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

Zhiwei Liu 1, Zhaogang Shu 1, Shuwu Chen1, Yiwen Zhong1, Jiaxiang Lin1

Abstract: This paper addresses the problem of mapping, scheduling, and routing of virtual network functions (VNFs) on a service function chain (SFC) that is sensitive to latency in a virtual network. A scheduling algorithm for VNFs 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.

Key words: service function chain, virtual network functions, Delay-aware, VNF scheduling, deep reinforcement learning

I 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 (VNFs) [4]. These VNFs 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 SFCs (e.g., tactile internet services), as these SFCs 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 SFCs, 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 VNFs in the SFC onto nodes that support NFV and map the virtual links between VNFs to the underlying links. The third sub-problem focuses on determining the execution plan for the VNFs in the SFC required to run a given

*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.Zhiwei Liu, Zhaogang Shu, Shuwu Chen, Yiwen Zhong and Jiaxiang Lin are with the computer and information college, Fujian Agriculture and Forestry University, Fuzhou, China. Corresponding Author: Zhaogang Shu(zgshu@fafu.edu.cn).
Although the NFV-RA problem consists of three issues, the placement and scheduling of VNFs have been the main research focus [9]. Regarding the placement of VNFs, there have been several studies. For example, some researchers have investigated the reliability-aware VNF placement problem, where two protection mechanisms were formulated as Integer Linear Programming (ILP) models, and a heuristic algorithm based on dynamic programming was proposed [10]. Hyodo et al. [11] formulated the VNF placement problem as an ILP model, and a heuristic algorithm was proposed to minimize the placement cost and link cost while allowing relaxed constraints on VNF access order and configuration of looped SFC. Alahmad and Agarwal [12] proposed two Mixed Integer Linear Programming (MILP) formulations to solve the cost and availability aspects of VNF placement and type selection, and the proposed solution can reduce the overall cost of requested network service without violating its availability requirements when compared to existing solutions. Feng et al. [13] proposed an advanced heuristic algorithm to migrate VNF to another available node, which effectively improves the utilization and acceptance rate of SFC. Since the existing deep reinforcement learning cannot generalize well to different network topologies, Sun et al. [14] combined deep reinforcement learning and neural networks to improve the generalization ability for different network topologies in VNF placement problems. Laaziz et al. [15] designed a multi-objective integer linear model to solve VNF placement problems with different topological results (linear or nonlinear). Ranbothge et al. [16] proposed two algorithms to solve the VNF placement and position adjustment problems for the placement of VNFs in new service requests and the change of network traffic.

There has been extensive research on the aspects of VNF placement algorithm, which is a crucial problem in network function virtualization. However, VNF scheduling is equally important and has been studied using various problem models and solution methods. For instance, Riera et al. [17] first formulated the VNF scheduling problem as a job-shop scheduling problem, and its mathematical model was presented without proposing a polynomial-time solution. Li and Qian [18] proposed a packet scheduling algorithm taking into account the characteristics of packet queues and the SFC chain's properties due to the low complexity of the algorithm. Chen and Wu [19] designed a processing and delay model, which handled communication delay behaviors in intermediate box processing flows, followed by two corresponding heuristic algorithms designed for scheduling. Mijumbi et al. [20] developed three greedy algorithms and a taboo search algorithm were proposed to solve the problem of VNF placement and scheduling on supporting virtual machines. Assi et al. [21] proposed an effective and energy-saving VNF placement and scheduling method to solve the problem using heuristic algorithms. Li et al. [22] proposed a joint VNF placement and scheduling algorithm, where a two-stage online algorithm was presented to solve the problem. It should be noted that in the aforementioned studies [19][20][21][22], the scheduling of network services does not consider network routing and traffic transmission delays, which are significant factors affecting SFCs that are sensitive to delay in practical environments.

