FNA05003.

* Corresponding Author
E-mail address: zgshu@fafu.edu.cn (Z. Shu).

https://doi.org/10.1016/j.comnet.2024.110418
Received 13 June 2023; Received in revised form 16 February 2024; Accepted 9 April 2024
Available online 10 April 2024

VNF to the underlying links. The third sub-problem focuses on determining the execution plan for the VNF in the SFC required to run a given service.

Although the NFV-RA problem consists of three problems, the placement and scheduling of VNF have always been the main research focus [9]. However, most papers address VNF placement and scheduling separately, which may not meet strict service deadlines in practical situations. For example, [11,12,13] mainly solves the problem of VNF placement, and secondly, there are also many problem models and solutions for VNF scheduling, such as [8,19], which solve the delay sensitive NS scheduling problem. In addition to individual processing, there are also studies on the joint solution of VNF placement and scheduling problems, such as references [20,21,22]. However, it should be noted that in the above studies, network service scheduling did not consider network routing and business transmission delay, which is an important factor affecting delay sensitive SFC in practical environments.

In summary, there are two main issues in most existing research papers on VNF scheduling. Firstly, VNF scheduling and VNF placement are discussed separately. Secondly, the transmission delay of network services in the network is not considered. Both of these issues are important factors in delay sensitive SFC in practical situations. To address these two issues, this article models and solves the VNF placement and scheduling problem as a whole, facing complex challenges, especially when considering traffic routing and transmission delay. This is a very complex combinatorial optimization problem. Some papers have developed various metaheuristic algorithms [25,29,30]. Although these heuristic algorithms typically have fast implementation and simplicity, their performance is largely influenced by problem characteristics and may decrease with increasing network size. In addition, in terms of runtime, metaheuristic algorithms such as genetic algorithms (GA) may experience slower convergence rates during the iteration process, especially when dealing with complex problems, resulting in increased computational costs and runtime, and may prematurely fall into local optima. Therefore, this article first represents the low latency VNF scheduling problem as NP Hard's mixed integer linear programming (MILP), Then, the original problem is rephrased as a Markov decision process problem, and a deep reinforcement learning framework is proposed to solve this problem. Of course, the use of deep reinforcement learning frameworks to set network parameters and states, actions, and reward sets ensures that intelligent agents can make correct decisions in different environments, which is also a complex challenge.

In this regard, the deep reinforcement learning scheduling algorithm proposed in this article solves the placement and scheduling problems of VNF, ensuring that delay sensitive network services are completed within strict service deadlines to minimize the total number of unfinished SFC. The main contributions are as follows:

A: This paper formulates the VNF scheduling problem as a Mixed-Integer Linear Programming (MILP) problem. The problem takes into account the joint mapping and scheduling of VNF while considering route optimization. The objective is to improve the acceptance rate of SFC and reduce routing overhead, all while meeting the strict service deadlines of delay-sensitive SFC.

B: Design the state, action, and reward models for the D3QN to enable the network to choose appropriate actions based on different current states, aiming to maximize rewards. Additionally, integrate the D3QN network with heuristic algorithms for route optimization, with the goal of minimizing transmission time and routing overhead while meeting SFC requirements as closely as possible.

C: Demonstrate the convergence of the proposed deep reinforcement learning algorithm through an extensive set of experiments. Compare the performance of the proposed method with metaheuristic algorithms, composite scheduling rules, and standard DQN. Simulation results indicate that the proposed approach outperforms in effectively addressing the delay-aware VNF scheduling problem.

The remainder of this paper is organized as follows. Section 2 describes related work,Section 3 describes the problem and divides it into several sub-problems, while also discussing the interactions and impacts between these sub problem problems and scheduling. Section 4 defines the problem and proposes a rule-based deep reinforcement learning scheduling model, and explains the methods for solving the problems of VNF mapping, scheduling, and traffic routing. The results of numerical experiments are given in Section 5. Finally, conclusions are drawn in Section 6.

2. Related Work

In the context where multiple network functions can be accommodated within a single node, and there are multiple VNF to be deployed across the network, the rational placement of VNF on network nodes to address issues such as VNF reusability or sharing for cost reduction poses a significant challenge known as the VNF Placement Problem. For instance, researchers have delved into the VNF Placement Problem with reliability considerations, formulating two protection mechanisms as Integer Linear Programming (ILP) models and proposing a Dynamic Programming-based heuristic algorithm [10]. Hyodo et al. [11] formalized the VNF Placement Problem as an ILP model and introduced a heuristic algorithm to minimize layout and link costs, while allowing for flexibility in VNF access sequence and cyclic SFC configurations. Alahmad and Agarwal [12] presented two Mixed-Integer Linear Programming (MILP) models addressing the cost and availability aspects of VNF placement and type selection. In comparison to existing solutions, their proposed approach reduces the overall cost of requested network services without violating availability requirements. Feng et al. [13] introduced an advanced heuristic algorithm involving VNF migration to an alternative available node, effectively enhancing SFC utilization and acceptance rates. Recognizing the limitations of existing deep reinforcement learning in generalizing across diverse network topologies, Sun et al. [14] combined deep reinforcement learning with neural networks, enhancing the generalization capabilities of VNF placement across different network topologies. Laaziz et al. [15] designed a multi-objective Integer Linear Model to address VNF placement problems with different topology outcomes (linear or nonlinear). Rankothge et al. [16] proposed two algorithms for VNF placement in response to new service requests and adjustments to VNF placement and location in response to changes in network traffic.

Additionally, when VNFs are placed on nodes, efficiently processing them in the shortest possible time to ensure the completion of as many SFC as possible within specified deadlines or to minimize the completion time for all services constitutes the VNF Scheduling Problem. For instance, Riera et al. [17] initially formulated the VNF Scheduling Problem as a job-shop scheduling problem and proposed its mathematical model without presenting a polynomial-time solution. Li and Qian [18] introduced a grouping scheduling algorithm considering the characteristics of grouped queues and SFC chains, given its lower complexity. Chen and Wu [19] designed a processing and delay model capturing communication delay behaviors in intermediate box processing flows, followed by the development of two corresponding heuristic scheduling algorithms. Mijumbi et al. [20] devised three greedy algorithms and a tabu search algorithm to address the placement and scheduling issues of VNF on supporting virtual machines. Assi et al. [21] proposed an effective and energy-efficient method for VNF placement and scheduling, utilizing heuristic algorithms to tackle the problem.

In the existing literature, reinforcement learning has been employed to tackle combinatorial optimization problems. For instance, in the work by [31], a reinforcement learning algorithm is proposed to determine a variable action set at each decision state, capturing the varying execution times of actions to achieve delay-aware VNF scheduling. Bello et al. [32] introduced a framework using neural networks and reinforcement learning to address combinatorial optimization problems. In the context of workshop scheduling, related research exists, such as [33] where T.

Gabel et al. interpreted the job-shop scheduling problem as a sequential decision problem handled by independent learning agents. Utilizing a probability scheduling strategy, the intelligent agent adjusts parameters using policy gradient reinforcement learning during continuous learning to enhance the performance of the joint policy measured by the standard scheduling objective function.In [34], Luo S. addressed the dynamic flexible job-shop scheduling problem (DFJSP) by setting up a deep Q network. Liu et al. [35] proposed a hierarchical and distributed architecture to solve the dynamic flexible job-shop scheduling problem, introducing specialized state and action representations to handle variable specifications in dynamic scheduling. Additionally, an alternative reward shaping technique was developed to improve learning efficiency and scheduling effectiveness.While the aforementioned workshop scheduling problems bear some similarities to VNF scheduling, they do not consider transmission delays between machines, which is a notable distinction. Most existing work often separates VNF placement and scheduling or neglects transmission delays, rendering it impractical to meet strict service requirements. Therefore, this paper integrates VNF placement and scheduling, considering route selection. A deep reinforcement learning agent is deployed to gather network state information at each moment and make optimal VNF scheduling decisions based on the defined reward function.

