Service-Aware Virtual Network Function Migration Based on Deep Reinforcement Learning

Zeming Li(✉), Ziyu Liu, Chengchao Liang, and Zhanjun Liu

School of Communication and Information Engineering,
Chongqing University of Posts and Telecommunications, Chongqing, China
1964625523@qq.com

Abstract. Network Function Virtualization (NFV) aims to provide a way to build agile and flexible networks by building a new paradigm of provisioning network services where network functions are virtualized as Virtual Network Functions (VNFs). Network services are implemented by service function chains, which are formed by a series of VNFs with a specific traversal order. VNF migration is a critical procedure to reconfigure VNFs for providing better network services. However, the migration of VNFs for dynamic service requests is a key challenge. Most VNF migration works mainly focused on static threshold trigger mechanism which will cause frequent migration. Therefore, we propose a novel mechanism to solve the issue in this paper. With the objective of minimizing migration overhead, a stochastic optimization problem based on Markov decision process is formulated. Moreover, we prove the NP-hardness of the problem and propose a service-aware VNF migration scheme based on deep reinforcement learning. Extensive simulations are conducted that the proposed scheme can effectively avoid frequent migration and reduce the migration overhead.

Keywords: Network Function Virtualization · Virtual Network Functions · Markov decision process · Migration · Deep reinforcement learning

1 Introduction

The explosive growth of Fifth Generation networks and their complexity exceeds the limitations of manual management. The growing demand for agile and robust network services has driven the telecom industry to design innovative network architectures with Network Functions Virtualization (NFV) and Software Defined Networking (SDN). NFV is the enabler supporting concept that is proposed to decouple network functions from proprietary hardware devices. NFV

H. Gao et al. (Eds.): ChinaCom 2021, LNICST 433, pp. 481–496, 2022.
https://doi.org/10.1007/978-3-030-99200-2_36

has the potential to significantly reduce operational and capital expenditures, and improve service agility compared to traditional network functions implemented by dedicated hardware devices. SDN aims to enable centralized network control using software running on common hardware rather than proprietary hardware devices. According to academia and industry, the combination of NFV and SDN is expected to revolutionize network operations, not only by dramatically reducing costs, but also by introducing new possibilities for enterprises, carriers and service providers [1].

With the extension of global Internet services, Cisco predicts that the global mobile networks will support more than 12 billion mobile devices by 2022 with mobile traffic approaching zettabytes (ZB). Network operators will further extend the depth and breadth of communication coverage of next-generation communication networks (e.g., 5G, 5G beyond, and 6G) to support multifarious network services for diversified QoS requirements [2]. Meanwhile, with the rapid development of new applications and services, it is indispensable to provide and deploy new devices continuously, which will result in the network resource and energy consumption of Data Center Networks (DCNs) increase rapidly [3]. By aggregating satellite networks to boost current network capacity, enhance system robustness, and extend ubiquitous 3D wireless coverage. Therefore, it is essential to develop satellite-terrestrial integrated network (STIN) to leverage the complementary benefits of disparate networks to realize seamless, robust, and reliable network services provisioning [4].

In SDN/NFV-enabled STIN, network services can be implemented with Service Function Chains (SFCs), which consist of a series of chained Virtual Network Functions (VNFs) by highly predefined sequences. These VNFs are instantiated and executed at heterogeneous substrate nodes (i.e., NFV nodes). When the demand of VNFs exceeds their mapped substrate nodes and links resource capacity, the network functions will be invalidated, which will seriously affect the success rate of SFCs deployment and services performance. In order to accommodate the time-varying service demands, it is inevitable to orchestrate the VNF migration among substrate nodes to improve service performance. However, most of the current works mainly focus on static threshold triggered mechanisms [5–10]. In a real-DCN scenario, service requests arrive at the system randomly with diversified QoS requirements, and the network resource conditions of the underlying NFV infrastructure (NFVI) dynamically vary over time. A static migration solution is unable to satisfy the dynamic property of the network and leads to function invalidation or service degradation.

The wide variety of challenges introduced by the disruptive deployment of STIN trigger the need for a drastic transformation in the way services are managed and orchestrated. It can become time-consuming to orchestrate services due to manual configurations around VNF migration. The intention is to have all operational processes and tasks, such as resource monitoring, VNF mapping, resource configuration, and optimization, executed automatically. Since the world is already moving towards data-driven automation, technologies like Machine Learning can be employed to tackle part of the workload. New advancements in Deep Reinforcement Learning (DRL), will pave the way for a more intelligent and self-organizing network.

Despite the significant advancements in networking thanks to the introduction of NFV and SDN technologies, the migration and orchestration tools still require additional research and development, to reach an acceptable level of automation. To this end, the motivations of this paper can be unfolded in three aspects. Firstly, to avoid substrate nodes and links overload, and guarantee service continuity with randomly arrive service requests, some VNFs should be migrated from poor nodes for the utilization of resources rationally. Secondly, existing migration mechanisms are prone to frequent migration, a reasonable mechanism should be designed to reduce the times of invalid migration and alleviate network overhead. Finally, we should consider how to optimize the average performance metrics at long-time scale through VNF migration for SFC with a certain life cycle and various resource constraints.

Given the above considerations, we investigate the VNF migration problem by taking migration triggering mechanism and migration overhead into account. Specially, rather than focus on the detailed migration process, we primarily concentrate on the migration overhead of the network. Since the appropriate deployment of VNF can increase the utilization of network resources, enhance the robustness of the system, and improve the adaptation to environmental changes [11]. Our intention is to optimize migration overhead for all affected SFCs. For this purpose, the main contributions in this paper can summarize as follows.

