Reliability-Aware Placement of Virtual Network Functions via Multi-Agent Deep Reinforcement Learning

Haojun Huang, Jialin Tian, Zhaoxi Li, Geyong Min, and Haozhe Wang

***Abstract*—Network Function Virtualization (NFV) often suffers from working failure in some mission-critical applications due to the reliability-agnostic deployment of Virtual Network Functions (VNFs), thus achieving suboptimal service performance. To cope with this issue, we propose a Reliability-aware VNF service Provisioning (RVFP) approach for NFV-based networks via Multi-Agent Deep Reinforcement Learning (MADRL), where each VNF is hosted in appropriate hardware with reliability guarantees. Specifically, a new MADRL-based framework of VNF placement with two alternate optimization objectives, i.e., maximizing the reliability and minimizing the failure probability of VNF instances, has been designed to explore action and exploit sample with less correlation in model learning. Furthermore, the newly-designed prioritized experience sampling, built on reward and sampling frequency, is used to exploit useful experience for model training with even faster convergence. Besides, the average and reliability-based agent allocation explorations are developed to alternately arrange the tasks of collaborative agents to execute reliability-aware placement of VNFs in distributed and parallel manners. Extensive simulation results obtained from various scenarios show that RVFP can significantly improve placement reliability and service acceptance ratios for VNF deployment compared with the state-of-the-art approaches.**

***Index Terms*—Network Function Virtualization, Deep Reinforcement Learning, Reliability, Multi-agents.**

I. Introduction

TELECOM networks have become an indispensable platform for various service provisioning, e.g., E-trade and online video sharing, in promoting economic and social well-being. Among such services, the necessary Network Functions (NFs), for example, load balancer, firewall, and intrusion detector, always are hosted in the dedicated hardware to be implemented. As a result, to launch a new network service, network operators often need to purchase and operate new hardware to accommodate the necessitated NFs, resulting in high Capital Expenditure (CAPEX) and Operational Expenses (OPEX). To cope with this issue, Network Function Virtualization (NFV) [1], [2], [3] has been proposed, as a promising initiative for network operators, by decoupling the NFs from the hardware using virtualization technologies and encapsulating a series of software-based virtual network functions (VNFs) in the form of Service Function Chains (SFCs) to provide services for users. NFV offers great benefits to design, deploy and manage network services by orchestrating software-based SFCs into the virtualized equipment to realise desirable services [4], [5].

Nevertheless, due to various instabilities and uncertainties like potential software bugs and fault-prone general hardware appliances, the service performance of NFV in reality occasionally suffers from the working failure [6], [7]. For example, the mobile networks of Vodafone in Germany experienced a three-hour outage on Nov 23, 2020 due to the failure of network elements, which leaves more than 100,000 mobile users without access to voice and data services [8]. This is unacceptable for emerging mission-critical network applications, like E-trade, autonomous driving and remote healthcare, which require ultra-reliable network services [1]. The reliability for them refers to the ability of NFV-based networks to provide stable NFV services for users in facing potential faults [9], [10]. Generally, such NFV-based services are expected to achieve nearly 99.999% (five nines) reliability of VNF instances as running in traditional dedicated hardware. How to realise reliability-aware VNF placement has become a timely burning challenge faced by telecom networks, including Data Center Networks, 5G and beyond, which have explicitly specified reliability requirements in Ultra-Reliable and Low-Latency Communication (URLLC) services.

The past years have witnessed a number of studies [11], [12], [7], [10] that focus on reliability-aware VNF deployment in networks to boost the NFV performance. Although preliminary progresses have been made, these traditional efforts still face two outstanding issues. First, it is difficult for them to explore the immense space of potential actions and network states for reliability-aware SFC instances in networks because the dynamic network conditions and the hidden network-related factors behind VNF model computation are not considered. Second, the optimization objectives of such solutions always target on reliability-related mathematical modeling or the combinations of reliability-related metrics with distinct regularity and obvious tendency of the pre-existing working performance. It is often impractical for them to obtain such observations, and thus the desirable Service Level Agreements (SLAs) cannot be achieved in most applications.

More recently, the emerging Deep Reinforcement Learning (DRL) has achieved remarkable performance gains in diversified applications, including autonomous driving, traffic engineering, and network slicing [13], [14]. Being an alternative promising paradigm, DRL has been successfully applied

Z. Li is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, 430074, China. Email: li@hust.edu.cn.

H. Huang, J. Tian, G. Min and H. Wang are with the Department of Computer Science, University of Exeter, Exeter, EX4 4QF, UK. Email: {h.huang2, jtian, g.min, h.wang3}@exeter.ac.uk.

in SFC placement to resolve many critical issues, including resource allocation, routing selection, delay optimization, and traffic scheduling [1], [15], [13], [16]. With the powerful learning capacity, DRL has distilled the close relationships among network states, taken actions, and their rewards known as the desired goals of VNF placement in networks. DRL can not only efficiently discover and characterize the features and the hidden rules behind model computation, but also take into account the important network-related factors, parameters and metrics in VNF placement in an intelligent manner.

Unfortunately, directly applying the current DRL to the reliability-aware placement of VNFs poses two critical challenges, which need to be addressed urgently [17]. First, there are no agents installed with alternative optimization objectives to execute reliability-aware VNF placement. The current DRL agents mainly devote to one objective, easily leading to the inertial action explorations and potential strong-relevant sample exploitation in VNF model computation, which will hinder the convergence of reliable-model training and decrease its learning accuracies. Therefore, it is requisite to introduce some alternative optimization objectives for agents to further adjust their action explorations and sample exploitation for reliability-aware placement of VNFs. Second, it is inefficient for a single agent, which is often equipped with limited resources, to learn a global model for VNF placement in multi-region networks, because the region interaction cannot be well captured for model learning. Essentially, the VNF placement in multi-region networks has involved interaction between collaborative regions, where emergent behaviors and complexity arise from regions co-evolving together. However, the conventional single-agent DRL schemes like Q-Learning are poorly suited to multi-region environments [1], [18]. These issues would be alleviated by introducing Multi-Agent DRL (MADRL) installed with alternative objectives, instead of the traditional single-agent DRL, due to its more powerful capability of hidden feature extraction and action explorations [19], [20], [18], and parallel VNF instancing in networks.

These observations motivate us to develop a Reliability-aware VNF service Provisioning (RVFP) approach, built on multiple-objective MADRL strategies, to execute VNF orchestration in NFV-based networks in distributed, parallel and collaborative manners. The core philosophy of RVFP is to host each VNF in appropriate hardware with reliability guarantees, which is achieved by training reliability-aware models of VNF placement in cross-region networks via MADRL with two alternate complementary goals in two perspectives, i.e., maximizing the reliability and minimizing the failure probability of VNFs, and parallelly instancing VNFs with collaborative multi-agents in an efficient manner. To further accelerate the convergence of model training and enhance learning accuracy, the newly-designed training sampling and action exploration included in the improved Multi-Agent Deep Deterministic Policy Gradient (MADDPG), which can work well in discrete action space by introducing Gumbel Softmax Trick, are developed. The main contributions of this paper are listed as follows.

- The reliability-aware MADRL-based framework of VNF placement is proposed to boost the NFV performance with reliability guarantees in a distributed and cooperative manner. The placement of given SFCs in cross-region network has been divided into a sequence of MADRL-based optimization problems. To the best of our knowledge, this is the first effort on reliability-aware VNF placement using MADRL built on the improved MADDPG, which can accelerate the learning efficiency of placement model, while without causing additional overhead.
- Two alternate reliability-related optimization goals are designed for reliability-aware placement of VNFs in each region based on the dynamically available resources and reliability requirements of VNFs over time, thus eliminating the strong correlation of sample exploitation. Furthermore, the prioritized experience sampling, built on reward and sampling frequency, is proposed to exploit useful experience for model training with quicker convergence. Besides, both the average and reliability-based agent allocation explorations have been developed to alternately arrange the tasks of collaborative agents to execute reliability-aware VNF placement in distributed manners.
- Extensive experimental scenarios, with a diverse number of SFCs, are conducted on given network topologies to evaluate the performance of the proposed RVFP. The results show that RVFP can significantly improve placement reliability and service acceptance ratio to deploy VNFs compared with the state-of-the-art approaches.

The rest of this paper is organized as follows. Section II presents the system model, followed by the problem statement in Section III. Section IV introduces in detail the proposed RVFP for VNF placement in networks via MADRL. Then, specific implementations, results and performance comparisons are represented in Section V. Section VI provides an overview of the related work. Finally, we conclude the paper in Section VII.

II. System Model

This section mainly elaborates the system model, including substrate networks, SFC Requests (SFCRs) and SFC deployment model, to facilitate the understanding of the proposed RVFP. The primary parameters and notations are listed in Table I.

A. Substrate Networks

Consider the scenarios where virtual machines are hosted on substrate hardware with different resources and reliabilities can provide services for a set of VNFs, in the form of SFCs. A typical NFV-enabled substrate network is comprised of massive physical nodes and links between them, and denoted as an undirected graph $G^s{=}(V^s{,}L^s)$ with $\mathbb{N}$ regions $d_1, d_2, \ldots, d_{\mathbb{N}}$, where $v_i{\in}V^s$ and $l_{ii'}{\in}L^s$ are a substrate node and the reliable link connecting nodes v_i and $v_{i'}$, respectively. Correspondingly, the resources that contribute to provide SFC services are also divided into node and link resources. The available resources of node v_i include CPU resource c_{v_i} and bandwidth resource b_{v_i}. Similarly, the available link resources of $l_{ii'}$ refer to bandwidth capacity $b_{l_{ii'}}$. It is worth mentioning that b_{v_i}

is further determined by the maximum value of $b_{l_{ii'}}$ of $l_{ii'}$, which is connected with v_i. There are one center orchestrator and $\mathbb{N}$ local orchestrators in regions $d_1, d_2, \ldots, d_{\mathbb{N}}$, who are responsible for the training and execution stages. The hop and delay of link $l_{ii'}$ are denoted as $hop_{l_{ii'}}$ and $t_{l_{ii'}}$, which depends on the shortest path between nodes v_i and $v_{i'}$. Similar to [16], [17], it is considered that the ordinary nodes like routers and switches used for information exchange in model training and execution in NFV-based networks can work well. The reliability of v_i is denoted as $r_{v_i}(0<r_{v_i}<1)$, which is decided by the probability of the normal node operations to host VNFs, i.e., $r_{v_i}=\Upsilon_{v_i}/(\Upsilon_{v_i}+\Lambda_{v_i})$. Here, Υ_{v_i} refers to the average time between failures of node v_i, i.e., the normal working time to host VNFs, while Λ_{v_i} is the abnormal working time of v_i after failures. Both of them can be calculated by statistical analysis based on the node log files and system configuration data.

B. Service Function Chain Requests

Suppose that the service requests stemmed from users are represented as a set $\mathcal{S}^v$. A single SFC $s_k \in \mathcal{S}^v$ is donated as a directed graph $G_k^v=(I_k^v, E_k^v, V_k^v, L_k^v)$, where I_k^v and E_k^v represent the physical ingress and egress of service s_k, respectively, V_k^v denotes the set of required VNFs, and $l_{k|uu'} \in L_k^v$ enforces the oriented flow routing from VNF $v_{k|u}$ to $v_{k|u'}$. Especially, $l_{k|I}$ describes the link between I_k^v and VNF $v_{k|1}$, and $l_{k|E}$ is the link between the last VNF and E_k^v. Each VNF, denoted by $v_{k|u}$, $1 \le u \le |V_k^v|$, may provide a different type of service, such as firewall and load balancer, and thus the type of $v_{k|u}$ is indicated as $t_{v_{k|u}}=f_j,\ f_j \in \mathscr{F}=\{f_1, f_2, ..., f_{|\mathscr{F}|}\}$. Besides, each VNF is associated with resource demands, i.e., CPU and bandwidth requests as $c_{v_{k|u}}^{req}$ and $b_{v_{k|u}}^{req}$. Similarly, each virtual link $l_{k|uu'}$ is related to bandwidth resource request $b_{l_{k|uu'}}^{req}$. What's more, reliability requirements are essential indicators that mainly refer to the reliability request of VNF as $r_{v_{k|u}}^{req}$.