3. Problem Description and Model

3.1. Problem Description

The resource allocation of VNF mainly consists of: (a) VNF composition, (b) VNF placement, and (c) VNF scheduling. Regarding VNF composition, a lot of existing literature have studied it and proposed feasible solution [27,28]. this article will not describe it further. In this section, we mainly focus on the joint problem of VNF placement and scheduling for latency-sensitive SFC.

SFC is a chain of network functions composed of different VNF based on customer demands at the beginning of the network service phase. These SFC have a sequential and dependent execution order (the next VNF can only start processing after the previous one is completed). For example, there are one SFC with the execution order: VNF11 → VNF12 → VNF13 → VNF14, in which VNF12 only start to run after the execution of VNF11 is completed.

The placement of VNF is performed based on the completion of the SFC components. Its main purpose is to find a node position that meets the constraints of the VNF and prepares for the subsequent scheduling phase. The VNF scheduling is performed on the traffic of each SFC to enable more SFC to be completed within the specified deadline. Therefore, create a virtual network to place the VNF in the SFC and run them on the virtual machines deployed on physical servers. This article considers issues such as deploying the SFC along the chain on the network, guiding the traffic between them while ensuring their order and required bandwidth, and ultimately scheduling the VNF for their traffic according to the deadline. It is assumed that each VNF instance on each virtual machine can be shared by multiple SFC, but each virtual machine can only handle the traffic of one SFC at a time as described [20,22]. In the remainder of this paper, the problem is refined through examples, and the impact on scheduling is discussed. Assuming an NFV infrastructure consists of four virtual nodes$N$that support VNFs and five links$L$,The available bandwidth of the links is 15$Mbps$, as shown in Fig 1.

Given a set of delay-sensitive SFC is composed of multiple VNF,the virtual network functions are represented by$F = \{f1, f2, \ldots, fm\}$,and VNF_irepresents the function corresponding to the VNF, where$1 \leq i \leq m$, For example,VNF_1means that the function of the VNF is$f1$, and EachVNF_imust be mapped to a node$N$that has the corresponding function. Since each VNF may have different processing capabilities in the network, its processing time is represented as $pt = \frac{w}{pv}$, where$w$is the size of the traffic and $pv$is the processing capability of the VNF. Apart from

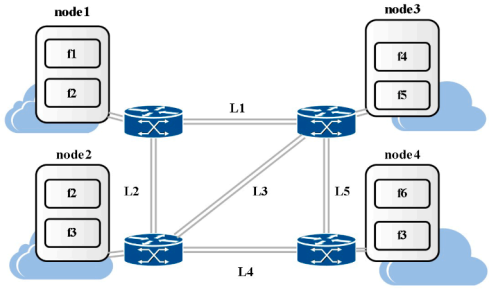


Fig. 1. network topology

the processing time of nodes, the time taken for traffic to be transmitted through a link can also be represented as$Dt = \frac{w}{b}$, $b$represents the required link bandwidth, and according to the above description, the SFC in this example can be represented by a 5-tuple, denoted as $SFC = \{VNF, w, b, pt, D\}$ VNFrepresents the set of VNF required for the SFC,$w$denotes the size of traffic,$pt$is the processing time of the node, and$D$is the deadline for the SFC, Assuming that there are three SFC in the example, $SFC = \{SFC_1, SFC_2, SFC_3\}$$SFC_1 = \{(VNF_1, VNF_4), 24Mb, 12Mbps, 2T, 10T\}$, $SFC_2 = \{(VNF_2, VNF_5), 12Mb, 12Mbps, 1T, 3T\}$, $SFC_3 = \{(VNF_2, VNF_4), 24Mb, 6Mbps, 2T, 8T\}$.The three SFCs arrive at the network at$T = 0$as shown in Fig 2. and in the first scenario where they are accepted in sequence, the situation is shown in Fig 3.

They are all mapped to Node 1 at the same time and processed sequentially in the order of $SFC1$, $SFC2$, and $SFC3$,at time$T = 0$to$T = 2$, Node 1 completed the processing of the first$VNF$of$SFC1$, and traffic began to be transmitted through virtual link$L1$for a duration of 2s. At$T = 4$,Node 3 began processing the nextVNF, and finally completed$SFC1$at$T = 6$, which is less than the deadline$D$of$SFC1$and meets the transmission delay requirements. Next, we look at$SFC2$. When$T = 2$, Node 1 started processing the first$VNF$of$SFC2$after completing$SFC1$, and completed it at$T = 3$with a processing time of 1s. It also began traffic transmission through virtual link $L1$, but since virtual link$L1$was still transmitting traffic for$SFC1$at this time, the remaining bandwidth$(15Mbps - 12Mbp < 12Mbps)$of the virtual link was not sufficient to meet the bandwidth demand, so$SFC2$had to wait, at$T = 4$after the transmission of$SFC1$is completed,$SFC2$starts to transmit and reaches node 3 at$T = 5$. However, since the processing of $SFC1$is not yet completed at this time, it has to wait again and complete at$T = 7$. But by this time, it has exceeded the deadline of$SFC2$and is therefore not accepted. Finally,$SFC3$starts processing at$T = 3$and completes at$T = 5$. At this time, the remaining bandwidth of virtual link$L1$is$15Mbps$, which satisfies the bandwidth required for$SFC3$to transmit. It arrives at node 3 at$T = 9$and finally completes processing at $T = 11$,which is the same as$SFC2$. However, the final processing completion time exceeds the deadline of$SFC3$and cannot be accepted. It is obvious that in this situation, two of the threeSFCthat entered the network at the same time cannot satisfy their latency requirements and are rejected. [8] also proposes the partitioning of the VNF scheduling problem into distinct sub-problems; however, their approach employs heuristic algorithms,

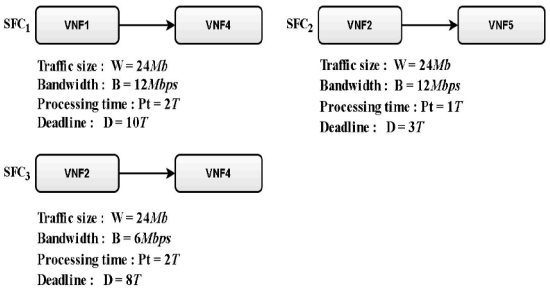


Fig. 2. information of SFC

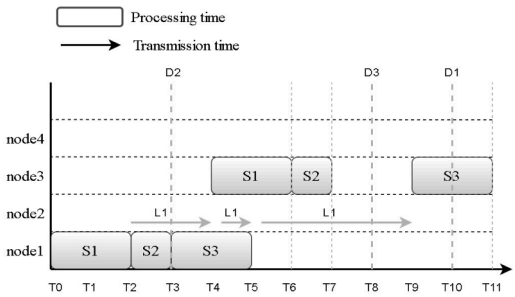


Fig. 3. Normal sequential processing

which may lead to slower convergence rates and susceptibility to issues like local optima when dealing with complex problems.

3.2. Mapping of Virtual Network Functions

In the previous section, the completion times of*SFC*2and *SFC*3were much greater than their deadlines, mainly due to the processing order of*SFC*and insufficient virtual link bandwidth, which resulted in excessive waiting and delayed*SFC*processing time. To reduce the waiting time on nodes and links, the first VNF of*SFC*3is mapped to node 2 here,*SFC*1 and*SFC*3are processed simultaneously at T=0 as shown in Fig 4. At T=2,*SFC*1and *SFC*3have both completed processing.*SFC*1continues to propagate on the virtual link*L*1, while*SFC*3propagates on the virtual link*L*3. When$T = 7$,*SFC*1reaches node 2 and begins processing, completing at T=6. Similar to the previous section,*SFC*2also completes at$T = 7$, while*SFC*3starts processing at$T = 7$ and completes at$T = 9$. Although*SFC*3has still not met the expected processing time, compared to the previous scenario, its completion time is much earlier.