From the above literature, it can be seen that most of the existing research on VNF scheduling focuses more on optimizing the scheduling order, rather than considering the impact of placing VNFs on different nodes. Because the processing and storage capabilities of the CPUs on different nodes vary slightly, the processing time of VNFs is also affected. Additionally, some studies include placement issues do not consider route optimization and transmission delay, which results in the inability to meet strict service deadlines in practical situations. Therefore, we propose a deep reinforcement learning scheduling algorithm to solve the problem of VNF placement and scheduling, to ensure that latency-sensitive network services are completed within strict service deadlines, and to minimize the total number of incomplete SFCs. The main contributions of this article are as follows: (a) Setting up five composite rules and training the D3QN model to obtain the action value of each rule, and then selecting different rules based on the value to determine the SFC with the highest priority. (b) To reduce transmission delay while meeting the timing requirements of SFCs as much as possible, a heuristic algorithm is used to select routes so as to choose the optimal next processing node while meeting the functionality of VNFs. (c) Numerical experiments show that D3QN performs better than composite rules, and the performance is more outstanding compared with traditional DQN.

The remainder of this paper is organized as follows. Section 2 describes the problem and divides it into several sub-problems, while also discussing the interactions and impacts between these sub problems. Section 3 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 4. Finally, conclusions are drawn in Section 5.
II Problem Description and Model

A. Problem Description

The resource allocation of VNFs 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 SFCs.

SFC is a chain of network functions composed of different VNFs based on customer demands at the beginning of the network service phase. These SFCs 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 VNFs 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 SFCs to be completed within the specified deadline. Therefore, create a virtual network to place the VNFs 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 VNFs for their traffic according to the deadline. It is assumed that each VNF instance on each virtual machine can be shared by multiple SFCs, 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 Fig1. Given a set of delay-sensitive SFCs, each composed of K VNFs, the virtual network functions are represented by \( F = \{ f_1, f_2, \ldots, f_m \} \), and \( VNF \) represents the function corresponding to the VNF, where \( 1 \leq t \leq m \). For example, \( VNF \) means that the function of the VNF is \( f_t \), and each \( VNF \) must 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 \( p_t = w / P_t \), where \( w \) is the size of the traffic and \( P_t \) is the processing capability of the VNF. Apart from the processing time of nodes, the time taken for traffic to be transmitted through a link can also be represented as \( D_t = w / b \). \( b \) 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 \( SFC = \{ VNF, w, b, p_t, D_t \} \). \( VNF \) represents the set of VNFs required for the SFC, \( w \) denotes the size of traffic, \( b \) represents the required bandwidth for each virtual link, \( P_t \) is the processing time of the node, and \( D_t \) is the deadline for the SFC. Assuming that there are three SFCs in the example, \( SFC = \{ SFC_1, SFC_2, SFC_3 \} \). \( SFC_1 = \{ (VNF_1, VNF_2), 24Mbps, 15Mbps, 2T, 10T \} \), \( SFC_2 = \{ (VNF_3, VNF_4), 12Mbps, 12Mbps, 3T \} \), \( SFC_3 = \{ (VNF_5, VNF_6), 24Mbps, 6Mbps, 2T, 8T \} \). The three SFCs arrive at the network at \( T = 0 \) as shown in Fig2, and in the first scenario where they are accepted in sequence, the situation is shown in Fig3, They are all mapped to Node 1 at the same time and processed sequentially in the order of \( SFC_1, SFC_2, SFC_3 \), at time \( T = 0 \) to \( T = 2 \), Node 1 completed the processing of the first VNF of \( SFC_1 \), and traffic began to be transmitted through virtual link \( L_1 \) for a duration of 2s. At \( T = 4 \), Node 3 began processing the next VNF, and finally completed \( SFC_1 \) at \( T = 6 \), which is less than the deadline \( D \) of \( SFC_1 \) and meets the transmission delay requirements. Next, we look at \( SFC_2 \). When \( T = 2 \), Node 1 started processing the first VNF of \( SFC_2 \) after completing \( SFC_1 \), and completed it at \( T = 3 \) with a processing time of 1s. It also began traffic transmission through virtual link \( L_1 \), but since virtual link \( L_1 \) was still transmitting traffic for \( SFC_1 \) at this time, the remaining bandwidth (15Mbps−12Mbps < 12Mbps) of the virtual link was not sufficient to meet the bandwidth demand, so \( SFC_2 \) had to wait, at \( T = 4 \) after the transmission of \( SFC_1 \) is completed, \( SFC_2 \) starts to transmit and reaches node 3 at \( T = 5 \).