It is considered that NFV-based services, for example, cloud-enabled services over the real-world networks, are constantly arriving and departing the network. The continuous time is divided into equal discrete slots as $\mathscr{T}=\{1, 2, ..., t\}$. Thus, the aforementioned two processes are considered at the beginning and end of the slot, respectively. Suppose that the arrival process of different SFCs is independent and identically distributed. Similar to [15], the arriving of κ SFCs at each time slot follows the Poisson distribution with λ, which equals to the average number of SFCs arriving in a unit slot. The departure probability of s_k at the end of a slot is d_k, which is a constant. As a result, each SFC may be active in the system for multiple slots. Thus, SFCs that require to be served at a certain slot include both the remaining SFCs from the previous slot and the SFCs that have just arrived at the beginning of that slot.

C. SFC Deployment Model

To clearly describe the deployment of SFCR G_k^v into the substrate network G^s, we build a directed deployment graph $G^d=(V^d, L^d)$, where V^d and L^d respectively stand for the state sets of deployment of VNFs and virtual links between them. First of all, $v_i^{k|u}(t) \in V^d$ is introduced and defined as a binary variable indicating whether VNF $v_{k|u}$ is deployed on node v_i at time slot t as

$$v_i^{k|u}(t) = \begin{cases} 1 & \text{if } v_{k|u} \text{ is deployed on } v_i \text{ at } t, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Similarly, $l_{ii'}^{k|uu'}(t)$ denotes the state whether a request flow processed by v_i shall be passed to the adjacent VNF $v_{k|u'}$ on node $v_{i'}$ at time slot t, following the VNF sequence defined in G_k^v. It is also a binary variable that can be defined as

$$l_{ii'}^{k|uu'}(t) = \begin{cases} 1 & \text{if } l_{k|uu'} \text{ is deployed on } l_{ii'} \text{ at } t, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Based on this, the deployment of SFCs is equivalent to decide the values of $v_i^{k|u}$ for each $v_{k|u}$ and v_i as well as flow value $l_{ii'}^{k|uu'}$ according to the requests for different SFCs. Furthermore, in order to describe the deployment states on nodes in more detail, $v_i^{f_j}$ is defined as another binary variable to present whether the VNF Instance (VNFI) of type f_j has been on server v_i at the time slot t as

$$v_i^{f_j}(t) = \begin{cases} 1 & \text{if VNFI of type } f_j \text{ is on } v_i \text{ at } t, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

which reflects the deployment of the existing VNFs on physical nodes, to some extent.

TABLE I
THE IMPORTANT PARAMETERS AND NOTATIONS

Notations	Description
V^s/L^s	Set of substrate nodes/links
c_{v_i}/b_{v_i}	Computing/bandwidth resource of $v_i \in V^s$
$b_{l_{ii'}}$	Bandwidth resource of link $l_{ii'} \in L^s$
r_{v_i}	Reliability of node v_i
$hop_{l_{ii'}}/t_{l_{ii'}}$	Hop/delay of shortest path $l_{ii'}$, $0<r_{v_i}<1$
$G_k^v=(I_k^v, E_k^v, V_k^v, L_k^v)$	Directed graph of SFC $s_k \in \mathcal{S}^v$
I_k^v/E_k^v	Ingress/egress of SFC $s_k \in \mathcal{S}^v$
V_k^v/L_k^v	Set of required VNFs/virtual links
$c_{v_{k\|u}}^{req}/b_{v_{k\|u}}^{req}$	Computing/bandwidth request of $v_{k\|u} \in V_k^v$
$b_{l_{k\|uu'}}^{req}$	Bandwidth required by $l_{k\|uu'} \in L_k^v$
$r_{v_{k\|u}}/t_{v_{k\|u}}$	Reliability request/type of VNF $v_{k\|u}$
$f(\|S^v\|)$	Probability function of arrival of $\|S^v\|$ SFCs
$v_i^{k\|u}(t)/l_{ii'}^{k\|uu'}(t)$	Binary variable indicating if $v_{k\|u}/l_{k\|uu'}$ is deployed on $v_i/l_{ii'}$ at time slot t
$v_i^{f_j}(t)$	Binary variable indicating if VNFI of type f_j has been on v_i at time slot t
$C_{set}(t)/C_{op}(t)/C_{rt}(t)$	VNF set/operation/communication resource cost for all SFCs at time slot t
$R_{v_k}(t)$	Combined reliability of all VNFs of SFC s_k at time slot t

III. PROBLEM STATEMENT

This section mainly illustrates the detailed problem statement. First, the important descriptions on reliability-aware deployment of SFCs are provided, followed by the presentations on the reliability of VNFs and resource cost. Then, the problem formulation with two reliability-related optimization goals is presented.

A. Problem Description

Given an SFC G_k^v=(I_k^v,E_k^v,V_k^v,L_k^v), there are multiple candidate nodes and links for its VNFs and virtual links to be implemented in G^s=(V^s,L^s). How to reliably embed V_k^v and L_k^v into V^s and L^s mainly depends on the consumed-resources and achieved-reliability of different deployment manners when facing the same SFCR. In addition, the arrival of SFCs is dynamic, and thus the deployment process changes dynamically. In this case, the probability distribution $P\{s(t+1)|s(t),...,s(1),a(t),...,a(1)\}$ of SFC deployment follows the finite Markov property, defined as

$$P\{s(t+1)|s(t),...,a(t),...\} = P\{s(t+1)|s(t),a(t)\}, \quad (4)$$

where $s(t+1)$ and $s(t)$ represent two states of adjacent time slots t and $t+1$, and $a(t)$ is an action in state $s(t)$.

There are multiple homogeneous agents in networks devoted to reliability-aware VNF service provisioning. Due to the close cooperations based solely on shared observations among them, the SFC deployment can be modelled as a Partially Observable Markov Decision Process (POMDP) [19], built on Markov games [18], to embed VNFs and their connected links into G^s=(V^s,L^s). The goal of POMDP is to find the policy, defined as $\pi[a(t)|s(t)]$, which specifies action $a(t)$ taken in state $s(t)$. The optimal policy $\pi^*[s(t)]$ can be expressed as

$$\pi^*[s(t)] = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t R[s(t),a(t),s(t+1)]\right], \quad (5)$$

which maximizes the expected discounted sum of rewards for taking action according to it in state $s(t)$.

B. Reliability of VNFs

Given SFC s_k instanced in G^s, let $R_{v_k}(t)$ be the combined reliability of its all VNFs at time slot t, which fails to take the reliability into account in their instances. Then, we have

$$R_{v_k}(t) = \prod_u r_{v_{k|u}}(t) = \prod_u \left[\sum_i v_i^{k|u}(t) r_{v_i}\right]. \quad (6)$$

Because of various instabilities and uncertainties, each VNF of SFC s_k at time slot t will have different levels of reliability. To improve the reliability of all VNFs of SFC s_k, the hot backups, as shown in Fig. 1, are adopted, which refers to simultaneously deploying VNF $v_{k|u}$ on the main node v_i and additional backup nodes $\tilde{v}_i$, along with the resource allocations for them. In this way, $v_{k|u}$ can also be served by the backup nodes in the event that main node v_i fails. With hot backups, the service for VNF $v_{k|u}$ can be provided as long as any of the main and backup nodes is available, which improves the combined reliability of VNFs, $R_{v_k}(t)$, as

$$R_{v_k}(t) = \prod_u \begin{cases} \sum_i v_i^{k|u}(t) r_{v_i}, & \sum_i \tilde{v}_i^{k|u}(t) = 0, \\ \tilde{r}_{v_{k|u}}(t), & \sum_i \tilde{v}_i^{k|u}(t) = 1, \end{cases} \quad (7)$$

where

$$\tilde{r}_{v_{k|u}}(t) = 1 - [1 - \sum_i v_i^{k|u}(t) r_{v_i}][1 - \sum_i \tilde{v}_i^{k|u}(t) r_{v_i}]. \quad (8)$$

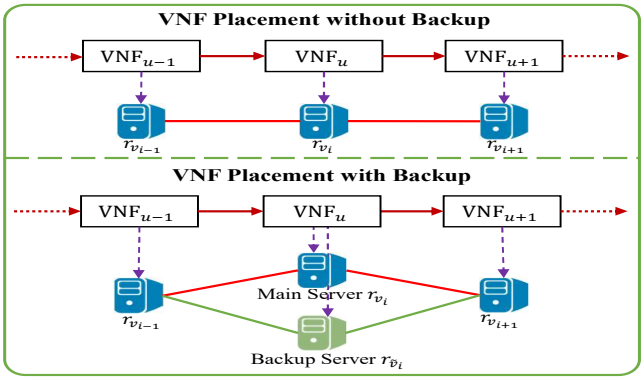


Fig. 1. The backup of VNF placement in networks to improve its reliability.

The parameter $\tilde{v}_i^{k|u}(t)$ is equal to 1 if node $\tilde{v}_i$ is chosen as a backup node for $v_{k|u}$ at time slot t, and $r_{\tilde{v}_i}$ is the reliability of the backup node, which is independent of the main node's reliability. The previous studies show that $R_{v_k}(t)$ with backup nodes is always bigger than $R_{v_k}(t)$, clearly meaning that hot backups can improve the reliability of VNF deployments.

C. Resource Cost

The deployment of SFCs can be considered as a process of allocating resources of the substrate networks to each VNF and steering the traffic forwarding. On the premise of providing required services, it is important for network service providers to reduce the cost of resource consumption, which consists of two major components, the cost incurred at nodes and that of routing between nodes. In addition, the resource cost on nodes at time slot t further includes VNF set cost $C_{set}(t)$ and operation cost $C_{op}(t)$. For VNF $v_{k|u}$ of type f_j, the set cost occurs when $v_{k|u}$ is deployed or backed up on v_i and there is no VNFI of type f_j on v_i at the time slot t. Therefore, $C_{set}(t)$ is denoted with the help of $v_i^{k|u}(t)$, $\tilde{v}_i^{k|u}(t)$, $v_i^{f_j}(t)$=$v_i^{t_{v_{k|u}}}(t)$ and $\tilde{v}_i^{f_j}(t)$=$\tilde{v}_i^{t_{v_{k|u}}}(t)$ as

$$\begin{aligned} C_{set}(t) = &\sum_k \sum_u \sum_i (1 - v_i^{t_{v_{k|u}}}(t)) v_i^{k|u}(t) s_{t_{v_{k|u}}} \\ &+ \sum_k \sum_u \sum_i (1 - \tilde{v}_i^{t_{v_{k|u}}}(t)) \tilde{v}_i^{k|u}(t) s_{t_{v_{k|u}}}, \end{aligned} \quad (9)$$

where $s_{t_{v_{k|u}}}$ is the set cost of VNF with type $t_{v_{k|u}}$ and only associated with the type of VNFs. As for the VNF operation cost $C_{op}(t)$, it occurs when $v_{k|u}$ is deployed or backed up on v_i at time slot t, which can be calculated as

$$C_{op}(t) = \sum_k \sum_u \sum_i [v_i^{k|u}(t) + \tilde{v}_i^{k|u}(t)] c_{v_{k|u}}^{req}. \quad (10)$$

Meanwhile, the communication cost for the routing between nodes where adjacent VNFs are deployed or backed up at the time slot t, e.g. nodes v_i and $v_{i'}$ for VNFs $v_{k|u}$ and $v_{k|u'}$, is indicated as $C_{rt}(t)$, which can be expressed as

$$C_{rt}(t) = \sum_k \sum_u \sum_i \sum_{i'} l_{ii'}^{k|uu'}(t) hop_{l_{ii'}} b_{l_{k|uu'}}^{req}. \quad (11)$$

D. Problem Formulation

The goal of our RVFP is to maximize the reliability of VNFs while minimizing the resource costs consumed in their instances, under the following important constraints. The first one is designed to guarantee the successful deployment of each VNF with at least one main node as well as a possible backup node, and it is denoted as

$$\begin{cases} \sum\limits_{i} v_i^{k|u}(t) = 1, \forall v_{k|u} \in V_k^v, \\ 0 \leq \sum\limits_{i} \tilde{v}_i^{k|u}(t) \leq 1, \forall v_{k|u} \in V_k^v, \\ v_i^{k|u}(t) + \tilde{v}_i^{k|u}(t) \leq 1, \forall v_{k|u} \in V_k^v, \forall v_i = \tilde{v}_i \in V^s, \end{cases} \tag{12}$$

where v_i and $\tilde{v}_i$ are the main and backup nodes for $v_{k|u}$, respectively. Since node v_i cannot be the main and backup node for the same VNF $v_{k|u}$, i.e., $v_i^{k|u}(t)$ and $\tilde{v}_i^{k|u}(t)$ cannot both be 1, $v_i^{k|u}(t) = 1$ will be used to indicate that node v_i is the main or backup node of $v_{k|u}$.

The second constraint is introduced to satisfy the network-flows conservation as

$$\sum_{k}\sum_{u} v_i^{k|u}(t) v_j^{k|u+1}(t) - \sum_{k}\sum_{u} v_j^{k|u}(t) v_i^{k|u+1}(t) = 0, \\ \forall v_i, v_j \in V^s, v_i, v_j \neq I_k^v, E_k^v, \forall k \in [1, K], \forall u \in [1, |V_k^v|]. \tag{13}$$

The static flow of all nodes except the ingress and the egress is 0. That's to say, the sum of forwarding flow and reverse flow between any two nodes is 0 as Eq. (13).

The third one is to avoid that the consumption of computing resources on each node v_i exceeds the limit of node resource capacity as

$$\sum_{k}\sum_{u} v_i^{k|u}(t) c_{v_{k|u}}^{req} \leq c_{v_i}, \quad \forall v_i \in V^s. \tag{14}$$

Besides, the handling capacity b_{v_i} of v_i must be larger than the bandwidth of all SFCs passing through it as

$$\sum_{k}\sum_{u}\sum_{i'} l_{ii'}^{k|uu'}(t) b_{l_{k|uu'}}^{req} \leq b_{v_i}, \quad \forall v_i \in V^s. \tag{15}$$

Furthermore, the routing determined by VNF placement also requires to meet some constraints of link bandwidth as

$$\sum_{k}\sum_{u} l_{ii'}^{k|uu'}(t) b_{l_{k|uu'}}^{req} \leq b_{l_{ii'}}, \quad \forall l_{ii'} \in L^s. \tag{16}$$

The last constraint is defined to guarantee the reliability requirements of incoming services, satisfying

$$R_{v_k}(t) \geq r_{v_k}^{req}, \quad \forall s_k \in \mathcal{S}^v. \tag{17}$$

Built upon the above constraints, the optimization problem of reliability-aware SFC deployment can be formulated as

Reli-Max:

$$\max \quad \frac{\bar{R}_v^\pi(s)^*}{\bar{C}_{set}^\pi(s)^* + \bar{C}_{op}^\pi(s)^* + \bar{C}_{rt}^\pi(s)^*}, \tag{18}$$
$$s.t. \quad \text{Eqs. (12)-(17)},$$

where $\bar{R}_v^\pi(s)^*$, $\bar{C}_{set}^\pi(s)^*$, $\bar{C}_{op}^\pi(s)^*$ and $\bar{C}_{rt}^\pi(s)^*$ are the expected discounted sum of normalized node reliability $R_{v_k}(t)^*$, normalized VNF set cost $C_{set}(t)^*$, normalized VNF operation cost $C_{op}(t)^*$ and normalized communication cost $C_{rt}(t)^*$ for all coming SFCs, respectively. Taking $\bar{R}_v^\pi(s)^*$ as an example, it is defined as

$$\bar{R}_v^\pi(s)^* = \mathbb{E}_\pi \left[\sum_t \gamma^t \sum_k R_{v_k}(t)^* | s(0) = s \right], \tag{19}$$

where $R_{v_k}(t)^*$ is the normalization of $R_{v_k}(t)$ as

$$R_{v_k}(t)^* = \frac{R_{v_k}(t) - R_{v_k}^{min}}{R_{v_k}^{max} - R_{v_k}^{min}}. \tag{20}$$

Here, $R_{v_k}^{max}$ and $R_{v_k}^{min}$ are the maximum and minimum values of reliability that $R_{v_k}(t)$ can achieve, respectively, while $\bar{C}_{set}^\pi(s)^*$, $\bar{C}_{op}^\pi(s)^*$ and $\bar{C}_{rt}^\pi(s)^*$ are defined in a similar way. The optimization objective is to maximize the reliability at unit cost for incoming SFCs with different demands, so as to improve reliability as much as possible while with less resource consumption. The considered constraints are set on the allocation of main and backup nodes, traffic routing, resource capacity of nodes and links, and reliability requirement of SFCs.

Given G^s, the reliability of SFCs mainly depends on the probability that nodes run without failure, which is relevant to the failure probability of nodes while with different action explorations and sample exploitation. Therefore, the reliability-aware SFC deployment can be formulated as another optimization problem aiming to minimize the failure probability of nodes as

Fail-Min:

$$\min \quad \frac{\bar{F}_v^\pi(s)^*}{\bar{C}_{set}^\pi(s)^* + \bar{C}_{op}^\pi(s)^* + \bar{C}_{rt}^\pi(s)^*}, \tag{21}$$
$$s.t. \quad \text{Eqs. (12)-(17)},$$

where $\bar{F}_v^\pi(s)^*$ is the expected discounted sum of normalized probability of node failure for all coming SFCs, which thus can be denoted as

$$\bar{F}_v^\pi(s)^* = \mathbb{E}_\pi \left[\sum_t \gamma^t \left(|S^v| - \sum_k R_{v_k}(t) \right)^* | s \right]. \tag{22}$$

Obviously, the considered constraints of the **Fail-Min** problem are the same as the **Reli-Max** problem. However, both of them are addressed via MADRL with different action explorations and sample exploitation.

IV. RVFP Design

This section presents in detail the proposed RVFP for reliability-aware placement of VNFs. First, the essential POMDP-oriented 5-tuple of MADRL and MADRL-based training framework are introduced, and then the local training executed at each agent is illustrated.

A. POMDP-Oriented 5-tuple Design

The essential elements required by MADRL for reliability-aware placement of VNFs can be represented as a POMDP-oriented 5-tuple, denoted by $<\mathbb{N}, \mathbb{S}, \mathbb{A}, \mathbb{R}, \gamma>$, which includes the multiple agents, state space, action space, reward, and discount factor, respectively. Both $\mathbb{N}$ and $\mathbb{S}$ represent the number of agents and the state space, which are decided by the substrate network and arriving SFCs. $\mathbb{A}$ stands for the action space which makes states change. $\mathbb{R}$ donates the reward space received after taking actions in corresponding states, and γ is the discount factor satisfying $0<\gamma\leq 1$.

Upon a state $s(t)\in\mathbb{S}$, one agent could observe a new state $s(t+1)\in\mathbb{S}$ and obtain the corresponding reward $r(t)$ after taking action $a(t)\in\mathbb{A}$, where $r(t)$ can be used to evaluate the effectiveness of action and the total reward of it is associated with the optimization goals of RVFP as Eqs. (18) and (21). To train reliability-aware models of VNF placement, the essential state, action and reward associated with the operations of RVFP are defined as follows:

*1) **State Space**:* The state space, $\mathbb{S}$, is the set of all possible states, $s(0), ..., s(t)$, that further comprises configurations of the network and demands of SFCRs, given by

$$\mathbb{S} = \bigcup_{t=0} \{s(t)\}. \tag{23}$$

The state of MADRL at the time slot t, $s(t)$, consists of:

- Available computing and bandwidth resources of nodes and links in the substrate networks at the beginning of each time slot, $\{c_{v_i}(t), b_{v_i}(t), b_{l_{ii'}}(t)\}$.
- The state of VNFI, which is equal to the types of VNFs that have been instantiated in nodes at the beginning of each slot, $\{v_i^{f_j}(t)\}$.
- Reliability that nodes can offer in substrate networks, $\{r_{v_i}\}$, which stays constant over a long period of time.
- The requirements of incoming SFCs at the beginning of each slot, containing ingress, egress, VNFs contained, required resources and reliability, $G_k^v(t)=(I_k^v, E_k^v, V_k^v, L_k^v)$.

Therefore, the state at the time slot t can be denoted as

$$s(t) = \bigcup_{v_i\in V^s} \bigcup_{l_{ii'}\in L^s} \bigcup_{f_j\in\mathcal{F}} \bigcup_{s_k\in\mathcal{S}^v} \left\{c_{v_i}(t), b_{v_i}(t), b_{l_{ii'}}(t), v_i^{f_j}(t), r_{v_i}, G_k^v(t)\right\}, \tag{24}$$

which is observed by agents and dynamically changes with time after each agent taking an action.

*2) **Action Space**:* The action space of MADRL consists of a set of subspaces of actions belonging to each agent as

$$\mathbb{A} = \{\mathbb{A}_1, ..., \mathbb{A}_n, ..., \mathbb{A}_{\mathbb{N}}\}, \tag{25}$$

where $\mathbb{A}_n$ is the set of possible actions taken by agent $\hat{a}_n$, and all actions, $\bigcup_{n\in[1,\mathbb{N}]}[a_n(t)\in\mathbb{A}_n]$, taken place at the time slot t will change the current state $s(t)$ together. Essentially, $\mathbb{A}_n$ mainly refers to possible actions for VNFs to be deployed, which further includes the deployment of each VNF to a main node and a possible backup node, and actions for virtual links between VNFs to be deployed on substrate links. Therefore, we define the sub action set, $\mathbb{A}_n$, as a combination of three action spaces as

$$\begin{cases} \mathbb{A}_n^v \cup \mathbb{A}_n^{\tilde{v}} \cup \mathbb{A}_n^l = \mathbb{A}_n, \\ \mathbb{A}_n^v \cap \mathbb{A}_n^{\tilde{v}} \cap \mathbb{A}_n^l = \emptyset, \end{cases} \tag{26}$$

where $\mathbb{A}_n^v$, $\mathbb{A}_n^{\tilde{v}}$ and $\mathbb{A}_n^l$ represent the action space that VNFs are placed on the main node and backup node, as well as the placement of virtual links, respectively. The action $a_n(t)$, accordingly containing $a_n^v(t)\in\mathbb{A}_n^v$, $a_n^{\tilde{v}}(t)\in\mathbb{A}_n^{\tilde{v}}$ and $a_n^l(t)\in\mathbb{A}_n^l$, is further decided by the introduced decision variables, $v_i^{k|u}(t)$, $\tilde{v}_i^{k|u}(t)$ and $l_{ii'}^{k|uu'}(t)$, as

$$a_n^v(t) = \bigcup_{v_i\in V^s} \bigcup_{v_{k|u}\in V_k^v} \left\{v_i^{k|u}(t)\right\}, \tag{27}$$

$$a_n^{\tilde{v}}(t) = \bigcup_{\tilde{v}_i\in V^s} \bigcup_{v_{k|u}\in V_k^v} \left\{\tilde{v}_i^{k|u}(t)\right\}, \tag{28}$$

$$a_n^l(t) = \bigcup_{l_{ii}\in L^s} \bigcup_{l_{k|uu'}\in L_k^v} \left\{l_{ii'}^{k|uu'}(t)\right\}. \tag{29}$$

Taking $a_n^v(t)$ as an example, $\left\{v_i^{k|u}(t)\right\}$ is the set that only contains $v_i^{k|u}(t)$, while $\bigcup_{v_i\in V^s}$ and $\bigcup_{v_{k|u}\in V_k^v}$ denote the union set of it over all v_i and $v_{k|u}$, respectively. $a_n^{\tilde{v}}(t)$ and $a_n^l(t)$ are defined in a similar way.

*3) **Reward Space**:* Corresponding to the action space, reward $r(t)$ at the time slot t, decided by action $a(t)$ and state $s(t)$, is the collection of rewards in all agents as

$$r(t) = \{r_1(t), ..., r_n(t), ..., r_{\mathbb{N}}(t)\}. \tag{30}$$

Each reward $r_n(t)$, independently calculated by agent $\hat{a}_n$ after taking action $a_n(t)$, is different from each other due to the diverse optimization goal that agent $\hat{a}_n$ adopts, either maximizing the reliability or minimizing the probability of node failure at unit cost as defined in Eqs. (18) and (21), respectively. Therefore, there are two optional rewards for each agent, based on **Reli-Max** and **Fail-Min** problems, respectively, which can be precisely modelled as

$$R^r(t) = \frac{R_{v_k}(t)^*}{C_{set}(t)^* + C_{op}(t)^* + C_{rt}(t)^*}, \tag{31}$$

$$R^f(t) = -\frac{(1-R_{v_k}(t))^*}{C_{set}(t)^* + C_{op}(t)^* + C_{rt}(t)^*}, \tag{32}$$

where $R^r(t)$ is the reward space of **Reli-Max**, determined by the achieved reliability after deployment, while $R^f(t)$ is that of **Fail-Min**, depending on the negative of corresponding failure probability.

B. MADRL-based Framework of VNF Placement

The framework of RVFP is shown in Fig. 2, which includes training and execution phases to perform MADRL-based reliability-aware placement of VNFs in cross-region networks via centralized training and decentralized execution. There are $2\mathbb{N}$ agents, who parallelly perform model training of VNF placement with two alternate goals, as well as one center orchestrator and $\mathbb{N}$ local orchestrators, which are responsible

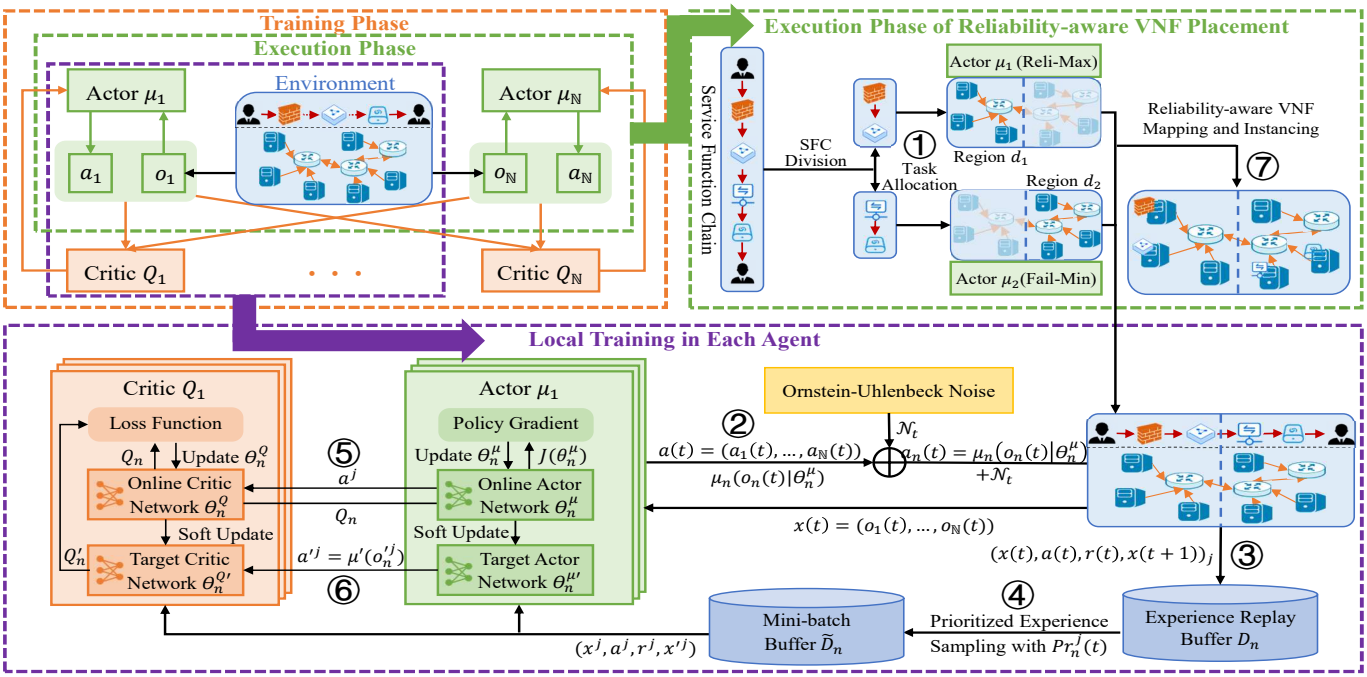


Fig. 2. The MADRL-based framework of reliability-aware VNF deployment, where two agents assigned into one region train reliability-aware models of VNF placement with the goals of **Fail-Min** and **Reli-Max**, respectively.

for the training and execution stages, respectively. With the learned models, the instance of VNFs in each region is executed by one local orchestrator in a decentralized manner and cooperates with others via MADRL, enabling to fulfill the reliability requirements of VNFs.

The whole network is divided into $\mathbb{N}$ regions. There are two agents assigned to each region to alternately learn placement model $\theta_n(t)$ ($n \in [1, \mathbb{N}]$) with the goal of either **Fail-Min** or **Reli-Max** under the constraints defined in Eqs. (12)-(17). Each agent will establish its own observation space $\mathbb{O}_n$ while facing the public state space $\mathbb{S}$. On this basis, there are $\mathbb{N}$ experience replay buffers that store states, actions, rewards, and next states of all agents as $\{x(t), x(t+1), a_1(t), ..., a_{\mathbb{N}}(t), r_1(t), ..., r_{\mathbb{N}}(t)\}$, where $x(t) = \{o_1(t), ..., o_{\mathbb{N}}(t)\}$ are states observed by agents.

Each agent included in RVFP is built on the improved MADDPG, which is specially competent to the ever-increasing discrete action explorations of VNF placement by introducing Gumbel Softmax Trick [21], and composed of two parts, i.e., the actor network (online actor network θ_n^μ and target actor network $\theta_n^{\mu'}$) as well as the critic network (online critic network θ_n^Q and target critic network $\theta_n^{Q'}$). The actor network θ_n^μ is in charge of the definition of parameterized policy μ and exploration of actions $a_n(t) = \mu(x(t)|\theta_n^\mu)$ based on the network states $x(t)$, while the critic network θ_n^Q is responsible for the evaluation of current action and simulation of action value function as $Q_n(x(t), a_1(t), ..., a_{\mathbb{N}}(t))$ which requires policy information of other agents and created by each agent based on their own reward $R^r(t)$ or $R^f(t)$ defined in Eqs. (31) and (32).

The training and execution phrases of RVFP shown in Fig. 2 mainly consist of seven steps: ① task allocation, ② action explorations, ③ experience replay buffer establishment, ④ prioritized experience sampling, ⑤ online critic/policy network update, ⑥ target network soft update, and ⑦ VNF mapping and instancing. The training phase includes steps ②-⑥, while the execution phase refers to steps ① and ⑦. In step ①, the SFCs to be deployed are split and allocated to diverse regions where there are two optional agents with different goals. Step ② explores actions concerned with VNF placement on main and possible backup servers under the constraints defined in Eqs. (12)-(17), and takes effects on the environment. Step ③ is responsible for the collection of experience which is generated by the interaction between agent and environment. On this basis, step ④ samples a mini-batch of experience in a prioritized sampling manner, which are further used for the local training of each agent in steps ⑤-⑥. Online critic/policy network of each agent is updated independently, based on the improved MADDPG and Q-values, at each time slot in step ⑤, which is followed by the soft update of target critic/policy network in step ⑥. Finally, VNFs are mapped into substrate networks and instanced in it under the guidance of step ⑦.

The entire SFCs will be divided and assigned to diverse regions $d_1, d_2, \ldots, d_{\mathbb{N}}$, determined by the amount of VNFs and available computing resources of each region. The priority order of each region in the SFC division is denoted as

$$\xi(d_n) = rank_d(\sum_i c_{v_i}), \tag{33}$$

where $\sum_i c_{v_i}$ is the sum of available computing resources of servers in d_n, and $rank_d(\cdot)$ is a function that sorts $\sum_i c_{v_i}$ in descending order and gets its serial number. The higher that $\sum_i c_{v_i}$ is, the smaller $\xi(d_n)$ will be, and more VNFs will be allocated to d_n. Thus, the number of VNFs assigned to d_n is

$$num_{d_n} = \lfloor clip(\frac{max[(K \bmod \mathbb{N}), N_{\xi(d_n)}] - N_{\xi(d_n)}}{(K \bmod \mathbb{N}) - N_{\xi(d_n)}}, 1, 0) + \frac{K}{\mathbb{N}} \rfloor, \tag{34}$$

where K, $\mathbb{N}$ and $N_{\xi(d_n)}$ are the amount of VNFs, all regions and regions up to the $\xi(d_n)$-th field, and the function $clip(\cdot,0,1)$ will be equal to 1 if $(K \bmod \mathbb{N}) \geq N_{\xi(d_n)}$. In this way, an SFC is sequentially sliced into several sub-SFCs and allocated to all regions $d_1, d_2, \ldots, d_{\mathbb{N}}$, where the number of VNFs is num_{d_n} for d_n.

As for the agent allocation, there are two kinds of agents in each network region, e_r and e_f, which share their states with each other and alternately guide the VNF placement following **Reli-Max** and **Fail-Min** defined in Eqs. (18) and (21), respectively. To achieve this goal and determine which agent is used for the current region, two agent selection solutions, i.e., average and reliability-based agent allocation, have been designed. The former will arrange an agent, with the goal of either **Reli-Max** or **Fail-Min**, to train a model of VNF deployment in a region, while the latter will switch the agent of each region, depending on its average reliability. Initially, each region d_n is assigned a sub-SFC with num_{d_n} VNFs, where the average reliability demand of these VNFs can be expressed as $\overline{r}^{req}_{d_n}$. Sorting d_n by $\overline{r}^{req}_{d_n}$ in a descending manner, the sequence number of d_n for the agent allocation can be expressed as $rank_d(d_n)$. Then, the average agent allocation is used to selected agents for each region. Agent e_r with reward R^r will be adopted in d_n when $1 \leq rank_d(d_n) \leq \lceil \mathbb{N}/2 \rceil$, otherwise agent e_f with reward R^f will be adopted as defined in Eq. (32). As the ever-increasing VNFs have been deployed, the average reliability that each region can provide for VNFs, denoted as $\overline{R}_{d_n}(t)$, is being calculated, along with the deviation as $\Delta R_{d_n}(t) = \overline{R}_{d_n}(t) - \mathbb{E}[\overline{R}_{d_n}(t)]$. If $\overline{R}_{d_n}(t) < \delta$ where $\delta < 0$ is a comparison threshold, reliability-based agent allocation will be adopted to change the agent of d_n. In this way, the division of SFCs and the selection of the agent e_r or e_f in each region are conducted according to average/reliability-based agent allocation explorations, which effectively contributes to training reliability-aware and failure-aware models for the follow-up VNF placement.

To store transition samples, the experience replay buffer, D_n, will be established at each agent in the form of $\{x(t), x(t+1), a_1(t), ..., a_{\mathbb{N}}(t), r_1(t), ..., r_{\mathbb{N}}(t)\}$. Generally, the actor network of each agent interacts with the environment in current policy μ_{θ_n} and generates the actor as $\mu(x(t)|\theta^{\mu}_n)$. It is further combined with random noise as

$$a_n(t) = \mu(x(t)|\theta^{\mu}_n) + \mathcal{N}_t, \tag{35}$$

where $\mathcal{N}_t$ is sampled from the noise process, e.g., Ornstein-Uhlenbeck process, to explore potential better policies so as to solve the exploration-exploit dilemma inherent in MADRL. In this way, $a_n(t)$ is sampled from the above random process and executed at each agent, followed by reward $r_n(t)$ and the new state $x(t+1)$ after each agent has performed its action.

On this basis, a mini-batch buffer $\tilde{D}_n$ will be built to store some transitions, sampled from buffer D_n, for the update training of actor and critic networks at each agent, which keeps an independent and identical distribution of samples to be trained, even when they are generated through a continuous exploration. Traditionally, the experience sampling is randomly executed, leading to the uneven quality of sampling experiences and slow convergence speed due to the imbalance of data. To address this issue, prioritized experience replay is adopted to sample transitions with different priority p^j_n, which can be further represented as

$$p^j_n(t) = rank_a[\alpha \cdot rank_d(r^j_n(t)) + \beta \cdot rank_d(T_j(t))], \tag{36}$$

where $r^j_n(t)$ is the instant reward of agent $\hat{e}_n$ in the j-th transition, and $T_j(t)$ is the number of times that the transition j has been trained. Both $rank_a(\cdot)$ and $rank_d(\cdot)$ are functions that output dimensionless numbers of variables in ascending and descending order, respectively. If $r^j_n(t)$ and $T_j(t)$ are smaller, transition j, with greater $p^j_n(t)$, will be sampled earlier. The probability priority of transition j being sampled can be expressed as

$$Pr^j_n(t) = \frac{p^j_n(t)}{\sum_j p^j_n(t)} + \delta, \tag{37}$$

where $\delta \in (0,1)$ is a probability offset to prevent a low probability of sampling in case $p^j_n(t)$ is too small. In this way, M experiences are sampled from D_n with the priority probability $Pr^j_n(t)$ at each agent, which are further put into $\tilde{D}_n$ with $Pr^j_n(t)$. Repeat the sampling of M experiences until the number of samples in $\tilde{D}_n$ reaches its specified size S.

The pseudocode of MADRL-based training phase is listed in **Algorithm 1**. Lines 1-6 mainly conducts task allocation, including SFC division and agent allocation, which is dynamically changed in Lines 17-18. Lines 7-10 describe the initialization of MADRL-based training phase. Lines 11-15 and Lines 20-26 refer to the establishment of experience replay buffer and prioritized experience sampling, respectively. The local training of each agent directly affects the reliability of VNF placement as well as the accuracy and robustness of the overall RVFP model, presented in detail in the following.

Algorithm 1: MADRL-based Training

Input: SFC s_k, region $d_1, d_2, \ldots, d_{\mathbb{N}}$

1 $d_n \leftarrow num_{d_n}$ VNFs as Eq. (34)
2 Sort d_n by $\overline{r}^{req}_{d_n}$
3 **if** $1 \leq rank_d(d_n) \leq \lceil \mathbb{N}/2 \rceil$ **then**
4 Adopt e_r in d_n
5 **else**
6 Adopt e_f in d_n
7 Initialize D_n, $\mu(x(t))$ and $Q_n(x(t), a(t))$ with θ^{μ}_n and θ^{Q}_n
8 **foreach** *episode* **do**
9 Initialize a random noise process $\mathcal{N}$
10 Recive $s(0)$ and $x(t)=\{o_1(t), ..., o_{\mathbb{N}}(t)\}$ of each agent
11 **for** *t= 1 to max-episode-length* **do**
12 **foreach** *agent* $\hat{e}_n$*=1 to* $\mathbb{N}$ **do**
13 Execute $a_n(t)$ and observe reward $r_n(t)$
14 Observe the new state $x(t+1)$
15 Store $\{x(t), a_1(t), , ..., r_{\mathbb{N}}(t), x(t+1)\}$ in D
16 $x(t) \leftarrow x(t+1)$
17 **if** $\overline{R}_{d_n}(t) - \mathbb{E}[\overline{R}_{d_n}(t)] < \delta$ **then**
18 Change the agent of d_n
19 **foreach** *agent* $\hat{e}_n$*=1 to* $\mathbb{N}$ **do**
20 **foreach** *experience j* **do**
21 $Pr^j_n(t) \leftarrow$Eq. (37)
22 **if** $Rand(0,1) < Pr^j_n(t)$ **then**
23 $\tilde{D}_n \leftarrow experience\ j$
24 **if** *The size of* $\tilde{D}_n$ *=* S **then**
25 Break
26 Local training according to **Algorithm 2**

C. Local Training in Each Agent

The local training in each agent is built on the improved MADDPG, by introducing Gumbel Softmax Trick into action space. Each agent is made up of two parts, the actor network (online actor network θ_n^{μ} and target actor network $\theta_n^{\mu'}$) as well as the critic network (online critic network θ_n^{Q} and target critic network $\theta_n^{Q'}$), where $\theta_n^{\mu'}$/$\theta_n^{Q'}$, consistent with θ_n^{μ}/θ_n^{Q} in structure, is used to calculate the gradient of online network in the training process. The actor/critic network of each agent is updated independently.

The actor network θ_n^{μ} and the target actor network $\theta_n^{\mu'}$ are responsible for the modeling of deterministic policy as $a_n(t) = \mu(x(t)|\theta_n^{\mu})$, a mapping from the state space to the action space. To obtain the optimal policy μ_n^*, θ_n^{μ} is trained to maximize the discount cumulative expected reward as

$$\mu_n^* = \arg\max_{\mu} J(\theta_n^{\mu})^*, \tag{38}$$

where $J(\theta_n^{\mu})^*$ is a variant of $J(\theta_n^{\mu})$ to avoid gradient disappearance of θ_n^{μ}, given by Gumbel Softmax Trick as

$$\begin{aligned} J(\theta_n^{\mu})^* &= GumbleMax(J(\theta_n^{\mu})) \\ &= Softmax(J(\theta_n^{\mu}) - \log[-\log(u)]) \\ &= Softmax(\mathbb{E}_{x,a\sim D}[\sum_t \gamma^t r_n(t)] - \log[-\log(u)]). \end{aligned} \tag{39}$$

Here, u is the random tensor with the same size as $J(\theta_n^{\mu})$, and $r_n(t)$ is the immediate return of agent $\hat{e}_n$ at the time slot t, depending on which optimization problem the agent chooses to solve, i.e., **Reli-Max** or **Fail-Min** as Eqs. (18) and (21). We define $J(\theta_n^{\mu})$=$\mathbb{E}_{x,a\sim D}[\sum_{t\in\mathcal{T}} \gamma^t r_n(t)]$. On this basis, θ_n^{μ} can be constantly adjusted by taking steps in the direction of $\nabla_{\theta_n} J(\theta_n^{\mu})$ through Gradient Ascent (GA) as

$$\nabla_{\theta_n} J(\theta_n^{\mu}) = \frac{1}{S}\sum_j \left[\nabla_{\theta_n}\mu_{\theta_n}(x^j)\nabla_{a_n} Q_n^{\mu}(x^j, a^j)\right], \tag{40}$$

where $a^j = (a_1^j, ..., a_{\mathbb{N}}^j)$ and (x^j, a^j) is the j-th transition in $\tilde{D}_n$, sampled from D_n, and S is the number of samples.

The centralized action-value function, $Q_n^{\mu}(x, a_1, ..., a_{\mathbb{N}})$, is represented at the critic network θ_n^{Q} to simulate the discount cumulative expected reward, defined by

$$Q_n^{\mu}(x, a_1, ..., a_{\mathbb{N}}) = \mathbb{E}_{x,a\sim D}\left[r_n(x, a) + \gamma Q_n^{\mu}(x', a')\right], \tag{41}$$

where a=$(a_1, ..., a_{\mathbb{N}})$, both a'=$(a_1', ..., a_{\mathbb{N}}')$, and x' are the action and state at the next time slot, respectively. In order to evaluate the current action as accurately as possible, the critic network θ_n^{Q} is also constantly adjusted to learn the optimal action-value function $\theta_n^{Q^*}$ with the help of target critic network $\theta_n^{Q'}$, by minimizing the loss function as

$$\mathscr{L}(\theta_n^{Q}) = \frac{1}{S}\sum_j \left[(Q_n^{\mu}(x^j, a^j|\theta_n^{Q}) - y_n^j)^2\right], \tag{42}$$

where Q-value is calculated based on the online critic network θ_n^{Q} of agent $\hat{e}_n$. y_n^j is defined as

$$y_n^j = r_n^j + \gamma Q_n^{\mu'}(x'^j, a_1', ..., a_{\mathbb{N}}'|\theta_n^{Q'})|_{a_n'=\mu'(o_n'^j)}, \tag{43}$$

where μ'=$(\mu'_{\theta_1}, ..., \mu'_{\theta_{\mathbb{N}}})$ is the set of target policies with target actor network parameters $\theta_n^{\mu'}$ ($n\in[1, \mathbb{N}]$), respectively. The Q-value here is calculated based on the target critic network $\theta_n^{Q'}$.

Algorithm 2: Local Training in Each Agent

1 Initialize target actor and critic network with $\theta_n^{\mu'}\leftarrow\theta_n^{\mu}$, $\theta_n^{Q'}\leftarrow\theta_n^{Q}$ at each agent
2 **foreach** *episode* **do**
3 **for** *t= 1 to max-episode-length* **do**
4 **foreach** *agent* $\hat{e}_n$*=1 to* $\mathbb{N}$ **do**
5 Sample S transitions from $\tilde{D}_n$ as $(x^j, a_1^j, ..., a_{\mathbb{N}}^j, r_1^j, ..., r_{\mathbb{N}}^j, x'^j)$, $1\leq j\leq S$
6 $\mathscr{L}(\theta_n^{Q})\leftarrow$Eq. (42)
7 Update θ_n^{Q} by minimizing $\mathscr{L}(\theta_n^{Q})$ with GD
8 $\nabla_{\theta_n} J(\theta_n^{\mu})\leftarrow$Eq. (40)
9 **if** *t mod d* **then**
10 Update θ_n^{μ} with GA on $\nabla_{\theta_n} J(\theta_n^{\mu})$
11 $\theta_n^{Q'}\leftarrow\tau\theta_n^{Q}+(1-\tau)\theta_n^{Q'}$
12 $\theta_n^{\mu'}\leftarrow\tau\theta_n^{\mu}+(1-\tau)\theta_n^{\mu'}$

Then, the online critic network of each agent, θ_n^{Q}, is updated through Gradient Descent (GD) on $\nabla_{\theta_n^{Q}}\mathscr{L}(\theta_n^{Q})$.

The completed pseudocode of local training in each agent is listed in **Algorithm 2**. Firstly, the critic network will be updated after $(x^j, a_1^j, ..., a_{\mathbb{N}}^j, r_1^j, ..., r_{\mathbb{N}}^j, x'^j)$ ($j\in[1, S]$) has been sampled at each agent, as shown in Lines 5-7. Line 8 focuses on its policy gradient calculation, which will be updated after delay d in Line 10 with the consideration that frequent update of actor network policy causes the instability of multi-agent environment and slow convergence speed. The target critic and actor networks will be finally updated in a soft-updated manner in Lines 11-12 of **Algorithm 2**.

V. Experimental Evaluation and Results

In this section, extensive simulations are conducted to evaluate the performance of RVFP along with its results reported. First, the simulation setup is introduced, followed by the evaluation metrics. Then, the simulation results with the performance comparisons, among RVFP, its derivatives, and the representative DRL-based solutions of VNF placement [6], [13], [17], are illustrated.

A. Simulation Setup

In the simulation, the substrate network used for experiments is based on the representative GEANT Network, which includes 45 nodes and 74 links. The whole network is divided into 2 regions based on network scale and the amount of available resources, and consists of 4 agents ($\mathbb{N}$=2). Each of them has its local orchestrator for the local guidance on service delivery, communication with the cross-region orchestrator, and routing of service data flows among regions depending on the service requirements. Each node and link is equipped with CPU and bandwidth resources, whose capacity follow the random distribution among [6, 10] Cores and [300, 1000] Mbps, respectively. Furthermore, the reliability of each node to host VNFs is designed to evenly distributed at [0.97, 0.999].

Built upon the data of Google Cluster-usage Traces and the SLA of its Apps [22], we consider 6 types of VNFs to deliver services. One piece of data, as a job in it, is considered as an SFC, where the tasks therein are regarded as VNFs. The CPU consumption of the task, around [1, 4] Cores, and the data size of the job, around [300, 600] Mbps, are adopted to set

the corresponding CPU request of VNFs and the bandwidth of SFCs, respectively. The reliability requirement of each VNF, associated with its type, is set as numbers randomly among {0.94, 0.95, 0.96, 0.97, 0.98, 0.99} [22]. Besides, the arrival and departure of SFCs follow TOTEM project [1], which dynamically change per 15 minutes.

The training phase, employing the improved MADDPG, is implemented in four candidate agents using the Pytorch framework. Two collaborative agents, e_r and e_f, adopt $R^r(t)$ and $R^f(t)$ as their rewards, respectively. We train these agents for 120000 episodes, each of which has 20 time slots. The discount factor γ is set to be 0.95. The Ornstein-Uhlenbeck process is adopted as the noise process $\mathcal{N}_t$ defined in Eq. (35) when establishing the experience replay buffer. There are 2 replay buffers D_n and 2 mini-batch buffers $\tilde{D}_n$, with the size of 10^4 and 128, respectively, to store experiences. The weights of the target actor/critic network $\theta_n^{\mu'}/\theta_n^{Q'}$ are copied from the online actor/critic network $\theta_n^{\mu}/\theta_n^{Q}$ every 16 iterations. Furthermore, the Adam optimizer is used to update the target networks with soft update parameter $\tau = 0.01$.

To estimate the performance of RVFP, four derivatives of it, i.e., RVFP^{++}, RVFP^{+-}, RVFP^{-+} and RVFP^{--}, are developed. Both RVFP^{++} and RVFP^{--} adopt two agents with the identical optimization targets, i.e., both **Reli-Max** or **Fail-Min**, respectively, while RVFP^{+-} and RVFP^{-+} use two agents with different optimization rewards. Differently, RVFP^{+-} gives higher priority to agent e_r with reward $R^r(t)$ mentioned in the average agent allocation, while RVFP^{-+} assigns VNFs with higher reliability requests to agent e_f in priority. Furthermore, three typical related solutions, i.e., Neural Combinatorial Optimization (NCO) [13], DRNA [17], and Betweenness centrality Algorithm for Component Orchestration (BACON) [6], are also implemented. Both NCO and DRNA are two DRL-based initiatives for optimal VNF placement with extended neural combinatorial optimization theory, and reliability-aware considerations, respectively. While, BACON is a static and heuristic effort for VNF placement, with the aim at minimizing the delays and enhancing the reliability of VNFs.

B. Evaluation Metrics

There are six evaluation metrics, i.e., average episode reward, average reliability, average network cost, average reliability per unit cost, acceptance ratio, and placement overhead, considered to estimate the performance of RVFP.

1) Average Episode Reward: It is the average value of rewards received by each agent along with episodes. The rewards for agent e_r and e_f are referred to as $R^r(t)$ and $R^f(t)$, defined in Eqs. (31) and (32), respectively.

2) Average Reliability: The average reliability of SFCs, whose a number of VNFs $v_{k|u}$ have been placed on main nodes and backups, can be given with the help of $R_{v_k}(t)$, as defined in Eq. (7).

3) Average Network Cost: It refers to the average cost of deploying each SFC on the network, which further includes the VNF set cost C_{set}, VNF operation cost C_{op} and communication cost C_{rt} as defined in Eqs. (9), (10), and (11).

4) Average Reliability per Unit Cost: The average reliability per unit cost is denoted as the quotient of average reliability and network cost, which can be expressed as $R_{v_k}(t)/[C_{set}(t)^*+C_{op}(t)^*+C_{rt}(t)^*]$ as defined in Eq. (18). Network cost here is the sum of normalized set cost $C_{set}(t)^*$, VNF operation cost $C_{op}(t)^*$ and communication cost $C_{rt}(t)^*$.

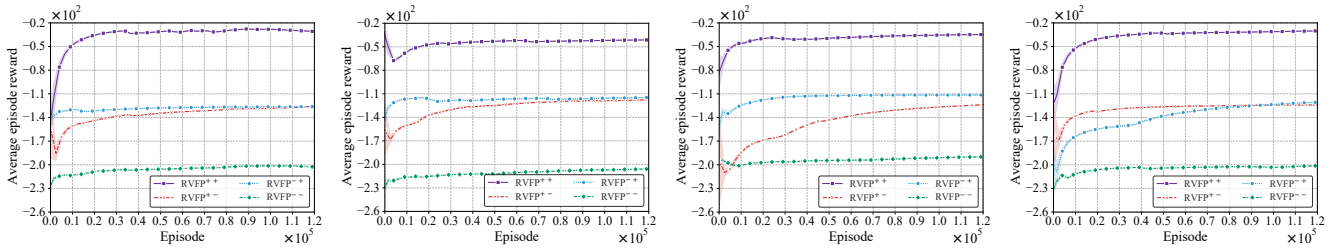
5) Acceptance ratio: The acceptance ratio, A_r, is defined as the ratio of the number of SFCs successfully deployed, $|\mathcal{S}_a|$, to the total number of SFCs, $|\mathcal{S}^v|$, denoted by $A_r = \frac{|\mathcal{S}_a|}{|\mathcal{S}^v|}$. It is used to indicate the probability of successful reliability-aware placement of VNFs.

6) Placement overhead: The placement overhead refers to the product of the amount of message delivered and the average delivered network distance in model training and test.

C. Simulation Results

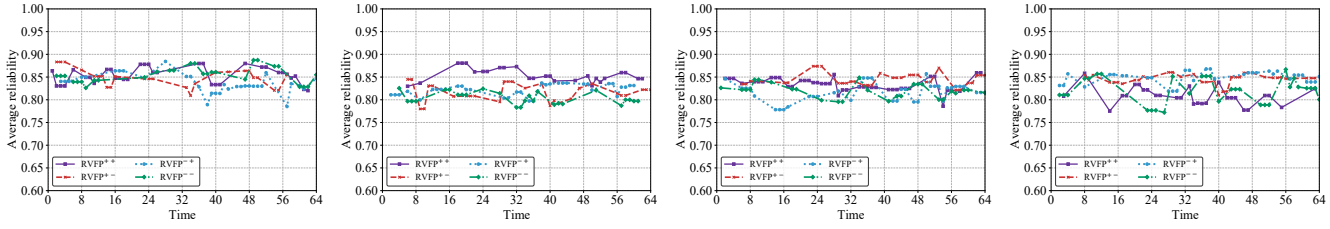
1) Convergence of RVFP^{++}, RVFP^{+-}, RVFP^{-+} and RVFP^{--}: The convergence of RVFP^{++}, RVFP^{+-}, RVFP^{-+} and RVFP^{--} are shown in the form of average episode reward in Fig. 3, where the learning rate of actor networks and critic networks are successively set as 0.001 and 0.01 in online and target networks of each agent during the training process. It can be seen that the average episode reward increases as the episode increases no matter what the values of actor and critic networks are set, indicating that agents continually learn advantageous knowledge in their interactions with the environment. Taking Fig. 3(a) as an example, the smoothed average episode rewards of RVFP^{++}, RVFP^{+-}, RVFP^{-+} and RVFP^{--} gradually increase after some fluctuations during the exploration, which gradually converge to around -37, -121, -125 and -204 after about 30000, 30000, 5000 and 5000 training episodes, respectively. The convergence of average episode reward signifies that RVFP has learned the hidden rules to choose the great actions. In contrast to Fig. 3(a), the average episode rewards in Figs. 3(b), 3(c) and 3(d) are iterated to convergence with less fluctuation, which is caused by the bigger learning rates of actor and critic networks and their faster convergence, especially Fig. 3(d) with both larger learning rates of actor and critic networks. More specifically, it can be seen from Figs. 3(c) and 3(d) that the change on learning rate of actor networks leads to a sharp increase on the reward of RVFP^{+-} and RVFP^{-+} during the 30000 and 35000 episodes, respectively, even after a period of stability. This is on account of the good explorations in policy update by agents in actor networks. Furthermore, RVFP^{++} owns the greatest average episode reward while RVFP^{--} performs the fastest rate of convergence with the worst performance on reward, especially when the learning rate of actor and critic networks keeps equal as shown in Figs. 3(a) and 3(d).

Overall, no matter which kind of agents is used in two regions for jointly training, RVFP can be trained to be convergent with different learning rates. With the appropriate settings of learning rates, the time for models of RVFP to converge is short enough to be ignored, thus RVFP is capable of achieving efficient model training in the limited time and conducting effective reliability-aware placement of VNFs.



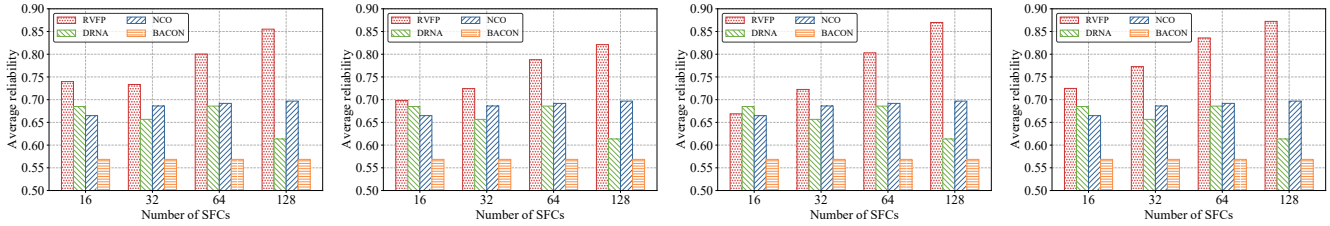
(a) $\eta_a = 0.001$ and $\eta_c = 0.001$ (b) $\eta_a = 0.001$ and $\eta_c = 0.01$ (c) $\eta_a = 0.01$ and $\eta_c = 0.001$ (d) $\eta_a = 0.01$ and $\eta_c = 0.01$

Fig. 3. Convergence of RVFP^{++}, RVFP^{+-}, RVFP^{-+} and RVFP^{--} with different settings of learning rates of actor and critic networks, i.e., η_a and η_c.



(a) $\eta_a = 0.001$ and $\eta_c = 0.001$ (b) $\eta_a = 0.001$ and $\eta_c = 0.01$ (c) $\eta_a = 0.01$ and $\eta_c = 0.001$ (d) $\eta_a = 0.01$ and $\eta_c = 0.01$

Fig. 4. Instant performance of RVFP^{++}, RVFP^{+-}, RVFP^{-+} and RVFP^{--} on reliability during the first 64 time slots with different settings of learning rates of actor and critic networks, i.e., η_a and η_c.



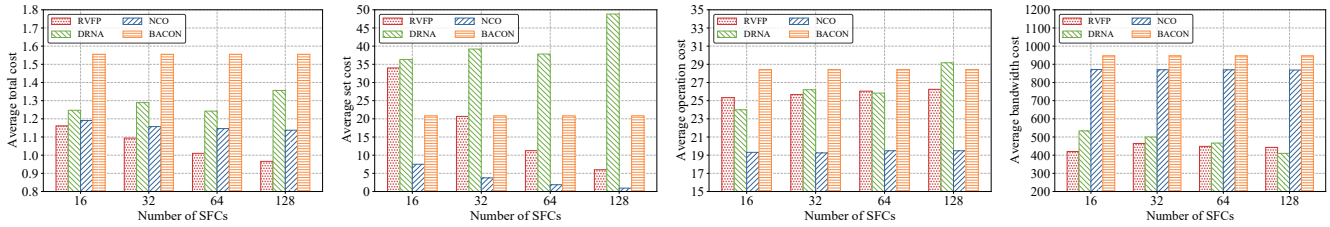
(a) $\eta_a = 0.001$ and $\eta_c = 0.001$ (b) $\eta_a = 0.001$ and $\eta_c = 0.01$ (c) $\eta_a = 0.01$ and $\eta_c = 0.001$ (d) $\eta_a = 0.01$ and $\eta_c = 0.01$

Fig. 5. Average reliability of SFCs achieved by RVFP, DRNA, NCO and BACON with different settings of learning rates of actor and critic networks, i.e., η_a and η_c, when the number of SFCs varies from 16 to 128.

2) Comparisons in Reliability: Based on the convergent models and trained agents as above, VNFs, dynamically arriving in the form of SFCs, are practically placed and orchestrated in the substrate networks. As a consequence, Fig. 4 shows the instant performance of RVFP^{++}, RVFP^{+-}, RVFP^{-+} and RVFP^{--} during the first 64 time slots under various settings of η_a and η_c, where each point represents the average reliability of SFCs deployed at each time slot. It can be seen from Fig. 4 that the reliabilities achieved by RVFP^{++}, RVFP^{+-}, RVFP^{-+} and RVFP^{--} are basically maintained around [0.78, 0.9], which effectively verify the ability and stability of RVFP to conduct VNF placement regardless of the learning rate settings of the two neural networks. Furthermore, the average reliability attained by RVFP^{+-} owns the best stability with the variance as 0.00017, compared with RVFP^{++}, RVFP^{-+} and RVFP^{--}. When η_a and η_c are both set as 0.001, it can be seen from Fig. 4(a) that the reliability of SFCs achieved by all four solutions are 0.853, 0.850, 0.836 and 0.854 on average, which are greater than that under other learning rate settings in Figs. 4(b)-(d). In this way, the subsequent simulations are implemented with RVFP^{+-} and the learning rates of actor and critic networks are set as 0.001.

In Fig. 5, the average reliabilities per unit cost obtained by RVFP, DRNA, NCO, and BACON are calculated with the number of SFCs ranging from 16 to 128 under various settings of η_a and η_c. It can be clearly seen from Fig. 5 that RVFP almost performs the best in comparison with DRNA, NCO, and BACON no matter what the values of actor and critic networks are set. More specially, the average reliability per unit cost of RVFP is greater than 0.7 at all SFC counts, even reaching 0.85 when the number of SFCs is 128, while that of the other three approaches, i.e., DRNA, NCO, and BACON, are less than 0.7. The average reliability per unit cost of RVFP in Fig. 5(a) significantly increases by around 19.0%, 14.1% and 37.6% on average, compared with DRNA, NCO and BACON, especially when there are massive SFCs to be deployed. Furthermore, DRNA and NCO perform similarly on account of their similar cost-minimum rewards and requirements for reliability. BACON, a static solution for VNF placement, performs worst and remains the same reliability for different number of SFCs in VNF dynamic deployment scenarios because the characteristics of static deployment make SFCs that can be successfully deployed in BACON are always the first few ones, regardless of the number of SFCs to be deployed. Therefore, RVFP, combining backups with dynamic reliability-aware VNF placement, is certified to implement better reliability with less network cost than others.

3) Comparisons in Network Cost: Fig. 6 illustrates the average network cost of SFCs achieved by RVFP, DRNA, NCO and BACON, in terms of the total cost, VNF set cost, VNF operation cost and communication cost, with the number of SFCs varying from 16 to 128. The total cost shown in



(a) Average total cost (b) Average VNF set cost (c) Average VNF operation cost (d) Average communication cost

Fig. 6. Average network cost of SFCs achieved by RVFP, DRNA, NCO and BACON, which refers to the total resource cost, VNF set cost, VNF operation cost and communication cost, when the number of SFCs varies from 16 to 128.

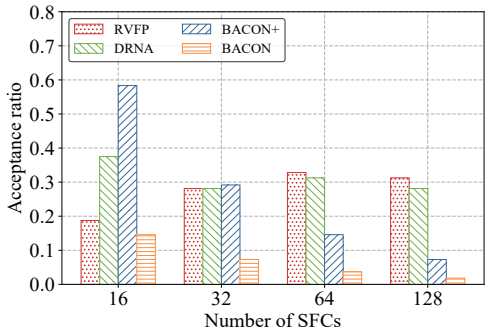


Fig. 7. Acceptance ratio of SFCs achieved by RVFP, DRNA, BACON and BACON+ when the number of SFCs varies from 16 to 128.

Fig. 6(a) is the sum of normalized values of VNF set cost, VNF operation cost and communication cost as in Fig. 6(b), Fig. 6(c) and Fig. 6(d), respectively. It can be seen from Fig. 6(a) that there is a considerable reduction in the average total cost for RVFP, nearly 15.4%, as the number of SFCs changes from 16 to 128. Furthermore, RVFP performs best at the average total cost regardless of the number of SFCs to be deployed and reduces it by 14.2%, 6.7% and 30.5% compared with DRNA, NCO and BACON. In order to acquire more insightful understanding, the detail about components of the average total cost, i.e., VNF set cost, VNF operation cost and communication cost, are shown in Fig. 6(b), Fig. 6(c) and Fig. 6(d). In Fig. 6(b), the average set cost of RVFP gradually decreases as the number of SFCs increases due to its reuse of instantiated VNFs and thus reduced resource fragmentation. In contrast, DRNA performs well in terms of operation and communication costs, but expends a mass of set cost, because it ignores the overhead of repeatedly instantiating VNFs. Besides, NCO performs best in the set and operation costs but bad in the communication cost on account of its inability to capture communication resources. In this way, it can be proved that RVFP implements efficient reliability-aware VNF placement at a lower network cost.

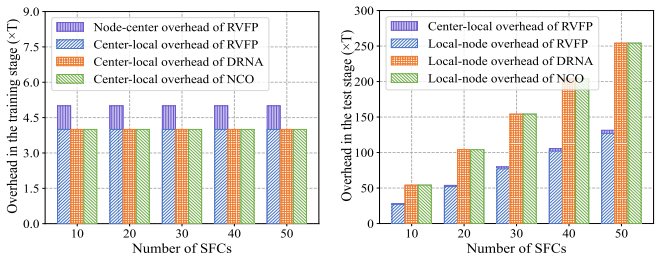
4) Comparisons in Acceptance Ratio: To fairly compare the SFC acceptance ratio of each approach in dynamic SFC deployment scenarios, the static BACON has been improved into a dynamic version, termed as BACON+, which deploys VNFs dynamically in groups of four SFCs. Fig. 7 evaluates the acceptance ratios obtained by RVFP, DRNA, BACON+ and BACON as the number of SFCs changes from 16 to 128. As a whole, the acceptance ratio of RVFP increases along with the number of incoming SFCs even when there are massive SFCs to deploy, which is because the agents of RVFP have learned to constantly interact with the environment and release resources, so as to deploy as many VNFs as possible with the restrictions of resource and reliability. Besides, the acceptance ratio of DRNA remains nearly constant. Differently, BACON and BACON+ both perform well at the beginning, especially BACON+ which utilizes as many resources as possible to provide service for SFCs, but they progressively become worse as the number of SFCs increases. Furthermore, it can be clearly seen that BACON+ greatly improves the acceptance ratio of SFCs, compared with static BACON, on account of dynamic utilization of resources. Therefore, it can be proven that RVFP performs best on the reliability-aware VNF placement in a wide range of SFCRs.

5) Placement Overhead: The placement overhead of RVFP mainly occurs during the model training and test. In order to facilitate the understanding, it is supposed that the average placement overhead, per unit data, from the center orchestrator to servers in region d_i is T, while that from the local orchestrator to servers in region d_i is $T/2$, and that from the center orchestrator to the local orchestrator is denoted as T'.

At the training stage, the placement overhead of RVFP is incurred in the center orchestrator at two main stages, i.e., network state collection and trained model delivery. Let Q and μ denote the data size of trained model and the number of message related to network states, respectively. Then, the overhead of network state collection can be represented as $\mu(T/2 + T')$, while the overhead of model delivery to all local orchestrators can be calculated as QT'. Thus, the overhead of RVFP at the training stage can be represented as $(\mu(T/2+T')+QT')$. Compared with RVFP, the single-agent-based DRNA and NCO only have the overhead of network states collection, denoted as μT.

As for the test stage, there are two kinds of information, i.e., the requirements of SFCs from the center orchestrator and the states of servers, which are collected by the local orchestrators. Let δ denote the data size of SFCs. Then, the first and second overheads can be represented as $\delta|V_k^v||S^v|T'$ and $\mu T/2$, respectively. Furthermore, the decisions of $|V_k^v||S^v|$ VNF deployments along with their requirements are sent to servers in each region, with the overhead of $|V_k^v||S^v|(\delta + 1)T/2$. Thus, the overhead of RVFP at the test stage is $(\delta|V_k^v||S^v|T' + \mu T/2 + |V_k^v||S^v|(\delta + 1)T/2)$. However, the overhead of DRNA and NCO only occurs between the agent and network nodes, which just contains the last two parts of the overhead of RVFP, denoted as $(\mu T + |V_k^v||S^v|(\delta + 1)T)$.

The placement overheads of RVFP, DRNA and NCO at the training and test stages, with the number of SFCs $|V_k^v|$ ranging from 10 to 50, are shown in Fig. 8. The number



(a) The overhead in the training stage (b) The overhead in the test stage

Fig. 8. The placement overhead of RVFP, NCO and DRNA when the number of SFCs varies from 10 to 50.

TABLE II
TIME COMPLEXITY

Solutions	Time complexity
RVFP	$O(EPNQ(M+Q)(\|F\|\|G\|+\|H\|\|J\|K))$
DRNA	$O(EPQM(M+Q)\|F\|\|G\|)$
NCO	$O(EPQ(M+Q)(M+Q^2M^2))$
BACON	$O(P(M+\log Q)+M\log M)$

of regions, nodes and VNFs in an SFC is set as 2, 45 and 10, respectively. Notice that the placement overhead from the local orchestrator to nodes is much more than that to center orchestrator, thus T'=$T/45$. Besides, the data size of trained model, Q, is set to be 1. It can be seen from Fig. 8 (a) that RVFP only generates slight more overhead than the single-agent schemes in the training stage, which is caused by the delivery of trained neural networks to the local orchestrators in each region. But, RVFP significantly reduces the overhead in the test stage, as shown in Fig. 8 (b), due to the decentralized execution and the closeness between the agent and servers. Besides, the saving overhead further increases with the number of SFCs, which greatly offsets its redundant overhead in the training stage and indicates the advantages of RVFP in the execution of VNF deployment.

6) Time Complexity: The time complexities of RVFP, DRNA, NCO and BACON have been elaborated in TABLE II.

The time complexity of RVFP is referred to as the environment establishment and the model training. The environment establishment of RVFP refers to N agents, hence its time complexity is $O(N)$. There are two kinds of neural networks in each MADDPG-based agent, i.e., the actor and the critic networks, thus the time complexity of model training also contains two parts. Let M and Q represent the number of nodes and VNFs of each SFC, respectively. The complexity in input layers of critic and actor neural networks can be counted as $O(M+Q)$. For critic networks, the number of neurons in two hidden layers and the output layer are $|F|$, $|G|$ and 1, respectively, thus the time complexity of each iteration loop of the critic neural network can be expressed as $O((M+Q)|F||G|)$. Let $|H|$, $|J|$ and K represent the number of neurons in two hidden layers and the action space as the output layer of actor network, thus the time complexity of each iteration loop in it is $O((M+Q)|H||J|K)$. Notice that there are P SFCs trained E episodes, therefore the time complexity of RVFP can be described as $O(EPNQ(M+Q)(|F||G|+|H||J|K))$.

DRNA exploits DQN as its learned framework, which holds the same training episodes, the number of training SFCs and neurons in the input layer with RVFP. Let $|F|$ and $|G|$ be the number of neurons in two hidden layers of DQN, thus the number of neurons in its output layer is equal to the number of nodes M. Therefore, the time complexity of DRNA can be represented as $O(EPQM(M+Q)|F||G|)$.

NCO is a DRL-based initiative to optimize VNF placement based on stacked LSTM cells. The time complexity of NCO can be described as $O(EPQ(M+Q)(M+Q^2M^2))$.

The time complexity of BACON is mainly dependent on the length of SFCs and the number of nodes. Given P SFCs with the length of Q and M nodes, the time complexity of BACON can be given by $O(P(M+\log Q)+M\log M)$.

It is seen from our above-mentioned qualitative performance evaluation that these DRL-based schemes all can execute dynamic and intelligent placement of VNFs, which are greatly applied to telecom networks to provide QoS-based services for users. Furthermore, the high reliability and service acceptance ratios of RVFP enable it to be especially applicable to URLLC services like autonomous driving and online treatment over 5G networks. However, the static heuristic BACON is only suitable for fixed and conventional scenarios of VNF deployment.

VI. RELATED WORK

Due to the occasional working vulnerability of NFV, the reliability-aware deployment of SFCs has attracted considerable efforts [11], [1], [12], [10], along with the goals of delay improvement [23], [6], resource cost reduction [24], [25], and routing optimization [26], etc. A comprehensive investigation of such solutions can be found in [1]. Taking into consideration the application scenarios, the previous work can be divided into static and dynamic reliability-aware VNF placement.

The static solutions mainly focus on available network resources and failure probabilities of nodes for given diverse VNFs with reliability requirements. The fault correction issues are considered after existing SFC deployment fails to service reliably [27], [28]. The faulty VNF modules have been firstly isolated in [27], and then been reconfigured neighboring modules to resume the failed operations. In [28], the backups are selected, based on the centrality and reliability of nodes, for some VNFs of SFCs that have suffered from working vulnerability. Different from [27] and [28], both SFC placement and redundant backup server allocation for VNFs have been designed simultaneously in [2], [11], [9], [29]. The states of SFCs are replicated to standby NFs (both local and remote) in [2], so as to support efficient resiliency of further failure detection and failover mechanisms. For the case of one primary VNF instance with one backup, a logarithmic approximation ratio-based approximation algorithm has been proposed in [9], aiming at the configuration of reliability-aware VNFs in mobile edge computing scenarios. Similarly, proprietary backup is adopted in [29] with the recognition of virtual monitoring functions. Unlike proprietary backup, an "N+P" shared backup model is proposed for a single VNF with "N" active instances and "P" backups.

The dynamic solutions are proceeded with the dynamic nature of available resources as well as the arrival and departure of services, where network utility will degenerate a lot if static approaches are still used, as verified in [30]. Therefore,

dynamic reliability-aware VNF placement is proposed in [22], [17], [10] and [31]. The infinite Markov decision process is proposed in [22], followed by the Viterbi-based reliable static service placement algorithm and its dynamic value iteration to evaluate and determine the best deployment approaches, respectively. An online GVB strategy is proposed in [31] to use renewable energy in large data centers and edge servers to provide resources for VNF backups. In [10], the reliability-enhanced provisioning of VNFs is formulated as distributed one-slot optimization problems and addressed with a two-stage online solution with a constant approximation ratio. In addition, a DRL-based approach is proposed in [17] to model the reliability-aware deployment of VNFs with only main servers and no backups. Although preliminary progresses have been made, the existing schemes cannot perform well in terms of intelligence and the rate of convergence, especially when the deployment on main and backup servers are simultaneously proceeded. For this reason, this paper proposes a reliability-aware VNF placement approach for NFV-based networks via MADRL, where MADDPG-based model training is executed in collaborative multi-agents to host NFV services in appropriate main and backup servers in a quickly convergent manner.

VII. Conclusion

In this paper, we proposed RVFP, a reliability-aware VNF service provisioning approach, for NFV-based networks via MADRL, where each NFV service is hosted in appropriate equipment with reliability guarantees. To achieve this goal, the improved MADDPG-based DRL is introduced into RVFP and executed in multiple collaborative agents to train reliability-aware model of VNF placement in a quickly convergent manner. Both optimization objectives, i.e., maximizing the reliability and minimizing the failure probability of VNF placement, have been designed to train models in replay buffer, avoiding the strong correlation of sample exploitation. Furthermore, the newly-designed prioritized experience sampling, built on reward and sampling frequency, is used to exploit useful experience for model training with even faster convergence. Besides, the average and reliability-based agent allocation explorations have been developed for two collaborative agents in each region to further train reliability-aware and failure-aware models of VNF placement, enabling VNFs to work more efficiently. Simulation results show that RVFP has significantly improved placement reliability and service acceptance ratio to place VNFs in networks compared with the state-of-the-art schemes over various scenarios.

Acknowledgment

This work is partially supported by the Natural Science Foundation of China (No.62372192), and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement (No.101030505). This article reflects only the authors' view. The European Union Commission is not responsible for any use that may be made of the information it contains.

References

[1] A. Laghrissi and T. Taleb, "A Survey on the Placement of Virtual Resources and Virtual Network Functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2019.

[2] S. G. Kulkarni, G. Liu, K. K. Ramakrishnan, M. Arumaithurai, T. Wood, and X. Fu, "REINFORCE: Achieving Efficient Failure Resiliency for Network Function Virtualization-Based Services," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 695–708, 2020.

[3] O. A. Wahab, N. Kara, C. Edstrom, and Y. Lemieux, "MAPLE: A Machine Learning Approach for Efficient Placement and Adjustment of Virtual Network Functions," *J. Netw. Comput. Appl.*, vol. 142, pp. 37–50, 2019.

[4] N. Moradi, A. Shameli-Sendi, and A. Khajouei, "A Scalable Stateful Approach for Virtual Security Functions Orchestration," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1383–1394, 2021.

[5] I. El Mensoum, O. A. Wahab, N. Kara, and C. Edstrom, "MuSC: A Multi-stage Service Chains Embedding Approach," *J. Netw. Comput. Appl.*, vol. 159, p. 102593, 2020.

[6] H. Hawilo, M. Jammal, and A. Shami, "Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 643–655, 2019.

[7] P. K. Thiruvasagam, V. J. Kotagi, and C. S. R. Murthy, "A Reliability-Aware, Delay Guaranteed, and Resource Efficient Placement of Service Function Chains in Softwarized 5G Networks," *IEEE Trans. Cloud. Comp*, vol. 10, no. 3, pp. 1515–1531, 2022.

[8] "Vodafone Hit by Three-Hour Mobile Network Outage in Germany," 2020. [Online]. Available: https://www.reuters.com/article/uk-vodafone-group-germany-idUKKBN2831WE.

[9] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan, "Reliability-Aware Virtualized Network Function Services Provisioning in Mobile Edge Computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2699–2713, 2020.

[10] Y. Qiu, J. Liang, V. C. M. Leung, X. Wu, and X. Deng, "Online Reliability-Enhanced Virtual Network Services Provisioning in Fault-Prone Mobile Edge Cloud," *IEEE Trans. Wire. Comm.*, vol. 21, no. 9, pp. 7299–7313, 2022.

[11] A. Engelmann and A. Jukan, "A Reliability Study of Parallelized VNF Chaining," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.

[12] R. Yu, G. Xue, and X. Zhang, "QoS-Aware and Reliable Traffic Steering for Service Function Chaining in Mobile Networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2522–2531, 2017.

[13] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual Network Function Placement Optimization with Deep Reinforcement Learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 292–303, 2020.

[14] B. He, J. Wang, Q. Qi, H. Sun, and J. Liao, "Towards Intelligent Provisioning of Virtualized Network Functions in Cloud of Things: A Deep Reinforcement Learning Based Approach," *IEEE Trans. Cloud. Comp*, vol. 10, no. 2, pp. 1262–1274, 2022.

[15] J. S. Pujol Roig, D. M. Gutierrez-Estevez, and D. Gündüz, "Management and Orchestration of Virtual Network Functions via Deep Reinforcement Learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 304–317, 2020.

[16] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently Embedding Service Function Chains with Dynamic Virtual Network Function Placement in Geo-Distributed Cloud System," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, 2019.

[17] H. R. Khezri, P. A. Moghadam, M. K. Farshbafan, V. Shah-Mansouri, H. Kebriaei, and D. Niyato, "Deep Reinforcement Learning for Dynamic Reliability Aware NFV-Based Service Provisioning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[18] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Is Multiagent Deep Reinforcement Learning the Answer or the Question? A Brief Survey," 2018.

[19] T.-Y. Tung, S. Kobus, J. P. Roig, and D. Gündüz, "Effective Communications: A Joint Learning and Communication Framework for Multi-Agent Reinforcement Learning Over Noisy Channels," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2590–2603, 2021.

[20] N. Toumi, M. Bagaa, and A. Ksentini, "On Using Deep Reinforcement Learning for Multi-Domain SFC Placement," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[21] E. Jang, S. Gu, and B. Poole, "Categorical Reparameterization with Gumbel-Softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[22] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "A Dynamic Reliability-Aware Service Placement for Network Function Virtualization (NFV)," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 318–333, 2020.
[23] X. Gao *et al.*, "Virtual Network Mapping for Reliable Multicast Services with Max-Min Fairness," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2015, pp. 1–6.
[24] W. Ding, H. Yu, and S. Luo, "Enhancing the Reliability of Services in NFV with the Cost-Efficient Redundancy Scheme," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.
[25] H. D. Chantre and N. L. S. da Fonseca, "Multi-Objective Optimization for Edge Device Placement and Reliable Broadcasting in 5G NFV-Based Small Cell Networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2304–2317, 2018.
[26] L. Qu, M. Khabbaz, and C. Assi, "Reliability-Aware Service Chaining in Carrier-Grade Softwarized Networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 558–573, 2018.
[27] M. T. Raza, S. Lu, M. Gerla, and X. Li, "Refactoring Network Functions Modules to Reduce Latencies and Improve Fault Tolerance in NFV," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2275–2287, 2018.
[28] Y. Wang *et al.*, "Reliability-Oriented and Resource-Efficient Service Function Chain Construction and Backup," *IEEE Trans. Netw. Services Manag.*, vol. 18, no. 1, pp. 240–257, 2021.
[29] P. K. Thiruvasagam, A. Chakraborty, A. Mathew, and C. S. R. Murthy, "Reliable Placement of Service Function Chains and Virtual Monitoring Functions with Minimal Cost in Softwarized 5G Networks," *IEEE Trans. Netw. Services Manag.*, pp. 1–1, 2021.
[30] B. Farkiani, B. Bakhshi, S. A. MirHassani, T. Wauters, B. Volckaert, and F. De Turck, "Prioritized Deployment of Dynamic Service Function Chains," *IEEE/ACM Trans. Netw.*, vol. 29, no. 3, pp. 979–993, 2021.
[31] X. Shang, Y. Liu, Y. Mao, Z. Liu, and Y. Yang, "Greening Reliability of Virtual Network Functions via Online Optimization," in *Proc. Int. Symp. Qual. Service (IWQoS)*, 2020, pp. 1–10.