3.3. Processing Order of VNF

In sections A and B,*SFC*2still failed to complete within the deadline because*SFC*1was always processed first, which caused *SFC*2, which is very sensitive to time delays, to wait for two time units. This is not ideal. Therefore, in this section, we consider scheduling to let*SFC*2process traffic first, as shown in Fig 5. At $T = 0$,*SFC*2and*SFC*3start processing at nodes 1 and nodes 2, respectively. At $T = 1$, *SFC*2completes processing, is transmitted on virtual link $L1$, and reaches node 3 at $T = 2$, completing traffic processing at $T = 3$. At this time, the delay requirement is satisfied, and it can be accepted in the network. *SFC*1and *SFC*3also complete at$T = 7$ and $T = 9$, respectively.

3.4. Routing of Traffic

Although*SFC*1and*SFC*2met the delay requirement in the previous sections,*SFC*3still exceeded the deadline by one time unit. This is mainly because*SFC*1arrived at node 3 first and node 3 was idle at that time, so it processed*SFC*1before*SFC*3arrived, causing *SFC*3to wait. In order to solve this problem, we choose a different routing strategy, as shown in Fig 6.

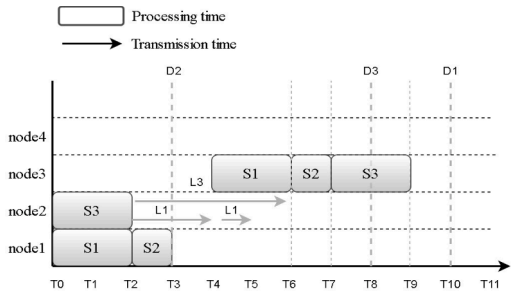


Fig. 4. Mapping of VNF

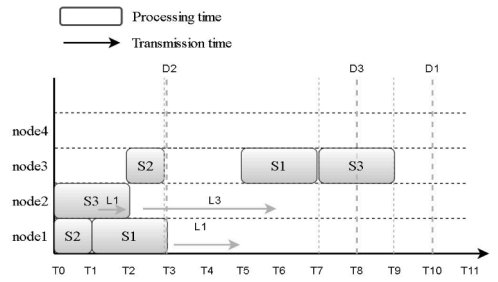


Fig. 5. Processing order of VNF

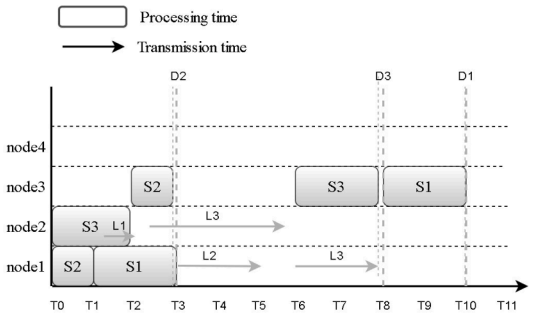


Fig. 6. Routing of Traffic

As before, at $T = 0$,*SFC*2and*SFC*3start processing at nodes 1 and nodes 2, respectively. After*SFC*2processing completes,*SFC*1is processed, and the first*VNF*is completed at$T = 3$. Then,*SFC*1traffic begins to be transmitted on virtual links*L*2 and*L*3instead of*L*1, which changes the transmission route. This allows*SFC*3to arrive at node 3 first and start processing at$T = 6$, completing processing at $T = 8$,thereby meeting the delay requirement. After *SFC*3processing completes,*SFC*1starts processing and completes at$T = 10$, meeting the delay requirement as well. In this way, all three SFCs are completed within the specified time and can be accepted by the network..

From the above scenarios, it can be seen that VNF mapping, processing order, and routing selection all have certain impacts on their respective schedules and thus affect the network acceptance rate. In the remainder of this paper, we will explore how to solve these problems and combine them for VNF scheduling.

4. Scheduling Model Based on DRL

In this section, we first introduce the relevant knowledge about D3QN, and then define the mathematical model of the scheduling problem for delay-sensitive service function chains based on the D3QN model.

4.1. Related Background

a. DQN

The concept of DQN was first proposed by Mnih [23]. It can be viewed as a neural network function approximator with weights. DQN can handle complex decision-making processes with large and continuous state spaces by directly taking raw data (state features) as input and the function values of each state-action pair as output. The training and improvement of DQN are mainly reflected in the following two aspects. First, in the DQN model, the optimal action is selected by interacting with the environment through policyπ, and a new environment is formed and rewards are obtained by the action acting on the environment, forming a new tuple$< S,A,R,St+1 >$, which will be stored in the experience pool for learning by DQN. When the capacity of the

experience pool is full, old experiences will be replaced by new ones, and each transition can be used multiple times to update the parameters, thereby achieving better data efficiency. The second is the target network. In the target network, the network parameters are updated every time during training to make it more stable during training. The DQN network calculates the parameters of the online network are updated according to the target values calculated formula [37],As follow, where$\gamma$is the discount factor [0,1],$a^{'}$is the learning rate, and$\theta^{'}$is the network parameter.

$$yi = ri + \gamma a^{'} maxQ(s^{'}, a^{'}; \theta^{'}) \tag{1}$$

b. Double DQN

However, traditional DQN also has some problems, such as overestimation [24]. The reason for the overestimation problem is that in the learning process of the neural network, bias and variance problems may occur. Bias refers to the insufficient fitting ability of the model itself, which cannot accurately fit the true Q value function. Variance refers to overfitting of the model to the training data during the training process, resulting in insufficient generalization ability for unknown data. This makes the estimated value function larger than the true value function, so that the worst actual value may become the best estimated value, while the best actual value may become the worst estimated value. To avoid this problem, the Double DQN form is adopted [38]. In DQN, a new network is added, whose structure is the same as the original network, but uses different network parameters. These two networks have different uses. The original network is used to control the agent's collection of learned experiences and selection of actions, while the new network is used to calculate the value of actions. This decoupling of selection and evaluation reduces overestimation, making learning more stable. The formula in Eq.(2):

$$yi = ri + \gamma Q(s^{'}, arga^{'} maxQ(s^{'}, a^{'}; \theta); \theta^{'}) \tag{2}$$

c. Dueling DQN

Next, in DQN, the neural network outputs the value of actions, but evaluating the value of actions alone may not be accurate. Because the value of actions$Q(S, A)$is related to the State and the Action, but the degree of this relationship or influence is not the same. We hope to reflect the difference between these two factors. Therefore,the Dueling DQN algorithm improves DQN from the network structure [36].The neural network output of the action value function can be divided into state value function and advantage function. The formula in Eq.(3):

$$Q_{\pi}(s, a) = V_{\pi}(\mathrm{s}) + A_{\pi}(s, a) \tag{3}$$

$V_{\pi}(\mathrm{s})$represents the state value function, which is mainly equal to the average of all action probabilities in that state, i.e., the sum of all action values multiplied by their probabilities. $Q_{\pi}(s, a)$value represents the action value in that state. The advantage function $A_{\pi}(s, a) = Q_{\pi}(s, a) - V_{\pi}(\mathrm{s})$ corresponds to the average high or low of each action, so that action values that are higher than the average will be even higher, while those that are lower will be even lower, which can speed up the convergence of the network. Considering the above description, it is proposed to use D3QN as the network model for training in this paper, which is expected to achieve better results.

4.2. Problem Definition and Formulas

In the physical network graph G (N, L), where N represents virtual nodes used to host and run different types of VNF, and L represents virtual links connecting every two nodes. There is a set of SFC, and each SFC requests its traffic to be processed by the network, which needs to satisfy the following requirements: a) each VNF of each SFC must be mapped to a node capable of processing its function; b) each VNF of each SFC is processed in order, and the next VNF cannot be processed until the previous one is completed; c) the bandwidth of the link must meet the traffic demand. The purpose of this paper is to find the optimal node to map and schedule VNF while meeting the above requirements, in order to maximize the reception rate of SFC in the network.

The main parameters involved and their meanings are shown in Table 1.