However, since the processing of \( SFC_1 \) 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 \( SFC_1 \) and is therefore not accepted. Finally, \( SFC_3 \) starts processing at \( T = 3 \) and completes at \( T = 5 \). At this time, the remaining bandwidth of virtual link \( L_1 \) is 15Mbps, which satisfies the bandwidth required for \( SFC_3 \) to transmit. It arrives at node 3 at \( T = 9 \) and finally completes processing at \( T = 11 \), which is the same as \( SFC_3 \). However, the final processing completion time exceeds the deadline of \( SFC_3 \) and cannot be accepted. It is obvious that in this situation, two of the three SFC that entered the network at the same time cannot satisfy their latency requirements and are rejected. The sub-problem impact of scheduling was proposed in [8], but the constraints and solution methods are different from those in this paper. Inspired by this, we describe several scenarios for scheduling the network, ensuring that all SFC are completed before the deadline as much as
possible. These scenarios are also problems that need to be jointly addressed.

Fig 1 network topology

Fig 2 information of SFC

Fig 3 Normal sequential processing

B. Mapping of Virtual Network Functions

In the previous section, the completion times of SFC1 and SFC3 were 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 SFC1 is mapped to node 2 here, SFC1 and SFC3 are processed simultaneously at T=0 as shown in Fig 4. At T=2, SFC1 and SFC3 have both completed processing, SFC1 continues to propagate on the virtual link L1, while SFC3 propagates on the virtual link L3. When T = 7, SFC3 reaches node 2 and begins processing, completing at T=6. Similar to the previous section, SFC3 also completes at T = 7, while SFC3 starts processing at T = 7 and completes at T = 9. Although SFC3 has still not met the expected processing time, compared to the previous scenario, its completion time is much earlier.

Fig 4 Mapping of VNFs

C. Processing Order of VNFs

In sections A and B, SFC3 still failed to complete within the deadline because SFC1 was always processed first, which caused SFC3, 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 SFC3 process traffic first, as shown in Fig 5. At T' = 0, SFC2 and SFC3 start processing at nodes 1 and nodes 2, respectively. At T' = 1, SFC3 completes 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. SFC1 and SFC3 also complete at T' = 7 and T' = 9, respectively.

Fig 5 Processing order of VNFs

D. Routing of Traffic

Although SFC1 and SFC3 met the delay requirement in the previous sections, SFC3 still exceeded the deadline by one time unit. This is mainly because SFC1 arrived at node 3 first and node 3 was idle at that time, so it processed SFC3 before SFC3 arrived, causing SFC3 to wait. In order to solve this problem, we choose a different routing strategy, as shown in Fig 6. As before, at T' = 0, SFC2 and SFC3 start processing at nodes 1 and nodes 2, respectively. After SFC3 processing completes, SFC3 is processed, and the first VNF is completed at T' = 3. Then, SFC3 traffic begins to be transmitted on virtual links L2 and L3 instead of L1, which changes the transmission route. This allows SFC3 to arrive at node 3 first and start processing at T' = 6, completing processing at
$T = 8$, thereby meeting the delay requirement. After SFC processing completes, SFC starts 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.

Fig 6 Routing of Traffic

III 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.