- According to the characteristics of VNF migration and traffic dynamics, we propose a dynamic threshold trigger mechanism to avoid the frequent migration and system instability incurred by the existing migration trigger mechanism.
- Basic on this mechanism, we model the migration overhead as power consumption and migration latency caused by migration jointly. Besides, a stochastic optimization problem based on Markov Decision Process (MDP) is formulated to describe the migration process of VNFs. Furthermore, a service-aware VNF migration scheme based on DRL is proposed.
- Extensive simulation results are provided to demonstrate that the validity of our scheme and a series of comparative experiments with the existing works also presented.

The remainder of this paper is organized as follows. In Sect. 2, we illustrate the system model and formulate the VNF migration problem. Section 3 models the migration problem as MDP, and then we employ DRL-based techniques to solve the optimization problem. Simulation analyses and results are presented in Sect. 4 and conclusions are drew in Sect. 5.

2 System Model and Problem Formulation

Considering a network scenario which is constructed upon SDN/NFV-enabled STIN network infrastructure in which physical resources are provided to services through resource virtualization, as shown in Fig. 1. To simplify the analysis, we

split the continuous decision time into time slots t and characterized by $\mathcal{T} = \{1, 2, ..., T\}$. When a service request arrives, the NFVI provides the appropriate NFVI node in terms of the resource requirements of the VNFs.

2.1 Network Model

The substrate network topology is modeled as undirected graph $\mathcal{G}^S = (\mathcal{V}^S, \mathcal{L}^S)$, where $\mathcal{V}^S = \{v_i | i = 1, ..., V\}$, $\mathcal{L}^S = \{l_{ij} | i, j = 1, ..., V, i \neq j\}$ represent the set of substrate nodes and links respectively. Notably, the substrate nodes contain satellite nodes $\mathcal{V}_S^S$ and terrestrial nodes $\mathcal{V}_T^S$ in STIN, i.e., $\mathcal{V}^S = \mathcal{V}_S^S \cup \mathcal{V}_T^S$. The nodes are characterized by computing resource capacity C_i^{max}, and memory capacity M_i^{max}, where $i \in \{1, 2, ..., V\}$. The link l_{ij} between nodes v_i and v_j is characterized by the bandwidth capacity $B_{(i,j)}^{max}$. At the time slot t, the resource occupancy can be denoted to $U^*(t) = \{U_i^C(t), U_i^M(t), U_{ij}^B(t)\}$ and the resource capacity is represented as $\Omega^* = \{C_i^{max}, M_i^{max}, B_{(i,j)}^{max}\}$ respectively, where $*$ represents the types of resource.

We abstract the virtual network as an ordered set $\mathcal{G}^V = \{\mathcal{V}^V, \mathcal{L}^V\}$, where $\mathcal{V}^V$ and $\mathcal{L}^V$ represent the set of VNFs and virtual links respectively. There are Q SFCs in virtual network are denoted as the set $\mathcal{S} = \{S_q | q = 1, 2, ..., Q\}$, and the qth SFC contains N_q types of VNFs, which is denoted to $S_q = \{V_{q,1}, V_{q,2}, ..., V_{q,N_q}\}$, $\forall V_{q,n} \in \mathcal{V}^V$. Each SFC is formed by diversified VNFs with a particular order and mapped to NFVI in term of resource requirements. Each VNF has a finite computing resource requirement assumed as $c^{q,n}(t)$ and caching resource requirement $m^{q,n}(t)$. The virtual link between adjacent VNFs in any SFC S_q are represented as $l_{(n,n+1)}^q \in \mathcal{L}^V$. Similarly, the finite bandwidth requirement of each virtual link $l_{(n,n+1)}^q$ is denoted by $b^{q,(n,n+1)}(t)$. Generally, SFC S_q has the maximum end-to-end latency limit, which can be set as D_q^{max}.

In order to describe the mapping relationship of VNFs, we define the binary variable $x_i^{q,n}(t) \in \{0, 1\}$, where $x_i^{q,n}(t) = 1$ means that $V_{q,n}$ is mapped into the substrate node v_i and $x_i^{q,n}(t) = 0$ otherwise. The mapping relationship of VNFs will change after each migration occurs. In the same way, we introduce the binary variable $z_{ij}^{q,(n,n+1)}(t) \in \{0, 1\}$ to indicate whether the virtual link $l_{(n,n+1)}^q$ is mapped onto the substrate link l_{ij}, where $z_{ij}^{q,(n,n+1)}(t) = 1$ means that the virtual link $l_{(n,n+1)}^q$ is mapped onto the physical link l_{ij} and $z_{ij}^{q,(n,n+1)}(t) = 0$ otherwise.

2.2 Dynamic Threshold Trigger Mechanism

When the resource occupancy reaches the thresholds, the migration is triggered immediately which can lead to frequent migration and reduce the stability of the system. For solving this problem, this paper firstly proposes a dynamic threshold trigger mechanism to evaluate whether to perform migration immediately at time slot t by Markov chains.

The resource demand interval for arbitrary $V_{q,n}$ is assumed as $[\Phi_{q,n}^{min}, \Phi_{q,n}^{max}]$. The system first allocates resources with $\Phi_{q,n}^{max}$ to each function in the absence

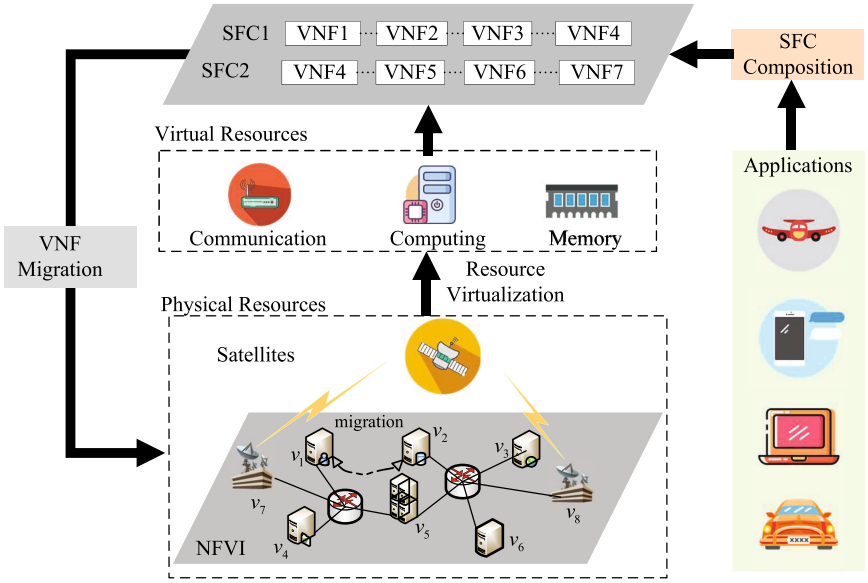

Fig. 1. System model

of historical information and prior knowledge. As the system runs, the future resource demand can be estimated in the short term through Markov chains after obtaining enough simple data of $V_{q,n}$. To model the resource estimation under each time slot t, we divide the resource demand interval $[\Phi_{q,n}^{min}, \Phi_{q,n}^{max}]$ into W sections, and the capacity of resources in each section is $\lambda^{q,n} = \frac{\Phi_{q,n}^{max} - \Phi_{q,n}^{min}}{W}$. Each section represents a different resource demand state, that is, the W sections map to W states of $V_{q,n}$, and the resource capacity under each state is equal to the average value of the section. Thereby, the resource state $\lambda_w^{q,n}(t), w \in 1, 2, ..., W$ of $V_{q,n}$ can be considered as a uniform continuous variation in time slot t. Then, the one-step Markov state transition matrix of $V_{q,n}$ is obtained by the statistical method of

$$\mathcal{P}^{q,n} = [p_{ww'}^{q,n}], w, w' = 1, 2, ..., W, \forall n, q, \tag{1}$$

where $p_{ww'}^{q,n} = \frac{\kappa_w}{\kappa_{ww'}}$ represents as the probability of transition from the state $\lambda_w^{q,n}$ to $\lambda_{w'}^{q,n}$, κ_w represents the times of the demand state maintains in $\lambda_w^{q,n}$, and $\kappa_{ww'}$ represents the times of the state $\lambda_w^{q,n}$ is transferred to $\lambda_{w'}^{q,n}$.

On this basis, the resource demands of $V_{q,n}$ in the time slot $t+1$ can be estimated from the Chapman-Kolmogorov equation by building a finite-state discrete Markov chain model [12]. The arbitrary possible demand state transition probability can be estimated as

$$\mathcal{F}_W^{q,n} = \mathcal{F}_{W-1}^{q,n}(\mathcal{P}^{q,n}) = ... = \mathcal{F}_0^{q,n}(\mathcal{P}^{q,n})^W, \tag{2}$$

where $\mathcal{F}_0^{q,n}$ represents the initial probability distribution of $V_{q,n}$. Thus, the probability of resource demand at time-slot $t+1$ can be expressed as $\mathcal{F}_0^{q,n}(\mathcal{P}^{q,n})^W$.

In order to smooth the migration process and determine the optimal timing of migration, we assume that the resource capacity threshold as η^{max}, and denote a migration factor ζ to estimate the probability of overload and its duration in the next time period $t+1$. By this means, the migration is triggered when and only when ζ in the time slot $t+1$ is greater than η^{max}, where,

$$\zeta = \frac{\sum_{w=0}^{W} \mathbb{I}(U^*(w) > \eta^{max}\Omega^*)}{W}, \tag{3}$$

where $\mathbb{I}(\cdot)$ is the indicator function. When $\zeta > \eta^{max}$, it indicates that the substrate nodes or links are unable to meet the service requirements and migration should be executed immediately. By this way, the migration is triggered only when ζ of the current time-slot is greater than a certain value and lasts for a period of time.

2.3 Problem Formulation

In this section, we formulate an optimization model for the VNF migration problem. Due to the VNF migration consumes certain energy and increases network latency, we have considered that the power consumption and migration latency of NFVI jointly. Based on literature [13,14], supposing that a linear model of the power consumption versus the traffic handled by the nodes. Hence the power consumption of node v_i at time slot t can be expressed as follows,

$$P_{v_i}(t) = [\delta + (1-\delta)\frac{c^{q,n}(t)}{\rho}]P^{max}. \tag{4}$$

In formula (4), P^{max} is the node maximum power consumption, and $\delta \in [0,1]$ is the ratio of the baseline power to the maximum power [5]. We assume that all of the nodes are equipped with identical maximum data computing capacity ρ, which represents the number of CPU cycles that can be processed per second [15]. Hence, the power consumption of SFC S_q at time slot t is given by

$$P_q(t) = \sum_{n=1}^{N_q} x_i^{q,n}(t)[1 - x_i^{q,n}(t+1)]P_{v_i}(t). \tag{5}$$

The migration latency of SFC S_q is related to the location of the SFC virtual link mapped to the substrate link, then the migration latency of SFC S_q at time slot t is the sum of the migration latency of all links, so the migration latency of SFC S_q can be expressed as

$$D_q(t) = \sum_{n=1}^{N_{q-1}} \frac{m^{q,n}(t) z_{(i,j)}^{q,(n,n+1)}(t)[1 - z_{(i,j)}^{q,(n,n+1)}(t+1)]}{b^{q,(n,n+1)}(t)}. \tag{6}$$

Consequently, the total migration overhead of SFC in time slot t can be normalized and expressed as following,

$$C(t) = \omega_1 \frac{\sum_{q=1}^{Q} P_q(t)}{P^{max}} + \omega_2 \frac{\sum_{q=1}^{Q} D_q(t)}{D_q^{max}}, \tag{7}$$

where ω_1 and ω_2 are the corresponding weights, and $\omega_1 + \omega_2 = 1$. Therefore, the VNF migration problem can be formulated as following,

$$
\begin{aligned}
\mathbf{P1:} & \min_{x,z} \frac{1}{T} \sum_{t=1}^{T} C(t) \\
s.t.\ & C1 : x_i^{q,n}(t) \in \{0,1\}, \forall i, q, n \\
& C2 : z_{(i,j)}^{q,(n,n+1)}(t) \in \{0,1\}, \forall i \neq j, q, n, \\
& C3 : \sum_{i=1}^{V} x_i^{q,n}(t) = 1, \forall i, q, n \\
& C4 : \sum_{q=1}^{Q} \sum_{n=1}^{N_q} c_{q,n}(t) x_i^{q,n}(t) \leq C_i^{max}, \forall i, q, n \\
& C5 : \sum_{q=1}^{Q} \sum_{n=1}^{N_q} m_{q,n} x_i^{q,n}(t)(t) \leq M_i^{max}, \forall i, q, n \\
& C6 : \sum_{q=1}^{Q} \sum_{n=1}^{N_q - 1} b_{n,n+1}^{q}(t)(z_{i,j}^{q,(n,n+1)}(t) + z_{j,i}^{q,(n,n+1)}(t)) \leq B_{(i,j)}^{max}, \forall i \neq j, q, n, \\
& C7 : \sum_{j} z_{i,j}^{q,(n,n+1)}(t) - \sum_{j} z_{j,i}^{q,(n,n+1)}(t) = x_i^{q,n}(t) - x_i^{q,n+1}(t), \forall i \neq j, q, n, \\
& C8 : D_q(t) \leq D_q^{max}, \forall q
\end{aligned}
\tag{8}
$$

The constraint C1 and C2 restrict the variable $x_i^{q,n}(t)$ and $z_{(i,j)}^{q,(n,n+1)}(t)$ to binary choice. The constraint C3 to avoid nodes overload and routing loops. The constraint C4∼C6 ensure the computing, caching and bandwidth demand of the VNF deployed in each node cannot exceed the maximum resource capacity. The constraint C7 guarantees that an arbitrary virtual link must exist a continuous path $l_{(i,j)}$ between substrate nodes v_i and v_j to which the adjacent $V_{q,n}$ and $V_{q,n+1}$, and maintains the traffic that inflow must be equal outflow. In order to make sure the link delay between VNF nodes in any SFC S_q satisfies the SFC end-to-end latency requirement, the constraint C8 is proposed.

The optimization problem **P1** with constraints C1 ∼ C8 is NP-hard. We can proof as following: we take the optimization problem **P1** within a fixed time slot t into account, which can reduce **P1** into the Multiple-choice Multidimensional Knapsack Problem (MMKP) [16]. Similar to [16], we consider the network resource constraints of substrate nodes and links, migration mapping variables as size vectors, as shown in constraints C1∼C5. It's not too difficult to spot that the special case is NP-hard. Moreover, problem **P1** is a dynamic MMKP problem with t varying, so problem **P1** is NP-hard.

3 Migration Decision Based on DRL

In this section, we introduce the theory of MDP firstly. Then model the optimization problem **P1** to MDP and the detailed state, action and reward of MDP in this paper are defined respectively. Furthermore, the service-aware VNF migration algorithm based on Twin Delayed Deep Deterministic Policy Gradient (TD3) is introduced in detail.

3.1 Markov Decision Process

In DCN and mobile communication networks, there are many traditional solutions for the optimization problem of VNF migration, such as exhaustive search and game theory [17,18]. However, these techniques have many drawbacks. On one hand, high algorithm complexity brings about low efficiency. On the other hand, it is hard to apply to large scale network scenarios due to heavy computational resource consumption and low redundant fault tolerance. In this paper, we take RL method to solve the joint optimization problem while avoiding imperfection and abuses of conventional algorithms to some extent.

Our goal is to optimize the migration overhead of VNF migration. Therefore, VNF migration requires a series of decisions to reach the final object, and thus, it is a sequential decision problem, where each migration has an impact on the subsequent one. Traditional mathematical modeling methods are difficult to accurately model the process, and therefore, solving it by traditional methods will make the problem extremely complex. MDPs is a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent states, and through those future rewards. Reinforcement learning is an effective class of methods used to solve MDPs.

We model the VNF migration as the MDP to describe migration process firstly. The transition process of VNFs on NFV nodes for each time slot t, described as a quadruple $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R})$:

- ***States S***: Each state $s(t) \in \mathbf{S}$ at time-slot t as $s(t) = \{v(t), c(t), m(t), b(t) | v \in \mathbb{V}, c \in \mathbb{C}, m \in \mathbb{M}, b \in \mathbb{B}\}$, where $\mathbb{V}$ denotes the substrate network topology space and $\mathbb{C}, \mathbb{M}, \mathbb{B}$ state the computing, caching and bandwidth demands state spaces of the VNFs, respectively.
- ***Action A***: $a(t) = \{x(t), z(t) | x \in \mathbb{X}, z \in \mathbb{Z}\}$, where $x(t) = \{x_i^{q,n}(t)\}, z(t) = \{z_{ij}^{q,(n,n+1)}\}, \forall i, j, q, n$, $a(t) \in \mathbf{A}$, $\mathbb{X}$ and $\mathbb{Z}$ represent the mapping action space of the VNFs and virtual links, respectively.
- ***Transition probability P***: After taking action $a(t)$ in time slot t, state $s(t)$ will be transferred to the next state $s(t+1)$ with state transition probability $p(s(t+1)|s(t), a(t))$.
- ***Reward R***: In time slot t, the DRL agent can obtain a reward based on the current state and selected action. In this paper, our optimization goals is to minimize migration overhead, so we define the additive inverse of migration overhead as the immediate reward for time slot t, i.e., $r(t) = -C(t)$.

The cumulative discount reward at time slot t is defined as

$$R_\pi(t) = \lim_{T \to +\infty} E_\pi\{\sum_{t=1}^{T} \gamma^{t-1} r(t)\}, \tag{9}$$

where π denotes the policy of taking action $a(t)$ at state $s(t)$. Agent in state $s(t)$ will takes an action $a(t) = \pi(s(t))$ to move the next state $s(t+1)$ according to policy π. The performance of the policy at time slot t is evaluated by the action value function $Q_\pi(s(t), a(t))$. Under the policy π, the expected reward for a given action and state can be expressed as

$$Q_\pi(s(t), a(t)) = E_{a \sim \pi}[R_\pi(t)|s(t), a(t)]. \tag{10}$$

Generally, $Q_\pi(s(t), a(t))$ can be iteratively expressed through the Bellman equation,

$$Q_\pi(s(t), a(t)) = E_{a \sim \pi}[r(t) + \gamma Q_\pi(s(t+1), a(t+1))]. \tag{11}$$

There must exist a stable optimal policy that maximizes the reward when $T \to +\infty$, so the optimal VNF migration policy π^* can be expressed as

$$\pi^* = \arg \max_a Q_\pi(s(t), a(t)). \tag{12}$$

In order to obtain the optimal policy π^*, the RL agent needs to select actions in each state to maximize the $Q_\pi(s(t), a(t))$.

3.2 Migration Scheme Based on DRL

According to the analysis in Sect. 3.1, the optimal policy π^* is obtained from the optimal action of different states of equation (12). Since the service requests as well as data flow arrivals in real-DCNs are random, the optimal policy cannot be obtained by iterating the Bellman equation for the action-value function. In this paper, we propose a service-aware VNF migration optimization scheme based on TD3 to solve the problem **P1**. The detailed scheme framework is described as Fig. 2.

Given the system state $s(t)$ at time slot t, the actor primary network selects a deterministic action $a(t) = \pi(s(t)|\phi) + \mathcal{N}_t$ according to the policy π, where ϕ is a parameter of the actor primary network, $\mathcal{N}_t$ is the OU noise introduced to increase the exploratory. After taking $a(t)$ in the network, the agent return the feedback of the immediate reward $r(t)$ and new state $s(t+1)$ to the actor network, and stores the transition tuple (s, a, r, s') to the experience replay pool $\mathcal{B}$, which will be used in the training processes are stated in the following.

Critic Network Training and Learning. The critic network evaluates the performance of the policy by minimizing the loss function $L(\theta_i)$, i.e.,

$$\min L(\theta_i) = E[(y_i - Q_{\theta_i}(s(t), a(t)|\theta_i))^2], i = 1, 2, \tag{13}$$

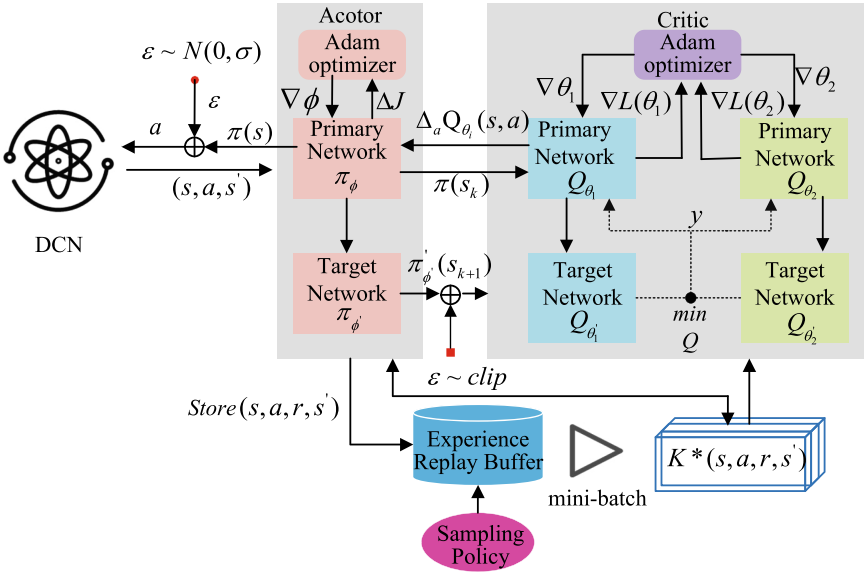

Fig. 2. The framework of our proposed scheme

where $y_i = r + \gamma\, min[Q_{\theta_i}(s'), \pi'_{\phi'}(s' \mid \phi') \mid \theta')]$ is the target Q-value after the target action $\tilde{a}(t+1) = \pi'_{\phi'}(s(t+1)|\phi')$ is taken in state s'. Notice that the target Q-value is obtained from the target network with parameters $\pi'_{\phi'}$ and $Q'_{\theta'_i}$. The minimum of $L(\theta_i)$ solving by stochastic gradient descent (SGD), and the gradient of $L(s(t), a(t))$ in relation to θ_i is calculated as,

$$\nabla_{\theta_i} L(s(t), a(t)) = E[2(y - Q_{\theta_i}(s(t), a(t) \mid \theta_i))\nabla Q_{\theta_i}(s(t), a(t))], i = 1, 2, \quad (14)$$

where $\nabla Q_{\theta_i}(s(t), a(t))$ is calculated by the chain-rule from the output return to each critic network parameter with θ_i. We observed that in formula (14) that $[y - Q_{\theta_i}(s(t), a(t) \mid \theta_i)]$ that is the TD-error actually. In fact, we usually randomly extract $K*(s_k, a_k, r_k, s_{k+1})$ mini-batch samples from the experience replay pool $\mathcal{B}$ to train the critic network with SGD, i.e.,

$$\theta_i = \theta_i - \frac{\alpha}{K}\sum_{k=1}^{N}[2 \cdot (y_k - Q_{\theta_i}(s_k, a_k \mid \theta_i))\nabla Q_{\theta_i}(s_k, a_k)], i = 1, 2, \quad (15)$$

where α states the learning rate of the critic network, and

$$y_k = r_k + \gamma \min_{i=1,2} Q_{\theta_i}(s_{k+1}, \pi'_{\phi'}(s_{k+1}|\phi')|\theta'_i). \quad (16)$$

Actor Network Training and Learning. The goal of the actor network is to obtain a policy that maximizes discount rewards, i.e.,

$$\max_{\pi} R_\phi(t) = E[(a^*(t) - a(t))^2], \quad (17)$$

where $a(t) = \pi_\phi(s(t)|\phi)$, $a^*(t)$ denotes the optimal action. The parameter ϕ of the actor network is updated using the deterministic policy gradient (DPG) maximizing Q value of the outputted action [19]. To this end, the gradient form the critic network concerning the actor network's output action $\tilde{a}(t) = \pi(s(t)|\theta_\pi)$, namely, $\triangle_{\tilde{a}}\ Q(s(t), \tilde{a}(t))$. The gradient of actor network can written as,

$$\triangle_\phi\ Q(s(t), a(t)) = \triangle_a\ (Q_{\theta_i}(s(t), a(t))|_{a=\pi_\theta(s)})\ \triangle_\phi\ \pi_\theta(s(t)), i = 1, 2, \tag{18}$$

where $\triangle_\phi\ \pi_\theta(s(t))$ is derived by the chain-rule. The parameters ϕ is updated by first training the DNN parameters with a randomly sampling from the pool $\mathcal{B}$, and then iteratively updating the policy with the goal of maximizing the total discounted reward. According to the literature [20], the stochastic gradient policy is the gradient of the policy performance, which is equivalent to the empirical DPG, i.e.,

$$\nabla J(\phi) = \sum\nolimits_S d(S) \sum\nolimits_A \nabla Q_{\theta_i}(s(t), a(t)) \nabla \pi_{\theta_i}(s(t)), i = 1, 2, \tag{19}$$

where $d(S)$ states the state distribution, and $\nabla Q_{\theta_i}(s(t), a(t))$ is approximated by the critic primary network and can be transformed into

$$\nabla_\phi J(\phi) \approx \nabla_\phi Q^* = E_\pi \nabla_a (Q_{\theta_i}(s(t), a(t))|_{a=\pi_\phi(s(t))})\ \triangle_\phi\ \pi_\phi(s(t)), i = 1, 2. \tag{20}$$

At each training step, by updating the DNN parameters along the direction of gradient ascent of the target policy function, the parameter ϕ is updated by randomly extract $K * (s_k, a_k, r_k, s_{k+1})$ samples from the $\mathcal{B}$, that is,

$$\phi = \phi + \frac{\beta}{K} \nabla_a (Q_{\theta_i}(s_k, a_k)|_{a_k=\pi_\phi(s_k)})\ \triangle_\phi\ \pi_\phi(s_k), i = 1, 2, \tag{21}$$

where β represents the learning rate of the actor network. Noticed that in the training of the actor network, only the state s_k of the pool $\mathcal{B}$ and the actions generated by the actor network for that state are required. According to the literature [21], direct updating of the network likely causes network instability. Therefore, the target network parameters are updated by Polyak Averaging, i.e.,

$$\begin{aligned} \phi' &= \tau\phi + (1 - \tau)\, \phi' \\ \theta_i{}' &= \tau\theta_i + (1 - \tau)\, \theta_i{}', i = 1, 2, \end{aligned} \tag{22}$$

where τ controls the magnitude of the update, and reflects update stability. The details of the proposed algorithm are illustrated in Algorithm 1.

4 Simulation Results

To evaluate the effectiveness and reliability of the proposed scheme in this paper, simulation validations of the proposed scheme and performance comparison with previous work are performed. We compared the FDQ scheme in [22], our proposed scheme based on the dynamic threshold trigger mechanism called Sa-VNFM and the method based on TD3 (TD3-based). The performance comparison has been carried out on a computer characterized by 2.50 GHz Intel(R)

Algorithm 1: Service-aware VNF migration algorithm (Sa-VNFM)

Input: policy update episode T, policy frequency d, experience replay buffer $\mathcal{B}$ capacity B, sampling size K, discount factor γ, soft update factor τ,learn rate α, β

Output: policy π^*

Initialization: Set the parameters of actor's online network ϕ and critic's online network θ_1, θ_2 randomly, and set the target network parameters by $\phi' \leftarrow \phi$, $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$. Set experience replay buffer $\mathcal{B}$ capacity $B = 0$;
for $t = 1$ *to* T **do**
  Select action with exploration noise $a_t \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$
  if *constraints C3∼C8 and* $\zeta > \eta^{max}$ *are satisfied* **then**
    Agent execute a_t from the simulation environment, and observe the immediate reward r_t and the new state s_{t+1}
    if *experience replay buffer* $\mathcal{B}$ *not exceed capacity* B **then**
      Store the transition tuple (s_k, a_k, r_k, s_{k+1}) into $\mathcal{B}$ as the train sample for the online network
    else
      Randomly replace any tuple in $\mathcal{B}$
    end
    Randomly sample K transition tuples as mini-batch training data for training and learning
    According to (13)-(14), calculate the gradient $\nabla_{\theta_i} L$ of critic's online networks
    Update the parameters θ_i of critic's online network with the Adam optimizer by (15)
    if t *mode* d **then**
      According to (18)-(20), calculate the policy gradient $\Delta_\phi J(\phi)$ of actor's online network
      Update the parameters ϕ of critic's online network with the Adam optimizer by (21)
      Soft update the parameters ϕ', θ_1', θ_2' of target network by (22)
    end
  end
end

Core(TM) i5-7200U CPU processor and by an 8 GB memory, and the simulation tool we used the Pycharm integrated development environment with TensorFlow 1.14.

4.1 Experiment Settings

Due to the high computational complexity of the optimal problem, we consider a fully connected DCN scenario, which contains 11 generic terrestrial nodes and 1 satellite node. The computing capacity and memory capacity of all the substrate node is uniformly distributed in $[100, 300]$ MIPS and $[150, 300]$ MB respectively, and the bandwidth capacity of each link is generated randomly in the $[50, 100]$ Mbps.

Table 1. Main simulation parameters

Parameters	Value
Number of satellite node	1
Number of terrestrial nodes	11
Node computing capacity ($MIPS$)	Uniform[100, 300]
Node memory capacity (MB)	Uniform[150, 300]
Link bandwidth capacity ($Mbps$)	Uniform[50, 100]
VNF computing requirement ($MIPS$)	[10, 20]
VNF caching requirement (MB)	[10, 30]
Virtual link bandwidth requirement ($Mbps$)	[1, 10]
SFC maximum time latency limit (ms)	50
Maximum data computing capacity ($MIPS$)	20
Node maximum power (W)	200
Experience replay pool capacity	10000
Learning rate α, β	0.002
Discount factor	0.99
Soft update factor	0.001
Mini-batch	32

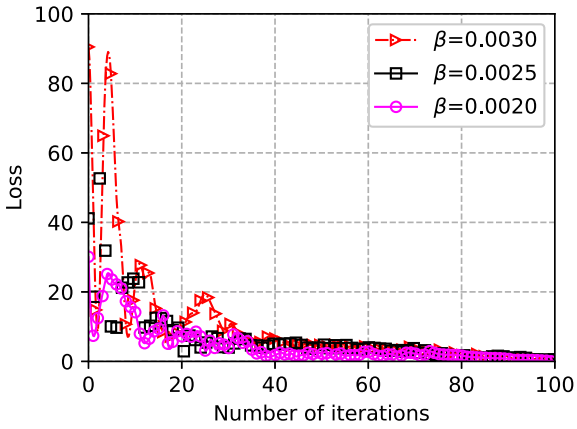

Fig. 3. Convergence of the proposed scheme.

In order to simulate the real environment, the resource demand of SFCs is simulated by using a Poisson process to accord with the characteristics of periodic and bursty real data flows. Assuming each of SFC consists of different types of VNFs conforming to a uniform distribution of $[3, 6]$. The relevant simulation parameters are shown in Table 1.

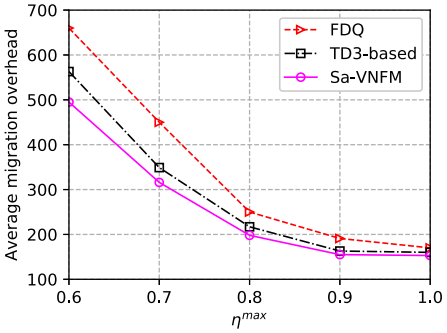

Fig. 4. Migration overhead versus η^{max}.

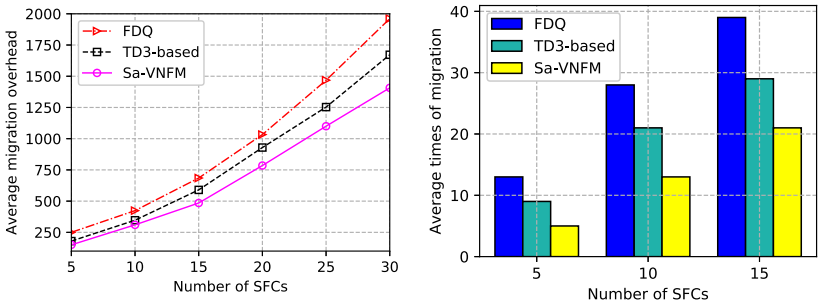

Fig. 5. Average migration overhead and times with different network setups.

4.2 Numerical Results and Discussions

To verify the convergence of the algorithm proposed by adjusting the β and while keeping the α constant. As shown in Fig. 3, when $\beta = 0.003$, the loss function fluctuates greatly, which easily causes the gradient to hover around the minimum value and makes it difficult to obtain the optimal solution. As β decreases, the jitter becomes less accordingly, and the loss function convergence rate becomes slow, making the loss curve smooth. Therefore, $\beta = 0.002$ is selected in this paper.

The migration overhead versus different thresholds is shown in Fig. 4. In order to analyze more specifically the effect of threshold setting on migration overhead, the differences between computing, memory and bandwidth resource requirements and the diversity in service performance between substrate nodes are temporarily masked. We fixed 5 SFCs in network. i.e., by comparing the service performance at the same threshold. As can be seen from the Fig. 4, the migration overhead gradually decreases as the η^{max} increases. This is because the larger the η^{max}, the less likely the resource demand will exceed the η^{max}, and therefore the fewer migrations need to be performed, incurring less overhead.

It is necessary to discuss the migration overhead of our proposed scheme. As shown in Fig. 5a, we can observe that the migration overhead increases consistently as the number of SFCs increases. Under the same service requests, the total migration overhead of Sa-VNFM is always lower than that of FDQ. Meanwhile, it can be seen that the overhead of Sa-VNFM lower than TD3-based, that is the former can further reduce overhead by adopting a dynamic threshold trigger mechanism compared with the latter. Fig. 5b shows the times of migration is triggered under different algorithms. Compared with TD3-based and FDQ, we notice that Sa-VNFM can reduces the times of migration respectively, which is because the Sa-VNFM can reasonably determine migration timing to reduce migration times and improve the stability of the system.

5 Conclusions

In this paper, we studied the VNF migration optimization problem in SDN/NFV-enabled STIN where VNFs are deployed in DCN. In order to avoid frequent migration, we firstly propose a dynamic threshold trigger mechanism and then formulate the problem as a stochastic optimization model based on MDP aiming at minimizing the migration overhead. For solving this problem efficiently, we resort to the DRL-Based service-aware VNF migration scheme to obtain the feasible solution. The simulation results show that the proposed scheme can effectively avoid frequent migration and reduce the migration overhead.

References

1. Yousaf, F.Z., Taleb, T.: Fine-grained resource-aware virtual network function management for 5G carrier cloud. IEEE Netw. **30**(2), 110–115 (2016)
2. Yang, P., Xiao, Y., Xiao, M., Li, S.: 6G wireless communications: vision and potential techniques. IEEE Netw. **33**(4), 70–75 (2019)
3. Sun, P., Guo, Z., Liu, S., Lan, J., Wang, J., Hu, Y.: SmartFCT: improving power-efficiency for data center networks with deep reinforcement learning. Comput. Netw. **179**, 107255 (2020)
4. Liu, J., Shi, Y., Fadlullah, Z.M., Kato, N.: Space-air-ground integrated network: a survey. IEEE Commun. Surv. Tutor. **20**(4), 2714–2741 (2018)
5. Eramo, V., Miucci, E., Ammar, M., Lavacca, F.G.: An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. IEEE/ACM Trans. Netw. **25**(4), 2008–2025 (2017)
6. Sarrigiannis, I., Ramantas, K., Kartsakli, E., Mekikis, P., Antonopoulos, A., Verikoukis, C.: Online VNF lifecycle management in an MEC-enabled 5G IoT architecture. IEEE Internet Things J. **7**(5), 4183–4194 (2020)
7. Cho, D., Taheri, J., Zomaya, A.Y., Bouvry, P.: Real-time virtual network function (VNF) migration toward low network latency in cloud environments. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), pp. 798–801 (2017)
8. Ben Jemaa, F., Pujolle, G., Pariente, M.: Analytical Models for QoS-driven VNF placement and provisioning in wireless carrier cloud. In: 19th ACM International Conference on Modeling. Analysis and Simulation of Wireless and Mobile Systems, pp. 148–155. ACM Press, Malta (2016)

9. Tang, L., He, X., Zhao, P., Zhao, G., Zhou, Y., Chen, Q.: Virtual network function migration based on dynamic resource requirements prediction. IEEE Access **7**, 112348–112362 (2019)
10. Ahmed, T., Alleg, A., Ferrus, R., Riggio, R.: On-demand network slicing using SDN/NFV-enabled satellite ground segment systems. In: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pp. 242–246 (2018)
11. Guo, Z., et al.: AggreFlow: achieving power efficiency, load balancing, and quality of service in data center networks. IEEE/ACM Trans. Netw. **29**(1), 17–33 (2021)
12. Ross, S.M.: Stochastic Processes, vol. 2. Wiley, New York (1996)
13. Li, B., Cheng, B., Liu, X., Wang, M., Yue, Y., Chen, J.: Joint resource optimization and delay-aware virtual network function migration in data center networks. IEEE Trans. Netw. Serv. Manag. 1 (2021)
14. Liu, Y., Feng, G., Chen, Z., Qin, S., Zhao, G.: Network function migration in softwarization based networks with mobile edge computing. In: ICC 2020–2020 IEEE International Conference on Communications (ICC), pp. 1–6 (2020)
15. Aroca, J.A., Chatzipapas, A., Anta, A.F., Mancuso, V.: A measurement-based characterization of the energy consumption in data center servers. IEEE J. Sel. Areas Commun. **33**(12), 2863–2877 (2015)
16. Islam, M.I., Akbar, M.M.: Heuristic algorithm of the multiple-choice multidimensional knapsack problem (MMKP) for cluster computing. In: 2009 12th International Conference on Computers and Information Technology, pp. 157–161 (2009)
17. Xia, J., Cai, Z., Xu, M.: Optimized virtual network functions migration for NFV. In: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), pp. 340–346 (2016)
18. Leivadeas, A., Kesidis, G., Falkner, M., Lambadaris, I.: A graph partitioning game theoretical approach for the VNF service chaining problem. IEEE Trans. Netw. Serv. Manag. **14**(4), 890–903 (2017)
19. Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Proceedings of the 35th International Conference on Machine Learning, vol. 80, pp. 1587–1596 (2018)
20. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: 31st International Conference on Machine Learning, ICML 2014 (2014)
21. Pujol Roig, J., Gutierrez-Estevez, D., Gündüz, D.: Management and orchestration of virtual network functions via deep reinforcement learning. IEEE J. Sel. Areas Commun. **38**, 304–317 (2020)
22. Yao, J., Chen, M.: A flexible deployment scheme for virtual network function based on reinforcement learning. In: 2020 IEEE 6th International Conference on Computer and Communications (ICCC), pp. 1505–1510 (2020)