This paper defines the following decision variables for the joint VNF placement and scheduling problem

$K_{ks}^{n\delta} \in \{0, 1\}$is a binary variable,indicating Whether the VNF$k \in k$of the SFC$s \in S$is mapped to the n node$n \in N$at time stampδ,$\delta \in \delta$ to start processing.(1)or not(0)

$l_{ij}^{se} \in \{0, 1\}$is a binary variable,indicating Whether the SFC$s \in S$virtual link $e \in E$is routed on the link $(i,j) \in L$, if yes, 1, otherwise 0.

$j_s \in \{0, 1\}$is a binary variable,indicating Whether the SFC$s \in S$is accepted by the network, that is, whether it can be completed within the specified time.(1)or not (0)

Next, let's consider the constraints required for optimizing the objective and its constraints.

We aim to schedule SFC while ensuring constraints, therefore the objective is to maximize SFC acceptance rate, which is equivalent to minimizing SFC rejection rate:

$$\begin{aligned} &Maximize \Sigma_{s \in S} j_s \\ &st \begin{cases} C1: K_{ks}^{n\delta} f_s^k = f_n \\ C2: \sum_{n \in N} K_{k+1s}^{n\delta} \le 1 - \sum_{n^{'} \in N} K_{ks}^{n^{'}\delta} \\ C3: K_{ks}^{n\delta} = 1 - K_{k^{'}s^{'}}^{n\delta}, k, k^{'} \in K, s, s^{'} \in S \\ C4: \sum_{k \in K} p_{ns}^k + \sum_{e \in E} p_s^e \le D_s, s \in S \\ C5: l_{ij}^{se} b_{k,k+1} \le C_{ij}, e \in E, (i,j) \in L \\ C6: \sum_{k \in K} K_{ks}^{n\delta} = 1 \end{cases} \end{aligned} \tag{4}$$

The optimization objective of this article is mainly subject to the constraints C1∼C6. C1 ensures that the VNF types on the SFC are the

Table 1
SUMMARY OF KEY NOTATIONS

parameters	
$G^{'}(K, E)$	Represents an SFC forwarding graph,
F	Type collection of VNF
f_t	VNF instance type $f_t \in F$
C_{ij}	Available capacity between link i j
S	Set of SFC
N	Set of node
δ	timestamp
$b_{k,k+1}$	Required capacity of virtual links between The VNF$k \in k$ and The VNF$k+1 \in k$ in each SFC
w_s	Flow size of SFC$s \in S$
D_s	Deadline of SFC$s \in S$
f_s^k	The type of the VNF$k \in k$of the SFC$s \in S$,$f_s^k \in f_t$
f_n	The available instantiate types on node n$n \in N$,$f_n \in f_t$
V_S	The number of VNF in the SFC$s \in S$
p_{ns}^k	The average processing time of the VNF$k \in k$ of the SFC $s \in S$on the node$n \in N$
p_s^e	The transit time of the traffic on the virtual link
pd_s	The number of VNF processed for the SFC
end_S	The time when the last VNF of the SFC $s \in S$was completed.
u_n^t	Represents the end time when the last VNF has been processed on node n$n \in N$.
$u_{n,k}^t$	Represents the time taken for the VNF$k \in k$that has been processed on node n$n \in N$.

same as the types of VNF instances mapped to the node. C2 states that the VNF$k+1 \in k$cannot be processed while the VNF$k \in k$of the SFC is still being processed. C3 ensures that a node's VNF instance cannot process VNF of another SFC while processing the current SFC VNF. C4 ensures that the remaining shortest completion time of the SFC can meet the deadline.C5 requires that the virtual link capacity required by the SFC must be less than the capacity of the physical link. C6 specifies that the VNF of the SFC can only be mapped to one node at the same time.

5. Scheduling Algorithm Based On D3QN

In this section, we first present the state definition of D3QN, followed by the candidate scheduling rules (actions) and reward definitions for each scheduling point. Then, we explain the node and route selection process, and finally, we discuss the network structure and training method of D3QN.

5.1. State Definition

Generally, the number of SFC or network nodes is usually very large. If we use the state features of each SFC or network at every moment as an indicator, it may lead to a large input volume, which can cause difficulties in adapting and training D3QN. Therefore, we propose to extracted the features of each SFC and their average value is calculated to facilitate the training of D3QN and make it easier to extend to other environments. Therefore, the state is defined as follows:

(1) The*ACR*denotes the probability of successfully mapped VNF within each SFC relative to the total number of VNF.:

$$ACR = \frac{\sum_1^s \frac{pd_S}{V_S}}{s} \tag{5}$$

(2) The*AUR*represents the total time spent processing VNF on the average node compared to the total time the node runs:

$$AUR = \frac{\sum_1^n \frac{\sum_1^k \left(u'_{n,k}\right)}{u'_n}}{n} \tag{6}$$

(3) The*EOR*represents the potential SFC rejection rate at the current scheduling time point, where if the estimated shortest remaining processing time of the unfinished part (i.e., the remaining unprocessed VNF) of the SFC exceeds the specified deadline, even if the SFC has not exceeded the deadline at this time, it is also counted as a potential rejection. The main process is as follows:

In Algorithm 1, the third line represents finding the SFC that has not yet completed the mapping, while the fifth to tenth lines represent the estimated processing time for the remaining VNF in the unfinished SFC, and determine whether the SFC can complete the processing within the expected processing time

(1) The*AOR*refers to the ratio of SFC that have exceeded their processing deadline to the total number of SFC. The algorithm overview is as follows:

In Algorithm 2, lines 3 to 7 represent all SFC in the loop to determine whether the current processing time has exceeded the deadline. If it has exceeded, the N value is added by 1 and divided by the total number of SFC

In order to mitigate the wide range of input variations, reducing the performance and generality of the agent, this paper sets the values of the aforementioned states within the range [0, 1] and takes their averages. This approach aims to enhance the agent's adaptability to different networks. Additionally, the average completion rate*ACR*of the SFC and the average utilization rate *AUR*of nodes are set to better capture the moment-to-moment changes in network states. This is done to enrich the rewards and avoid sparsity in the reward structure.*ACR*and*AUR*also effectively reflect the current SFC completion status, enabling the agent to make informed decisions. The actual completion rate *AOR*reflects the quantity of existing incomplete SFC in the current network, while the expected completion rate*EOR* represents the anticipated number of SFC that may remain incomplete. These metrics provide direct insights into the current network state, allowing the agent to make better decisions. However, due to their limited variability, they may result in sparse rewards. Hence, they need to be complemented by*ACR*and *AUR*to provide a comprehensive understanding of the current network state for the agent.

5.2. Action Definition

In most existing research, the applicability of single-rule scheduling algorithms is limited, as they may not effectively cater to all states. Evolutionary algorithms like genetic algorithms often incur lengthy processing times. Therefore, this paper introduces Composite Rule Scheduling, presenting five rules tailored for intelligent agents. Different rules are designed to address various network state scenarios. During each scheduling iteration, the algorithm selects the rule with the highest current reward value based on the network state. Subsequently, the chosen rule is employed to identify the most suitable SFC for processing in the current iteration, laying the groundwork for subsequent route optimization.

Algorithm 1
The Calculation Process Of *EOR*

Input: V_S,pd_S,D_s, end_S
Output: *EOR*
1: N= 0
2: **for** k = 1: n **do**
3: **if** pd_S<V_S**then**
4: T= 0
5: **for** i =pd_S+1:V_S**do**
6: T += (p^k_{ns}+p^e_s)
7: **end for**
8: **if** end_S+T >D_s**then**
9: N+=V_S- i
10: Break
11: **end if**
12: **end if**
13:**end for**
14:$EOR = \frac{N}{\sum_1^S V_S}$
15: **return** *EOR*

Algorithm 2
The Calculation Process of *AOR*

Input: V_S,pd_S,D_s, end_S
Output: *AOR*
1:N= 0
2: **for** k = 1: n **do**
3: **if**$end_S > D_s$**then**
4: N+=1
5: **end if**
6: **end for**
7:$AOR = \frac{N}{\sum_1^S V_S}$
8: **return** *AOR*

(1) Rule **1**:

For Rule 1, first, the deadline of each SFC is compared in order, and the SFC with the smallest deadline is selected as the highest priority. If there are deadlines that are the same, the SFC with the largestend_Svalue is selected as the highest priority, because if the current$end_S$of an SFC is greater, it means there is less time available to process subsequent VNF.

(2) Rule **2**:

For Rule 2, this paper will use scheduling based on the lowest slackness as high priority, because slackness reflects the urgency of a task, and the lower the slackness, the less available time for the task and the higher the urgency.

(3) Rule **3**:

For Rule 3, we divide it into two cases depending on whether there exists an SFC whoseend_Svalue exceeds the deadline,that is, O is not empty. If such an SFC exists, the SFC with the smallest slack time is given the highest priority. If not, each SFC remaining time is divided by the remaining number of operations, and the smallest value is assigned the highest priority. The specific Algorithm 3 is shown below:

(4) Rule **4**:

Similar to Rule 3, two scenarios are also handled in this case. If O is empty, the remaining time is divided by the estimated average processing time of SFC. Otherwise, the minimum slack time is used as the selection criterion. The specific Algorithm 4 is as follows:

(5) Rule **5**:

The expected shortest processing time refers to calculating how much time is needed to process each unfinished SFC.

5.3. Reward

Since the objective of this article is to minimize the rejection rate, a reward functionRtbased on this objective value is designed. Each SFC must be completed as much as possible within its deadline to reduce the rejection rate and obtain more rewards. However, since the change in the rejection rate can only be known when the entire SFC exceeds its deadline or has been processed, the reward may become sparse. Therefore, this article defines returns by considering the values of the four key state characteristics of the current state *AOR, EOR, ACR, AUR*, and the next state *AOR', EOR', ACR', AUR'*.*Rt*as follows:

$$rt = \begin{cases} 1 & \text{if } AOR^{'} < AOR \\ -1 & \text{if } AOR^{'} < AOR \\ f\ (EOR, EOR^{'}, ACR, ACR^{'}) & \text{if } AOR^{'} = AOR \end{cases} \tag{7}$$

When$AOR^{'} = AOR$we need to define an auxiliary function$f(EOR, EOR^{'}, ACR, ACR^{'})$to further determine the value of rt.

$$rt = \begin{cases} 1 & \text{if } EOR^{'} < EOR \\ -1 & \text{if } EOR^{'} < EOR \\ \mathrm{g}(ACR, ACR^{'}) & \text{if } EOR^{'} = EOR \end{cases} \tag{8}$$

When $EOR^{'} = EOR$,rt is as follows:

$$rt = \begin{cases} 0 & \text{if ACR}^{'} > \text{ACR} \\ 1 & \text{if ACR}^{'} > 1.1 \times \text{ACR} \\ -1 & \text{otherwise} \end{cases} \tag{9}$$

For*AUR*, the reward $rt^{'}$ in this article is set as follows:

$$rt^{'} = \mathrm{e}^{\mathrm{AUR}^{'} - \mathrm{AUR}} \tag{10}$$

The final reward R is set as follows

$$R = rt + rt^{'} \tag{11}$$

5.4. Node Selection and Routing Optimization

According to the above description, the D3QN actions can be used to select the highest priority SFC for mapping. However, selecting the optimal node and routing for the SFC to meet the latency requirements is also a critical issue. Therefore, the rest of this section will describe how node and routing selection is performed in this paper.

a. Node Selection

The purpose of node selection is to choose a node that can handle the VNF type instance and can start processing its traffic as early as possible. This is because only by processing the traffic as early as possible can the

Algorithm 3
The Process Of Rule 3

Input:V_S,pd_S,D_s, end_S
1: O←{S| $pd_S < V_S$ && $D_s < end_S$}
2: P← {S| $pd_S < V_S$}
3: **if** Isempty(O) **then**
4: $SFC = \mathrm{argmin}_{S \in P} \frac{D_s - end_S}{VS - pd_S}$
5: **else**
6: $SFC = \mathrm{argmin}_{S \in O}[D_s - end_S - \sum_{v=pd_S}^{v_s}(p_{ns}^k + p_s^\epsilon)]$
7: **end if**

Algorithm 4
The Process Of Rule 4

Input: V_S,pd_S,D_s, end_S
1: O← {S| $pd_S < V_S$ && $D_s < end_S$}
2: P← {S| $pd_S < V_S$}
3: **if** Isempty(O) **then**
4: $SFC = \text{argmin}_{s\in p}\frac{D_s - end_s}{\sum_{v=pdS}^{vs}(p_{ns}^k + p_s^e)}$
5: **else**
6: $SFC = \text{argmin}_{S\in O}[D_s - end_S - \sum_{v=pdS}^{vs}(p_{ns}^k + p_s^e)]$
7: **end if**

completion time be shorter and the latency requirement be met. This is described by the following formula:

$$\min_N \max(NT, A_i) \tag{12}$$

NTrepresents the completion time of the last VNF processed on node N,$A_i$represents the time when the traffic of the selected SFC arrives at node N.$\max(NT, A_i)$is used to represent the earliest start time for the traffic to arrive at that node. Therefore, $\min_N \max(NT, A_i)$represents selecting the earliest available among all available nodes.

b. Routing Optimization

The above formula represents the node selection, but currently, it is not possible to determineA_i, which is the time when the traffic can reach the next candidate node after flowing through the link from the previous node. Therefore, the Dijkstra algorithm is used to find the shortest time path between two nodes while ensuring that the physical link capacity meets the virtual link when routing traffic, in order to determineA_i. Because it is to find the shortest time path between two points, the path weight is set to the link transmission time plus the waiting time, as shown in Fig 7.

The traffic shown on Fig 7(a) is from the source node A to the destination node C, and the transmission time of the traffic on the link is one time node. Link L1 needs to wait for 3t nodes due to insufficient resources from T=0 to T=3, so the weight of L1 is 4, while the weight of L2 is 1 as it has sufficient resources. The values of nodes A, B, and C are 0, 1, and 4, respectively. When taking B as the starting point, as shown in Fig 7(b), because the processing time of L3 is from T=0 to T=1 and the value of node B is 1, L3 does not need to wait, so the weight of L3 is 1. Therefore, the values of nodes A, B, and C are updated to 0, 1, and 2. Hence, the shortest time path from A to C is 2. The specific node and routing selection algorithm are shown as follows:

In Algorithm 5, from lines 2 to 5, all nodes are looped to find all nodes that meet the constraint condition (2). The Dijkstra algorithm is used to calculate the distance between the node mapped by the previous VNF on the SFC and the available nodes that need to be mapped to the current VNF. From lines 6 to 10, the time from the previous node to all available nodes is calculated as A_iFinally,$A_i$is compared with$NT$to find a larger value and assigned to M, where M represents the earliest start time of the node, and line 12 represents the next processing node of the current VNF that finds the earliest start node.

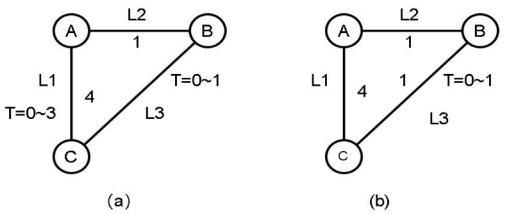


Fig. 7. weight setting

5.5. D3QN Network Architecture and Training

The network structure of D3QN used in this article consists of an input layer and a hidden layer fully connected, which are then split into two branches: the advantage branch and the value branch. The Q value of each sub-action is obtained by aggregating the value branch and the corresponding advantage branch. respectively. The discount factor is 0.9, the learning rate is 0.0002,The Batch size is 64 and the exploration rate is set to 0.5. In the training process, the scheduling point is defined as the beginning of each operation, and the training method and overall framework based on D3QN are shown in the following Algorithm 6.

6. Simulation and Performance Analysis

This section conducts simulations and performance analysis on the proposed rule-based selection D3QN scheduling algorithm. The paper compares the performance of D3QN with each composite rule in different scenarios and demonstrates its superiority over traditional DQN algorithm and genetic algorithm.

6.1. Parameter Settings

To conduct the evaluation, a network model similar to the one in [25] was designed. One is a medium-sized network consisting of 15 fully connected nodes, and 15, 20, 25, 30 and 35 randomly generated SFC were introduced into the network. The other is a large network consisting of 30 fully connected nodes, and 30, 35, 40, 45 and 50 randomly generated SFC were introduced into the network. The available bandwidth of each link is fixed at 50Mbps. In this article, it is assumed that each VNF can be processed on at least one VM node, and each VM node has the ability to host 2-3 VNF. For each SFC, random traffic generation ([25-75]Mbits), bandwidth requirements ([15-25]Mbps), and variable VNF ([2-4]) were used as service compositions. Their deadlines were set to 4/3 times the sum of their processing and transmission delays, without considering any waiting delays [26].

6.2. The training process of D3QN

Test D3QN on a predefined instance containing 15 nodes and 35 SFC insertions. The total reward results obtained by D3QN in the first 300 training steps are shown in Fig 8 and Fig 9 .

As shown in Figs. 8 and 9, Fig. 8 shows the variation of reward value with training frequency. The vertical axis represents the reward value, and the horizontal axis represents the training frequency. The newly opened training reward value oscillates within 0-100 steps. The neural network randomly selects actions, and as the training progresses, the reward value increases and eventually converges to the maximum value. The reason for the oscillation is that there is still a small probability of random action selection in the later stage of training. Fig. 9 shows the changes in the loss value during the training process. It can be seen that the overall loss value is decreasing, with a small fluctuation in the middle. The final loss value converges to close to 0, indicating that the predicted results of the model are very close to the true results in the training data.

Algorithm 5
Node Selection And Routing Optimization

```
M = []
for m= 1: N do
  if This node satisfies constraint (2) then
    L = [],A_i= end_s
    L← Dijkstra, Find the shortest time path from the previous node of the SFC to the candidate node and keep the pass nodes
  for s = L do
    A_i+=p_s^e
    end for
  end if
M←max(NT, A_i)
end for
selected node = ArgminM
```

Algorithm 6
The Process Of D3QN

```
Initialize experience pool D
Initialize online network weight parameters θ
Initialize target network weight parameters θ' = θ
for episode do
  Generate state set{ACR, AUR, EOR, AOR}= {0, 0, 0, 0, 0}train parameters by inputting into the network.
  for I = 1:∑_{i=1}^{n} V_S do
    Select the action with the highest Q-value through the network model
    Execute action A, select the SFC with the highest priority, return it to the environment for scheduling, and generate St+1
    Generate reward Rt using Eq.(7)-(11)
    Store the quadruple < S, A, R, St + 1 > in the experience pool D
    Randomly sample a batch from the experience pool D
    Update online network parameters with gradient descent
    Update target network parameters at step Y,θ' = θ
  end for
end for
```

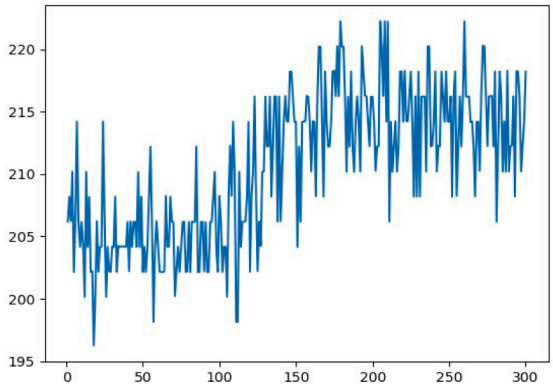


Fig. 8. Reward iteration

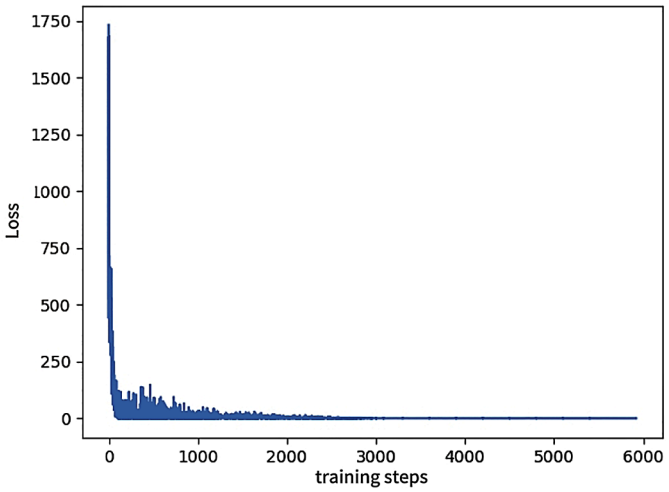


Fig. 9. Changes in loss values during training

6.3. Compared with Composite Rules

To verify the effectiveness of D3QN, it was compared with each composite rule used. In order to ensure the generality of D3QN, the comparison was made both on the 15-node and 30-node networks. In addition, to eliminate randomness, this paper trained 50 times for each comparison and took the average value. This more effective comparison can avoid randomness, and the average value is more convincing, as shown in Fig 10 and Fig 11:

Based on the test results shown in the above figure, it can be seen that D3QN maintains the lowest rejection rate compared to each individual composite scheduling rule. This highlights the ability of a trained intelligent agent to choose the current optimal rule for scheduling in different network environments, resulting in lower rejection rates. Of course, there is a certain difference between each rule, and it can be seen that the rational design of each rule and the appropriate selection process in D3QN operation are the main reasons for performance. Overall, D3QN can select a composite rule that is more advantageous to the current situation at each scheduling time point, making it more effective than a single rule.

6.4. Compared to Other Algorithms

Apart from comparing with a single compound rule, this paper also compares with a random selection algorithm (i.e., randomly selecting a compound scheduling rule with equal probability at each scheduling point) and a genetic algorithm. In addition, in order to compare the superiority of D3QN in handling discrete spaces, D3QN was compared with traditional DQN, which used the same set of available actions in each state and compared the results with a node count of 15 and 30, as

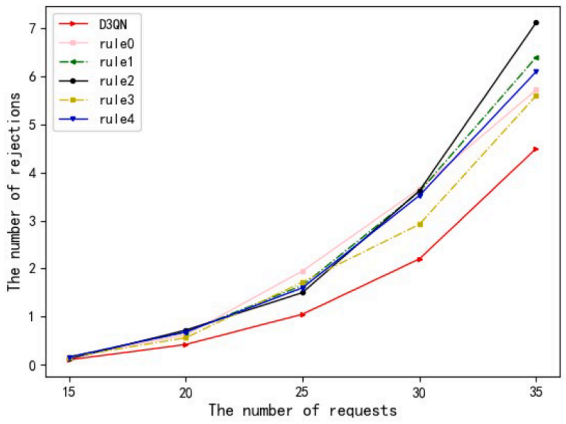


Fig. 10. the SFC rejection rate with 15 nodes

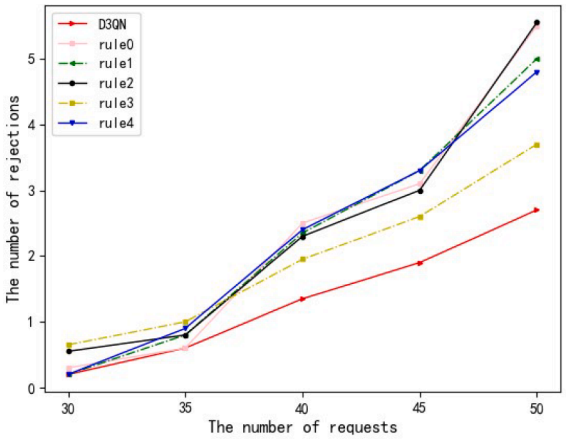


Fig. 11. the SFC rejection rate with 30 nodes

shown in Fig 12 and Fig 13.

From the above figures, it can be seen that compared with the random action selection strategy and genetic algorithm, D3QN can almost always obtain lower total rejection numbers in all instances, which means that D3QN can select appropriate actions for compound rule selection at each rescheduling stage to achieve better scheduling results. Furthermore, in the instances, the agent performance of D3QN is superior to that of DQN, which may be due to DQN's inability to accurately distinguish different states in the network environment, thus inevitably deteriorating overall performance. Additionally, the dual-

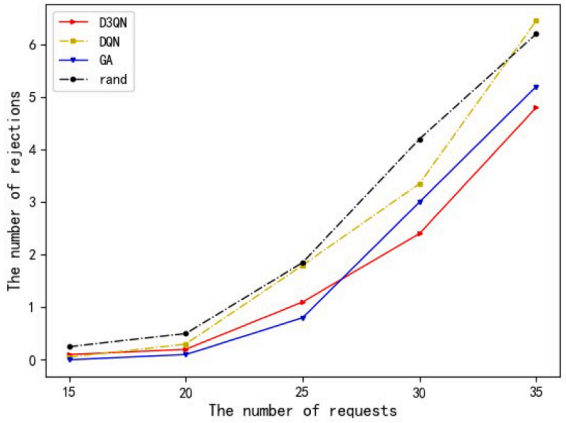


Fig. 12. the SFC rejection rate with 15 nodes

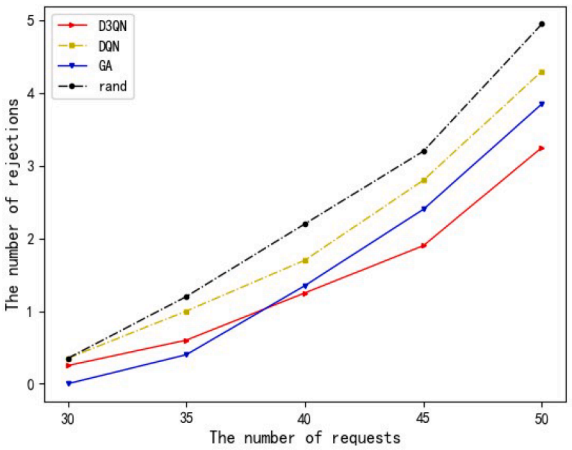


Fig. 13. the SFC rejection rate with 30 nodes

branch structure of D3QN can better reflect the advantages of each different action and select the optimal action. In summary, the D3QN agent is more reasonable and effective than the DQN agent in handling discrete state spaces.

6.5. Comparison of node selection performance

In order to verify the effectiveness of node selection in this article, we will compare the average single VNF routing cost and lateness rate with randomly selected available nodes. The average single VNF routing cost is the resource required for a route from one VNF to the next VNF, and its formula is as follows:

$$ASV = \frac{\sum_{1}^{S}\left(\frac{\sum_{1}^{k} B^{s}_{k,k+1} \times L_{k,k+1}}{k} \times j_s\right)}{S} \tag{13}$$

Where $B^{s}_{k,k+1}$ represents the bandwidth requirement from the VNF $k \in k$ to the VNF $k+1 \in k$ of the SFC $s \in S$, and $L_{k,k+1}$ represents the routing path experienced between these two VNF.Next, this article will compare the performance of node selection with and without node selection when the number of nodes is 30, as shown in Fig 14 and Fig 15.

Referring to Fig 14 and Fig 15, where YD3QN represents node selection function and ND3QN represents no node selection function. It can be seen that both SFC acceptance and single VNF routing cost have good performance, indicating that when there is a good node selection, VNF scheduling can be completed in a shorter time to improve VNF

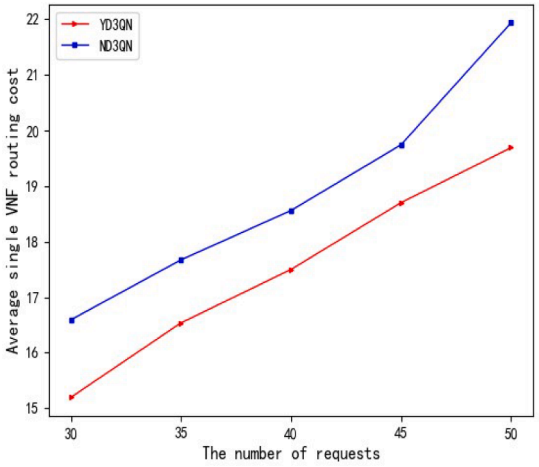


Fig. 14. the SFC rejection rate with 30 nodes

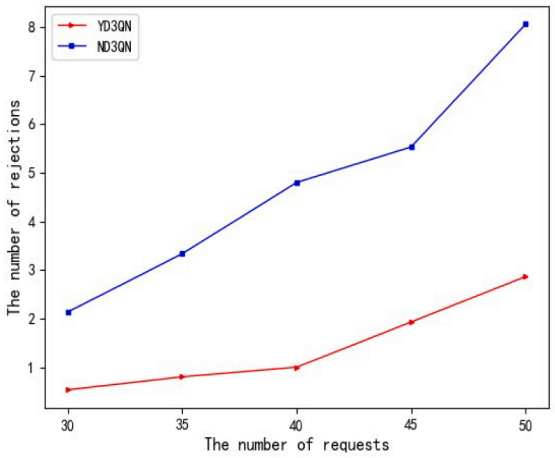


Fig. 15. Average single VNF routing cost

acceptance rate. At the same time, from Fig. 15, it can be seen that a good node selection can effectively reduce the cost of VNF routing and increase the revenue of operators.

7. Conclusion

The method proposed in this paper is based on rule-based D3QN scheduling to solve scheduling problems in networks. Five rules are specified to select an unprocessed VNF through rules and allocate it to a node through node and route selection at each scheduling time point. In addition, D3QN is used for training to select more suitable rules at each scheduling node.

We also compared two network environments to verify the effectiveness and generality of D3QN. The results show that after training, D3QN has better performance than other compound rules, random selection strategies, and genetic algorithms. Furthermore, D3QN has a significant advantage over traditional DQN, which further demonstrates the superiority of D3QN in handling discrete state spaces.Finally, D3QN also has better results compared to genetic algorithms.

In future work, more practical factors in scheduling will be studied, such as whether another virtual machine instantiates the VNF to be processed when there is not enough capacity on the node, or VNF migration and resource preemption between different VNF on virtual machines. In addition, the scheduling rules will be optimized, or other more advanced policy-based RL methods.

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.

References

[1] J Sherry, S Ratnasamy, J S At, Tech. Rep, EECS Department, University of California, Berkeley, 2012.
[2] R Mijumbi, J Serrat, J L Gorricho, et al., Network function virtualization: State-of-the-art and research challenges [J], IEEE Communications surveys & tutorials 18 (1) (2015) 236–262.
[3] R Mijumbi, J Serrat, J L Gorricho, et al., Management and orchestration challenges in network functions virtualization [J], IEEE Communications Magazine 54 (1) (2016) 98–105.
[4] B Han, V Gopalakrishnan, L Ji, et al., Network function virtualization: Challenges and opportunities for innovations [J], IEEE communications magazine 53 (2) (2015) 90–97.
[5] Z Shu, T. Taleb, A novel QoS framework for network slicing in 5G and beyond networks based on SDN and NFV [J], IEEE Network 34 (3) (2020) 256–263.
[6] Z Shu, T Taleb, J. Song, Resource allocation modeling for fine-granular network slicing in beyond 5G systems [J], IEICE Transactions on Communications 105 (4) (2022) 349–363.
[7] X Fu, F R Yu, J Wang, et al., Service function chain embedding for NFV-enabled IoT based on deep reinforcement learning [J], IEEE Communications Magazine 57 (11) (2019) 102–108.
[8] H A Alameddine, L Qu, C. Assi, Scheduling service function chains for ultra-low latency network services [C], //, in: 2017 13th international conference on network and service management (CNSM), IEEE, 2017, pp. 1–9.
[9] J G Herrera, J F Botero, Resource allocation in NFV: A comprehensive survey [J], IEEE Transactions on Network and Service Management 13 (3) (2016) 518–532.
[10] L Fang, X Zhang, K Sood, et al., Reliability-aware virtual network function placement in carrier networks [J], Journal of Network and Computer Applications 154 (2020) 102536.
[11] N Hyodo, T Sato, R Shinkuma, et al., Virtual network function placement for service chaining by relaxing visit order and non-loop constraints [J], IEEE Access 7 (2019) 165399–165410.
[12] Y Alahmad, A Agarwal, T. Daradkeh, Cost and Availability-Aware VNF Selection and Placement for Network Services in NFV [C], //, in: 2020 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, 2020, pp. 1–6.
[13] H Feng, Z Shu, T Taleb, et al., An Aggressive Migration Strategy for Service Function Chaining in the Core Cloud [J], IEEE Transactions on Network and Service Management (2022).
[14] P Sun, J Lan, J Li, et al., Combining deep reinforcement learning with graph neural networks for optimal VNF placement [J], IEEE Communications Letters 25 (1) (2020) 176–180.
[15] L Laaziz, N Kara, R Rabipour, et al., FASTSCALE: A fast and scalable evolutionary algorithm for the joint placement and chaining of virtualized services [J], Journal of Network and Computer Applications 148 (2019) 102429.
[16] W Rankothge, F Le, A Russo, et al., Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms [J], IEEE Transactions on Network and Service Management 14 (2) (2017) 343–356.
[17] J F Riera, E Escalona, J Batalle, et al., Virtual network function scheduling: Concept and challenges [C], //, in: 2014 international conference on smart communications in network technologies (SaCoNeT), IEEE, 2014, pp. 1–5.
[18] X Li, C. Qian, Low-complexity multi-resource packet scheduling for network function virtualization [C], //, in: 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, 2015, pp. 1400–1408.
[19] Y Chen, J. Wu, Flow scheduling of service chain processing in a NFV-based network [J], IEEE Transactions on Network Science and Engineering 8 (1) (2020) 389–399.
[20] R Mijumbi, J Serrat, J L Gorricho, et al., Design and evaluation of algorithms for mapping and scheduling of virtual network functions [C], //, in: Proceedings of the 2015 1st IEEE conference on network softwarization (NetSoft), IEEE, 2015, pp. 1–9.
[21] C Assi, S Ayoubi, N El Khoury, et al., Energy-aware mapping and scheduling of network flows with deadlines on VNFs [J], IEEE Transactions on Green Communications and Networking 3 (1) (2018) 192–204.
[22] J Li, W Shi, P Yang, et al., On dynamic mapping and scheduling of service function chains in SDN/NFV-enabled networks [C], //, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.
[23] V Mnih, K Kavukcuoglu, D Silver, et al., arXiv preprint, 2013.
[24] H. Hasselt, Double Q-learning [J], in: Advances in neural information processing systems, 23, 2010.
[25] L Qu, C Assi, K. Shaban, Delay-aware scheduling and resource optimization with network function virtualization [J], IEEE Transactions on communications 64 (9) (2016) 3746–3758.
[26] N Promwongsa, A Ebrahimzadeh, R H Glitho, et al., Joint VNF placement and scheduling for latency-sensitive services [J], IEEE Transactions on Network Science and Engineering 9 (4) (2022) 2432–2449.
[27] S Clayman, E Maini, A Galis, et al., The dynamic placement of virtual network functions [C], //, in: 2014 IEEE network operations and management symposium (NOMS), IEEE, 2014, pp. 1–9.
[28] M T Beck, J F Botero, Coordinated allocation of service function chains [C], //, in: 2015 IEEE global communications Conference (GLOBECOM), IEEE, 2015, pp. 1–6.
[29] F Pezzella, G Morganti, G. Ciaschetti, A genetic algorithm for the flexible job-shop scheduling problem [J], Computers & operations research 35 (10) (2008) 3202–3212.
[30] C Pham, N H Tran, CS Hong, Virtual network function scheduling: A matching game approach [J], IEEE Communications Letters 22 (1) (2017) 69–72.
[31] J Li, W Shi, N Zhang, et al., Delay-aware VNF scheduling: A reinforcement learning approach with variable action set [J], IEEE Transactions on Cognitive Communications and Networking 7 (1) (2020) 304–318.
[32] Bello I, Pham H, Le Q V, et al. Neural combinatorial optimization with reinforcement learning [J]. ar**v preprint ar**v:1611.09940, 2016.
[33] T Gabel, M. Riedmiller, Distributed policy search reinforcement learning for job-shop scheduling tasks [J], International Journal of production research 50 (1) (2012) 41–61.
[34] S. Luo, Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning [J], Applied Soft Computing 91 (2020) 106208.
[35] R Liu, R Piplani, C. Toro, Deep reinforcement learning for dynamic scheduling of a flexible job shop [J], International Journal of Production Research 60 (13) (2022) 4049–4069.
[36] Z Wang, T Schaul, M Hessel, et al., Dueling network architectures for deep reinforcement learning [C], //, in: International conference on machine learning, PMLR, 2016, pp. 1995–2003.
[37] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[38] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, 2, AAAI, Phoenix, AZ, 2016, p. 5.

Zhiwei Liu is currently pursuing a master's degree at Fujian Agriculture and Forestry University, China. His research interests include deep reinforcement learning, network functions virtualization, and cloud computing. He is now conducting research work on the scheduling of VNFs through deep reinforcement learning.

Zhaogang Shu is currently an Associate Professor at the College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou, China. He also is the director of Department of Computer Science and Cloud Computing Lab, Fujian Agriculture and Forestry University. He received B.S. and M.S. degrees in computer science from Shantou University, China in 2002 and 2005 respectively. He also received Ph.D. degree from South China University of Technology, Guangzhou, China, in 2008. From Sept. 2008 to July 2012, he worked as a senior engineer and project manager at Ruijie Network Corporation, Fuzhou, China. From Oct. 2018 to Oct. 2019, he worked as a visiting professor in MOSIAC lab at the Department of Communications and Networking, Aalto University, Finland. He directed more than 10 research projects and was the author of more than 30 papers and 5 patents. His research interests include software-defined network, network function virtualization, 5G network and next generation network architecture, network security, machine learning based network optimization, cloud computing and edge computing. He serves as the reviewers of many famous journals on network and communications, including IEEE Network, IEEE/ACM Transactions on Networking, IEEE Transactions on Network Service and Management, ACM/Springer Mobile Networks, Elsevier Computer networks and so on. He also is the member of CCF (China Computer Federation) and Fujian Computer Society.

Shuwu Chen is currently a professor at the College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou, China. He also is the director of Innovation Lab of IoT technology, Fujian Agriculture and Forestry University. He received bachelor's degree in industrial automation from Chang'an University, China, in 1998. And, he received master's degree in radio physics from Xiamen University, China, in 2003. He is the Co founder of Xiamen Four-Faith Communication Technology Co., Ltd., which focus on IoT technology and solutions. He directed dozens research projects and was the author of more than 10 patents. His research interests include IoT technology, edge computing and AI algorithm.

Yiwen Zhong received the M.S. and Ph.D. degrees in computer science and technology from Zhejiang University, Hangzhou, China, in 2002 and 2005, respectively. He is currently a Professor with the College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou, China. His current research interests include computational intelligence, data visualization, and bioinformatics.

Jiaxiang Lin received the Ph.D. degree in Communication and Information System from Fuzhou University, China, in 2010. He is currently an Associate Professor with the College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou, China. He has hosted four national, provincial and ministerial level research projects, authored over 40 referred scientific papers and hold three patents of invention. His research interests include spatial data mining, artificial intelligence, and big data analysis.