A. 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 policy $\pi$, and a new environment is formed and rewards are obtained by the action acting on the environment, forming a new tuple $(s', r, s, a)$, 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 according to the target values calculated formula as shown in Eq.(1), where $\gamma$ is the discount factor [0,1], $\alpha'$ is the learning rate, and $\theta'$ is the network parameter.

$$y_i = r_i + \gamma \max_a Q(s', a'; \theta')$$  (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. 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):

$$y_i = r_i + \gamma Q(s', \arg \max_a Q(s', a'; \theta'; \theta'))$$  (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. The neural network output of the action value function can be divided into state value function and advantage function. The formula in Eq.(3):

$$Q_\pi(s, a) = V_\pi(s) + A_\pi(s, a)$$  (3)

$V_\pi (s)$ 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 $A_\pi (s, a) = Q_\pi (s, a) - V_\pi (s)$ 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.

B. 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 VNFs, and \( L \) represents virtual links connecting every two nodes. There is a set of SFCs, 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 VNFs while meeting the above requirements, in order to maximize the reception rate of SFCs in the network.

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

Table 1 Parameters and meanings

<table>
<thead>
<tr>
<th>parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G (K, E) )</td>
<td>Represents an SFC forwarding graph,</td>
</tr>
<tr>
<td>( y )</td>
<td>Represents the collection of VNF instances hosted by VMs on the node</td>
</tr>
<tr>
<td>( F )</td>
<td>Type collection of VNF</td>
</tr>
<tr>
<td>( t_f )</td>
<td>VNF instance type ( t_f \in F )</td>
</tr>
<tr>
<td>( C_{ij} )</td>
<td>Available capacity between link ( ij )</td>
</tr>
<tr>
<td>( S )</td>
<td>Collection of SFCs</td>
</tr>
<tr>
<td>( \delta )</td>
<td>Timestamp</td>
</tr>
<tr>
<td>( b_{k,k+1} )</td>
<td>Required capacity of virtual links between k VNF and ( k+1 ) VNF in each SFC</td>
</tr>
<tr>
<td>( w_s )</td>
<td>Flow size of SFC</td>
</tr>
<tr>
<td>( D_s )</td>
<td>Deadline of SFC</td>
</tr>
<tr>
<td>( f_{k} )</td>
<td>The average processing time of the k VNF of this SFC on the node ( n \in N )</td>
</tr>
<tr>
<td>( P^v_{sw} )</td>
<td>The transit time of the traffic on the virtual link</td>
</tr>
<tr>
<td>( j_s \in {0,1} )</td>
<td>Whether the SFC is accepted by the network, that is, whether it can be completed within the specified time, can be 1, otherwise it is 0</td>
</tr>
<tr>
<td>( K^s_{nw} \in {0,1} )</td>
<td>Whether the k VNF of the SFC is mapped to the ( n ) node at time ( \delta ) to start processing, ( n \in N )</td>
</tr>
<tr>
<td>( I^v_{ij} \in {0,1} )</td>
<td>Whether the SFC virtual link ( e \in E ) is routed on the physical link ( (i, j) \in L ), if yes, 1, otherwise 0</td>
</tr>
</tbody>
</table>

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

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

\[
\text{Maximize} \sum_{s \in S} I_s \tag{4}
\]

\[
C_1: \sum_{n \in N} K_{n}^{s} t_f = j_s t_{ks} \\
C_2: \sum_{n \in N} K_{n}^{s+1} k+1 \leq 1 - \sum_{n \in N} K_{n}^{s} k+1 = 1 \\
C_3: K_{n}^{s} = 1 - K_{n}^{s+1} k, k' \in K, s, s', s' \in S \\
C_4: \sum_{k \in K} p_{ns}^k + \sum_{e \in E} p_s^e \leq D_s, s \in S \\
C_5: l_{ij}^{se} b_{k,k+1} \leq c_{ij}, e \in E, (i, j) \in L \\
C_6: \sum_{k \in K} K_{n}^{s} = 1
\]

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 same as the types of VNF instances mapped to the VM. C2 states that the \( (k+1) \) VNF cannot be processed while the k VNF 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's 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 VNFs of the SFC can only be mapped to one node at the same time.

IV. 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.

A. State Definition

Generally, the number of SFCs 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 extract 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.

To facilitate the understanding of the subsequent definitions, this article first defines several parameters: \( n \) represents the total
number of SFCs, \(pd_s\) represents the number of VNFs processed for the S SFC, \(V_S\) represents the number of VNFs in the S SFC, and \(end_s\) represents the time when the last VNF of the S SFC was completed. Therefore, the state is defined as follows:

1. The average completion rate of processed SFC \(ACR\):

\[
ACR = \frac{\sum_{s=1}^{n} \frac{pd_s}{V_S}}{n}
\]

2. The average remaining processing time of SFC \(APT\):

\[
APT = \frac{\sum_{s=1}^{n} (D_s - end_s)}{n}
\]

3. The potential SFC failure rate \(EOR\):

\(EOR\) 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 VNFs) 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:

Algorithm1. 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. \(\text{if } pd_s < V_S \text{ then}\)
4. \(T = 0\)
5. for \(i = pd_s + 1: V_S\) do
6. \(T += (P_{ns}^k + P_S^c)\)
7. end for
8. \(\text{if } end_s + T > D_s \text{ then}\)
9. \(N += V_S - i\)
10. end if
11. end if
12. end for
13. \(EOR = \frac{N}{\sum_{s=1}^{n} V_S}\)
14. return \(EOR\)

(4) Actual incomplete rate \(AOR\) of SFC:

The main algorithmic process of \(AOR\) is to only consider the SFCs that have not been completed at this moment and whose running time has exceeded the deadline. The steps are as follows:

Algorithm2. 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. \(\text{if } pd_s < V_S \text{ then}\)
4. \(\text{if } end_s > D_s \text{ then}\)
5. \(N += V_S - pd_s\)
6. end if
7. end if
8. end for
9. \(AOR = \frac{N}{\sum_{s=1}^{n} V_S}\)
9. return \(AOR\)

B. Action Definition

In most of the existing research, single-rule scheduling algorithms have limited applicability and may not work well for all states. Evolutionary algorithms such as genetic algorithms take too long to process. Therefore, we propose composite rule scheduling, which designs five rules for intelligent agents to select the optimal composite behavior strategy based on the current network state, further improving scheduling algorithm performance.

1. **Rule1**:

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 largest \(end_s\) value 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 VNFs.

2. **Rule2**:

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. **Rule3**:

For Rule 3, we divide it into two cases depending on whether there exists an SFC whose \(end_s\) value exceeds the deadline. If such an SFC exists, the SFC with the smallest slack time is given the highest priority. If not, each SFC’s remaining time is divided by the remaining number of operations, and the smallest value is assigned the highest priority. The specific algorithm is shown below:
Algorithm 3: The process of Rule 3

Input: $V_S, pd_S, D_s, \text{end}_s$

1: O← {S} $pd_S < V_S$ & & $D_s < \text{end}_s$
2: P← {S} $pd_S < V_S$
3: if O = = NULL then
4: \hspace{1em} b = \arg \min_{s \in P} \frac{D_s - \text{end}_s}{V_S - pd_S}$
5: else
6: \hspace{1em} b = \arg \min_{s \in O} \{D_s - \text{end}_s - \sum_{v=\text{pds}} \left( p_{s}^{k} + p_{s}^{e} \right) \}$
7: end if

(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:

Algorithm 4: The process of Rule 4

Input: $V_S, pd_S, D_s, \text{end}_s$

1: O← {S} $pd_S < V_S$ & & $D_s < \text{end}_s$
2: P← {S} $pd_S < V_S$
3: if O = = NULL then
4: \hspace{1em} b = \arg \min_{s \in P} \frac{D_s - \text{end}_s}{V_S - pd_S}$
5: else
6: \hspace{1em} b = \arg \min_{s \in O} \{D_s - \text{end}_s - \sum_{v=\text{pds}} \left( p_{s}^{k} + p_{s}^{e} \right) \}$
7: end if

(5) Rule 5:

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

C. Reward

Since the objective of this article is to minimize the rejection rate, a reward function $R_t$ is as follows:

$$R_t = \alpha \frac{\text{AOR}_t + 1}{\text{AOR}_t} + \beta \frac{\text{EOR}_t + 1}{\text{EOR}_t}$$

(7)

$\alpha$ and $\beta$ are constants that represent the weights of two different features in calculating the reward value $R_t$. When $\frac{\text{AOR}_t + 1}{\text{AOR}_t}$ is greater than 1, it indicates that there is a higher actual unfinished rate in the $S_t + 1$ state than in the $S_t$ state, so this value is set to -1. When it equals 0, the value is designed to be 0. When it is less than 1, it indicates a reduction in the actual unfinished rate in the $S_t + 1$ state, so the reward is 1. The same applies to $\frac{\text{EOR}_t + 1}{\text{EOR}_t}$.

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

$$\min_s \max(N_t, A_t)$$

(8)

$N_t$ represents the completion time of the last VNF processed on node $N$, $A_t$ represents the time when the traffic of the selected SFC arrives at node $N$, $\max(N_t, A_t)$ is used to represent the earliest start time for the traffic to arrive at that node. Therefore, $\min_s \max(N_t, A_t)$ 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 determine $A_t$, 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 determine $A_t$. 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 Fig7(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:

Algorithm 5: Node Selection and Routing Optimization

1: \( M = [\] \\
2: for \( m = 1: N \) do \\
3: \( \text{if This node satisfies constraint (2) then} \) \\
4: \( L = [], A_1 = end \) \\
5: \( L\leftarrow \text{Dijkstra, Find the shortest time path from the previous node of the SFC to the candidate node and keep the pass nodes} \) \\
6: for \( s = L \) do \\
7: \( A_s = (P_n^k + P_n^e) \) \\
8: end for \\
9: end if \\
10: \( M = \text{max}(N_r, A_1) \) \\
11: end for \\
12: selected node = \text{Arg min } M

The traffic shown on Fig7(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 3 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 Fig7(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:

Algorithm 5: Node Selection and Routing Optimization

1: \( M = [\] \\
2: for \( m = 1: N \) do \\
3: \( \text{if This node satisfies constraint (2) then} \) \\
4: \( L = [], A_1 = end \) \\
5: \( L\leftarrow \text{Dijkstra, Find the shortest time path from the previous node of the SFC to the candidate node and keep the pass nodes} \) \\
6: for \( s = L \) do \\
7: \( A_s = (P_n^k + P_n^e) \) \\
8: end for \\
9: end if \\
10: \( M = \text{max}(N_r, A_1) \) \\
11: end for \\
12: selected node = \text{Arg min } M

V. 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.

A. 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 SFCs 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 SFCs 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 VNFs. For each SFC, random traffic generation ([25-75] Mbits), bandwidth requirements ([15-25] Mbps), and variable VNFs ([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].
B. 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 Fig8 and Fig9:

![Figure 8](image1.png)
**Fig8** the SFC rejection rate with 15 nodes

![Figure 9](image2.png)
**Fig9** the SFC rejection rate with 30 nodes

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.

C. 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 shown in Fig 10 and Fig 11.

![Figure 10](image3.png)
**Fig10** the SFC rejection rate with 15 nodes

![Figure 11](image4.png)
**Fig11** the SFC rejection rate with 30 nodes

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-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.

VI 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 VNFs on virtual machines. In addition, the scheduling rules will be optimized, or other more advanced policy-based RL methods.

References:


