RuleDRL: Reliability-Aware SFC Provisioning With Bounded Approximations in Dynamic Environments

Yue Zeng, Zhihao Qu, *Member, IEEE*, Song Guo, *Fellow, IEEE*, Bin Tang, *Member, IEEE*, Baoliu Ye, *Member, IEEE*, Jing Li, *Member, IEEE*, and Jie Zhang, *Member, IEEE*

***Abstract*— As a key enabling technology for 5G, network function virtualization abstracts services into software-based service function chains (SFCs), facilitating mission-critical services with high-reliability requirements. However, it is challenging to cost-effectively provide reliable SFCs in dynamic environments due to delayed rewards caused by future SFC requests, limited infrastructure resources, and heterogeneity in hardware and software reliability. Although deep reinforcement learning (DRL) can effectively capture delayed rewards in dynamic environments, its trial-and-error exploration in a vast solution space with massive infeasible solutions may lead to frequent constraint violations and traps in poor local optima. To address these challenges, we propose a RuleDRL algorithm that combines the capability of DRL to capture delayed rewards and the strength of rule-based schemes to explore high-quality solutions without violating constraints. Specifically, we first formulate the reliable SFC provision problem as an integer nonlinear programming problem, which is proven to be NP-hard. Then, we jointly design DRL and rule-based schemes that are coupled to make the final decision and establish a bounded approximation ratio in general cases. Extensive trace-driven simulations show that RuleDRL can save the total cost by up to 65.67% and improve the SFC acceptance ratio by up to 82%, compared to the state-of-the-art solution.**

***Index Terms*—Network function virtualization, edge computing, 5G, deep reinforcement learning.**

I. Introduction

5G IS envisioned as an ideal infrastructure to support numerous promising applications or services, and it includes two key enabling technologies, Network Functions Virtualization (NFV) and Software Defined Networking (SDN) [1]. Among them, NFV abstracts hardware-based network functions into software, which enables network functions to be flexibly implemented on commodity servers. Then, the service can be abstracted into a sequence of virtual network functions (VNFs), called a service function chain (SFC) [2]. This greatly simplifies service deployment and helps service providers save on CAPEX, and OPEX [3], [4], [5]. Moreover, SDN physically separates the network control plane from the forwarding plane, providing a global view of the entire network for the network operators to facilitate resource management [6].

In 5G networks, service failures are common due to software or hardware factors [7]. Naturally, emerging mission-critical applications urge network providers to support highly reliable services, such as medical monitoring and industrial automation, requiring 99.9999% reliability [8]. Frequent service failures can lead to lost turnover, lost potential customers, tarnished brands, and harm to corporate reputation. An effective way to support these highly reliable services is to provide backup instances for its component VNFs. Once a VNF fails, the traffic is redirected to its backup instance, thereby maintaining the service and masking the failure [9].

Although more backups could provide higher reliability, this may incur higher resource costs. Thus, several existing works [9], [10], [11], [12], [13] investigate how to back up and deploy SFCs to meet their reliability expectations with minimal resource cost. Then, efficient rule-based algorithms are designed based on their insights, which can explore high-quality solutions that satisfy constraints, such as computational capacity and reliability constraints. However, their solutions rely on assumptions such as sufficient memory, enough bandwidth, and reliable hardware, which limit their practicality for bandwidth- and memory-intensive VNFs.[1] Moreover, these schemes usually make one-shot decisions over individual SFC requests and fail

Manuscript received 24 December 2022; revised 16 April 2023; accepted 22 May 2023. Date of publication 31 May 2023; date of current version 8 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 61832005, 62172204, and 62102131, in part by the Natural Science Foundation of Jiangsu Province, China under Grants BE2020001-3 and BK20210361, in part by the Future Network Scientific Research Fund, China under Grant FNSRFP-2021-ZD-07, and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization. Recommended for acceptance by E. Damiani. *(Corresponding authors: Zhihao Qu; Song Guo; Baoliu Ye.)*

Yue Zeng and Baoliu Ye are with the State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China (e-mail: zengyue@smail.nju.edu.cn; yebl@nju.edu.cn).

Zhihao Qu and Bin Tang are with the Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing 211100, China (e-mail: cstb@hhu.edu.cn; zhihaoqu@hhu.edu.cn).

Song Guo, Jing Li, and Jie Zhang are with the Department of Computing, The HongKong Polytechnic University, Hung Hom, Hong Kong (e-mail: song.guo@polyu.edu.hk; jing5.li@polyu.edu.hk; 18104473r@connect.polyu.hk).

Digital Object Identifier 10.1109/TSC.2023.3281759

[1]Memory-intensive VNFs are common, such as IDS (Intrusion-Detection System) and NAT (Network Address Translation), which are memory-intensive to cache static rules and NAT tables [14], [15]. Bandwidth-intensive VNFs involve virtual routers that intensively consume bandwidth to forward traffic [16]. Besides, for VNFs, there is a correlation between their memory and computing capacity [17]. Memory-intensive VNFs typically require significant computing capacity to process data efficiently. Then, we need to carefully allocate different types of resources for cost efficiency.

to capture delayed rewards brought by future SFC requests in dynamic environments,[2] resulting in sub-optimal solutions.

DRL is precisely designed to capture delayed rewards in complex dynamic environments, where neural networks are used to extract the hidden rules in complex environments by interacting with them. As a result, several pioneering works [19], [20], [21], [22] apply DRL to SFC provision and show excellent performance and great potential. To further satisfy service reliability, Jia et al. [23] first propose a rule-based scheme to determine the number of backups to satisfy service reliability and then use DRL to schedule VNFs intelligently. However, they invoke some strict assumptions, such as that hardware reliability is homogeneous, software failures are negligible, and such that the backup and deployment problems are addressed separately. These assumptions are usually not held in edge environments, where the reliability of edge sites (ESs) is heterogeneous, and software reliability is non-negligible for services with high-reliability requirements. Moreover, the above-mentioned DRL-based schemes lack theoretical guarantees of performance.

In this paper, we study how to achieve reliable SFC provisioning cost-effectively in dynamic environments by considering all the above-mentioned intrinsic and critical factors, including heterogeneous hardware and software reliability, limited memory, bandwidth, and computing resources. We model these features and formulate the reliable SFC provision (RSFCP) problem as a nonlinear integer programming problem, which is proved to be NP-hard. To tackle this problem, we first exploit a typical DRL algorithm, called Deep Deterministic Policy Gradient (DDPG) [24], to capture delayed rewards in dynamic environments, which can effectively handle our problem with continuous state space and large-scale action space. However, the solution space in RSFCP problem is huge, and trial-and-error exploration in DRL can lead to long training times and poor local optima (as shown in Fig. 1(a), DRL converges to a solution worse than a simple rule-based scheme). Moreover, RSFCP problem suffers from multifold constraints resulting in massive infeasible solutions in the solution space, and trial-and-error exploration leads to frequent constraint violations, which in turn lead to low SFC acceptance ratios (as shown in Fig. 1(b)). To address the above challenges, we incorporate a rule-based scheme into DDPG, where DDPG is responsible for capturing delayed rewards and outputting a fractional solution to guide the rule-based decision. The rule-based scheme is designed based on our insights (the least reliable VNF limits SFC reliability), which prioritizes adding backup instances for the least reliable VNFs and deploys these instances on ESs with sufficient resource capacity under the guidance of the fractional solution to satisfy SFC reliability without violating capacity constraints. Finally, rigorous theoretical proof and trace-driven simulation results verify our algorithm's optimality, superiority, and robustness.

The main contributions of this paper are summarized as follows.

[2]**Delayed reward**. In dynamic environments, the current decision determines the immediate reward and the next state of the environment [18]. The **delayed reward** is a metric to measure the benefits of the next state, which indicates the impact of the current decision on the future, helping to achieve higher long-term returns.

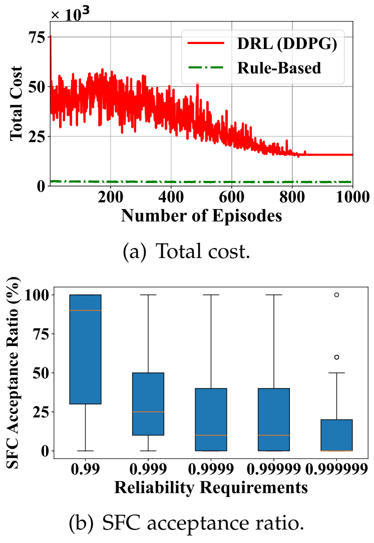


Fig. 1. Motivation. (a) DRL starts with a poor solution, requires long training times, and gets stuck in poor local optima. (b) DRL suffers from frequent constraint violations, resulting in low SFC acceptance ratios.

- To the best of our knowledge, we are the first to study how to achieve reliable SFC provisioning cost-effectively in dynamic environments, considering intrinsic and critical factors, including heterogeneous hardware and software reliability and limited infrastructure resources. Specifically, we formulate the reliable SFC provisioning problem as an integer nonlinear programming and prove its NP-hardness.
- We design RuleDRL to combine the advantages of both DRL and rule-based schemes, where DDPG is used to capture delayed rewards, and rule-based schemes are used to incorporate our insights to explore a cost-efficient feasible solution. This algorithm is rigorously proven to output solutions with bounded approximation guarantees in general cases. In particular, it conditionally achieves a constant approximation ratio in most realistic scenarios.
- Extensive simulation results driven by Alibaba trace show that, compared with the state-of-the-art solution, RuleDRL can save the total cost by up to 65.67% and improve the SFC acceptance ratio by up to 82%.

The rest of the paper is organized as follows. Section II briefly introduces the related work. Section III formulates the problem we studied and proves its NP-hardness. RuleDRL is proposed and analyzed in Section IV and evaluated in Section V. Section VI concludes the paper.

II. Related Work

This section summarizes and discusses related works in SFC provision from two categories, i.e., Rule-based solutions and DRL-based solutions.

A. Rule-Based Solution for SFC Provision

There is a series of works [9], [10], [11], [12], [13], [25], [26] have investigated how to support SFC provisioning cost-effectively, and efficient rule-based solutions are designed based

TABLE I
SUMMARY OF RELATED RESEARCH

Related Work	Delayed Reward	SFC	Deployment	Backup	CPU/Memory/Bandwidth	Software/Hardware Reliability
[9], [10]	✗	✓	✗	✓	✗	✗
[11]–[13]	✗	✓	✓	✓	✗	✗
[25], [26]	✗	✗	✓	✓	✗	✓
[20], [21]	✓	✓	✓	✗	✗	✗
[19], [22]	✓	✓	✓	✗	✓	✗
[27]	✓	✓	✓	✗	✗	✗
[23]	✓	✓	✓	✓	✗	✗
This paper	✓	✓	✓	✓	✓	✓

on expert insights. As pioneers, Potharaju et al. [7] revealed that failures of network functions are attributable to software or hardware failures and experimentally verified that backups can effectively mask failures and maintain services. Then, the authors in [9], [10] studied how many backups are required for each VNF to ensure SFC reliability while minimizing backup resource costs and propose rule-based algorithms to provide backups for VNFs with the greatest reliability improvement greedily. Further, considering the limited physical resources, several studies [11], [12], [13], [25], [26] investigated how many instances need to be allocated for each VNF and to which ESs these instances should be deployed to meet service reliability efficiently with resources, and proposed efficient rule-based solutions. The above research has greatly promoted reliable service provision, and a series of efficient rule-based algorithms have been proposed. However, they ignore the limitations of bandwidth and memory resources and the resource cost of memory and bandwidth. Moreover, these rule-based solutions make one-shot decisions, which fail to capture delayed rewards brought by future SFC requests in dynamic environments.

B. DRL-Based Solution for SFC Provision

To cope with complex dynamic environments, several recent works [19], [20], [21], [22], [23], [27] have investigated how to apply DRL for service provisioning. As pioneers, in order to ensure capacity constraints, the authors in [19], [20] added penalties in the reward function to punish capacity violations. Besides, to meet the flow conservation constraints, Pei et al. [21] proposed an efficient rule-based scheme to generate a set of feasible routing paths to ensure the flow conservation constraints, and then DRL is used for routing and deployment decisions. Similarly, Fan et al. [22] proposed a rule-based scheme to generate a set of solutions satisfying capacity constraints and then used DRL to select the best one. However, the reliability requirements of SFCs are out of their concern. Considering reliability, Khezri et al. [27] studied how to utilize DRL to deploy VNFs in SFC to meet its reliability requirements. However, SFC backups are beyond their concern, which fails to provide high reliability for the service. A work closely related to our research is [23], which investigated how to back up and deploy SFCs to satisfy their reliability requirements. However, several assumptions sacrifice their practicality, such as hardware reliability being homogeneous, software reliability being negligible, bandwidth and memory resources being sufficient, and memory and

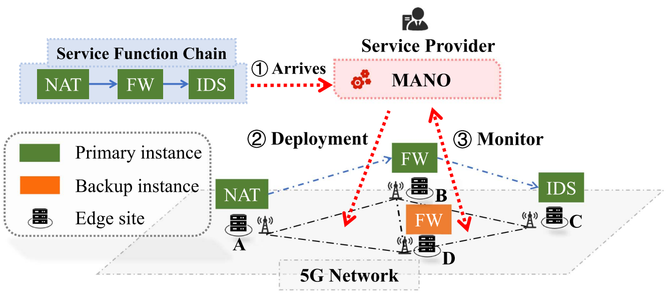


Fig. 2. System Framework.

bandwidth costs being negligible. These assumptions can lead to frequent constraint violations and suboptimal solutions for services with high-reliability requirements and heterogeneous ESs. Moreover, the above-mentioned DRL-based schemes lack theoretical guarantees.

Although existing works have studied how to achieve reliable SFC, they ignore bandwidth and memory capacity, heterogeneous software and hardware reliability, or delayed reward in dynamic environments. We summarize the differences between this paper and existing work in Table I.

III. SYSTEM MODEL AND PROBLEM FORMULATION

This section introduces the system model, formulates the reliable SFC provision problem, and analyzes its complexity.

A. Primer

1) System Model: As shown in Fig. 2, our system consists of three components [28], which can be modeled as follows.

SFC: SFC is the abstraction of the service and consists of multiple VNFs, which can be implemented on commodity servers via virtual machines or containers. In the system, SFC requests arrive one by one. Let S^i denote the i-th SFC request, and $\mathcal{I}$ denote the set of SFC requests. Each SFC request S^i has a reliability requirement $\Re^i$, and it consists of a set of VNFs $\mathcal{V}^i$, where the j-th VNF is denoted as $v^{i,j}$. Each VNF $v^{i,j}$ has software reliability $r^{i,j}$, demand computing resources $c^{i,j}$, and memory resources $m^{i,j}$. After a service's traffic is processed by a VNF, it may vary in size [29], and the outgoing traffic from the VNF $v^{i,j}$ is denoted as $b^{i,j}$.

NFVI: NFV Infrastructure (NFVI) includes diverse physical resources, such as compute, memory, and bandwidth resources, that are virtualized as virtual resources to support flexible VNF execution. In NFVI, there is a set of ESs $\mathcal{E}$, each ES $e \in \mathcal{E}$ with hardware reliability r_e has C_e units of computing resources and M_e units of memory resources. In addition, we use κ_e and ζ_e to denote the costs per unit of computing resources and memory resources, respectively [30], [31]. Let $\mathcal{L}$ denote the set of overlay links connecting the ESs. Each link $(e, e') \in \mathcal{L}$ has a capacity $B_{e,e'}$ and incurring a cost $\tau_{e,e'}$ per unit bandwidth resource [32].

MANO: NFV Management and Orchestration (MANO) is owned by service providers and is responsible for orchestrating and managing virtualized computing, memory, and bandwidth resources in NFVI. Once an SFC request arrives, MANO determines how to implement it on NFVI, continuously monitors the implemented service, and recovers the failure accordingly. The MANO is implemented by fault-tolerant SDN controllers [33], [34], [35] and can be considered reliable.

2) Backup Strategies: To support highly reliable services, there are two typical backup strategies, i.e., on-site backup and off-site backup [26], [36]. The on-site backup refers to providing backup instances in the same ES as the primary VNF instance to mask the VNF instance failures, and it can be restored quickly. However, in this case, the VNF reliability is limited by ES reliability. Once an ES fails, all VNF instances on the ES fail. The off-site backup refers to providing backup instances on geographically separated ESs for the primary VNF instance, which can provide higher reliability and is suitable for applications with high-reliability requirements. Therefore, similar to [10], this backup strategy is adopted in our research to support mission-critical applications with high-reliability requirements.

3) Reliable SFC Provision: In our dynamic environment, the SFC request arrives one by one, and we need to implement them on NFVI, and after the SFC is deployed, the traffic is forwarded along this SFC. Besides, to meet the SFC reliability requirements, each VNF may also need to be added with backup instances. Therefore, reliable service provisioning involves three decisions: 1) how many instances (backup and primary) are needed for each VNF; 2) which ESs should these instances be deployed to; 3) which VNF instances should be selected as the primary instances for forwarding traffic. An example is illustrated in Fig. 2, in which a service is abstracted as an SFC, Network Address Translator (NAT) $\rightarrow$ Fireware (FW) $\rightarrow$ Intrusion Detection System (IDS). Their VNF instances are deployed to ESs A, B, C, D, where the instances deployed to ESs A, B, and C are selected as primary instances for forwarding traffic, and the instance deployed to ES D is used as a backup to improve reliability to meet service reliability requirements. The notations to be used are listed in Table II.

B. Problem Formulation

In our research, we investigate how to achieve SFC provisioning in dynamic environments to meet service reliability requirements while minimizing resource costs under limited resource capacity. Next, we analyze and formulate the cost, reliability, and capacity models.

TABLE II
SUMMARY OF NOTATIONS

Inputs	Descriptions
$\mathcal{E}, \mathcal{I}, \mathcal{L}$	Set of ESs, SFCs, overlay links connecting ESs
$\mathcal{V}^i$	Set of VNFs in SFC S^i
$\Re^i$	Reliability required by SFC S^i
$B_{e,e'}$	Bandwidth capacity of link (e, e')
C_e, M_e	Computing and memory capacity of ES e
$\tau_{e,e'}$	Cost of unit bandwidth resources on link (e, e')
κ_e, ζ_e	Cost of unit computing, bandwidth resources on ES e
r_e	Hardware reliability of ES e
$r^{i,j}$	Software reliability of VNF $v^{i,j}$
$b^{i,j}$	Bandwidth resource required by the traffic from VNF $v^{i,j}$
$c^{i,j}, m^{i,j}$	Computing and memory resources required by VNF $v^{i,j}$
Variables	**Descriptions**
$R_e^{i,j}$	Reliability of VNF $v^{i,j}$ on ES e
$R^i, R^{i,j}$	Reliability of SFC S^i and VNF $v^{i,j}$
Decisions	**Descriptions**
$x_e^{i,j}$	Binary variable indicating whether to deploy VNF $v^{i,j}$ on ES e
$y_e^{i,j}$	Binary variable indicating whether to set the instance deployed on ES e for VNF $v^{i,j}$ as the primary instance
$z_{e,e'}^{i,j}$	Binary variable indicating whether the outgoing traffic from VNF $v^{i,j}$ traverses link (e, e')

1) Cost Model: Implementing SFC to NFVI will incur financial costs, including computational, bandwidth, and memory costs [20], [37]. Specifically, deploying a VNF on an ES will incur computational and memory resource costs [30], [31], while forwarding traffic between primary VNF instances will incur bandwidth resource costs [32].[3] Then, the resource cost of SFC S^i can be defined as

$$\begin{aligned} \mathcal{C}^i = & \sum_{j \in \mathcal{V}^i} \sum_{e \in \mathcal{E}} c^{i,j} \kappa_e x_e^{i,j} + \sum_{j \in \mathcal{V}^i} \sum_{e \in \mathcal{E}} m^{i,j} \zeta_e x_e^{i,j} \\ & + \sum_{j \in \mathcal{V}^i} \sum_{e,e' \in \mathcal{L}} b^{i,j} \tau_{e,e'} z_{e,e'}^{i,j}, \forall i \in \mathcal{I}, \end{aligned} \tag{1}$$

where $x_e^{i,j}$ is a binary variable indicating whether the instance of VNF $v^{i,j}$ is deployed to ES e, and $z_{e,e'}^{i,j}$ is a binary variable indicating whether the traffic from VNF $v^{i,j}$ traverses link (e, e').

2) Reliability: Similar to [39], we define reliability as the probability that an item performs the required function without failure.

For each SFC, once a VNF fails, the entire SFC fails [40]. Similar to [9], [10], [23], [41], we consider that the virtual link connecting VNFs is reliable, which is reasonable.[4] Then, the reliability constraint can be formulated as

$$R^i = \prod_{j \in \mathcal{V}^i} R^{i,j} \geq \Re^i, \forall i \in \mathcal{I}, \tag{2}$$

[3]It should be noted that we only consider the bandwidth cost of forwarding between the primary VNFs. This is because the reliability of VNF software and ES hardware is generally higher than 0.99 [7], [38], which means that the traffic being forwarded to the backup instance is negligible.

[4]According to [42], more than 90% of the underlying links fail no more than once every 5709 hours. The virtual link can be restored quickly (within 100ms) by re-planning an underlying path [43]. That is, the reliability of the virtual link is about $2 * 10^{-8}$, which is negligible for 5G mission-critical applications with reliability requirements not exceeding 10^{-6}[8].

where R^i and $R^{i,j}$ indicate the reliability of SFC S^i and VNF $v^{i,j}$ respectively.

For each VNF, its instances may be deployed on different ESs. A VNF fails only when its instance fails on all ESs [11]. Then, the VNF reliability can be derived as

$$R^{i,j} = 1 - \prod_{e\in\mathcal{E}}(1 - R_e^{i,j}), \forall i \in \mathcal{I}, j \in \mathcal{V}^i, \tag{3}$$

where $R_e^{i,j}$ indicates the reliability of VNF $v^{i,j}$ on ES e.

Once a VNF instance is deployed to an ES, it fails when its VNF instance has a software failure, or the ES has a hardware failure [25]. Thus, the reliability of VNF $v^{i,j}$ on ES e can be denoted as

$$R_e^{i,j} = r^{i,j} r_e x_e^{i,j}, \forall i \in \mathcal{I}, j \in \mathcal{V}^i, e \in \mathcal{E}, \tag{4}$$

where $x_e^{i,j}$ is a binary variable indicating whether VNF $v^{i,j}$ is deployed on ES e.

3) Capacity: In NFVI, physical resources are limited, including bandwidth, memory, and computing resources. The resource capacity constraints can be formulated as

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{V}^i} c^{i,j} x_e^{i,j} \le C_e, \forall e \in \mathcal{E}, \tag{5}$$

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{V}^i} m^{i,j} x_e^{i,j} \le M_e, \forall e \in \mathcal{E}, \tag{6}$$

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{V}^i} b^{i,j} z_{e,e'}^{i,j} \le B_{e,e'}, \forall (e, e') \in \mathcal{L}. \tag{7}$$

4) SFC Completeness: For each SFC, each of its VNFs needs to be deployed with at least one instance as the primary instance for forwarding traffic to ensure service completeness, which can be written as

$$\sum_{e\in\mathcal{E}} x_e^{i,j} \ge 1, \forall i \in \mathcal{I}, j \in \mathcal{V}^i. \tag{8}$$

5) Primary Instance: Each VNF in each SFC has a primary instance for forwarding its traffic (9), and other instances serve as backup instances. The instance on ES e can be served as the primary instance of VNF $v^{i,j}$ only if the instance of the VNF is deployed on the ES (10). The traffic from VNF $v^{i,j}$ traverses link (e, e') only if its primary instance is deployed to ES e and the primary instance of its successor VNF $v^{i,j+1}$ is deployed to ES e' (11). The above conditions can be formalized as

$$\sum_{e\in\mathcal{E}} y_e^{i,j} = 1, \forall i \in \mathcal{I}, j \in \mathcal{V}^i, \tag{9}$$

$$x_e^{i,j} \ge y_e^{i,j}, \forall i \in \mathcal{I}, j \in \mathcal{V}^i, e \in \mathcal{E}, \tag{10}$$

$$z_{e,e'}^{i,j} = y_e^{i,j} y_{e'}^{i,j+1}, \forall i \in \mathcal{I}, j \in \mathcal{V}^i, (e, e') \in \mathcal{L}, \tag{11}$$

where $y_e^{i,j}$ is a binary variable indicating whether the instance of VNF $v^{i,j}$ deployed on ES e is selected as the primary instance.

Thus, the offline version of the reliable SFC provision (RSFCP) problem in dynamic environments can be formulated as

$$\min_{x_e^{i,j}, y_e^{i,j}, z_{e,e'}^{i,j}} \sum_{i\in\mathcal{I}} \mathcal{C}^i$$

$$s.t. \quad (1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11),$$
$$x_e^{i,j} \in \{0, 1\}, y_e^{i,j} \in \{0, 1\}, z_{e,e'}^{i,j} \in \{0, 1\}.$$

C. Problem Complexity

1) Hardness Result: We first prove that it is NP-hard to determine whether RSFCP problem has a feasible solution and then prove that it is still NP-hard when all SFC requests are accepted.

Theorem 1: It is strongly NP-hard to determine whether there is a feasible solution for RSFCP problem.

Proof: This theorem can be proven by showing that a special case of RSFCP problem is equivalent to the minimized version of the general assignment problem (MinGAP) [44]. We assume $|\mathcal{I}| = 1, M_e = B_e = \infty, \Re^i = \zeta_e = \tau_{e,e'} = 0, \forall i \in \mathcal{I}, e \in \mathcal{E}, (e, e') \in \mathcal{L}$. In this case, our problem is equivalent to the MinGAP problem; determining whether it has a feasible solution has been proved to be strongly NP-hard. Therefore, our problem is also strongly NP-hard. □

Theorem 2: Assuming all SFCs are accepted, RSFCP problem is NP-hard.

Proof: This theorem can be proved by showing that a special case of RSFCP problem is equivalent to a classical NP-hard problem, i.e. the 0-1 knapsack problem [45]. We assume $|\mathcal{V}^i| = 1,\ M_e = B_e = \infty,\ \Re^i = \zeta_e = \tau_{e,e'} = 0, \forall i \in \mathcal{I}, e \in \mathcal{E}, (e, e') \in \mathcal{L}$. Besides, there are two ESs A and B, in which $\kappa_A = 0$ and $C_B = \infty$. In this case, our problem is equivalent to the 0-1 knapsack problem [45]. For brevity, the formal proof is omitted. □

Unless *P* = *NP*, the above theorems rule out polynomial time algorithms for this problem, even when all SFC requests are accepted. Although there are approximation algorithms [46] for the 0-1 knapsack problem, RSFCP problem cannot be effectively solved by these algorithms due to the following pain points.

2) Pain Points: The existing solutions to the above combinatorial optimization problems in dynamic environments can be divided into two categories: rule-based solutions and data-driven DRL. The pain points of these solutions to handle RSFCP problem are revealed as follows.

Rule-Based Heuristic: Rule-based heuristics are widely used, which leverage expert insights to explore high-quality solutions subject to constraints. However, rule-based schemes are difficult to handle our problem efficiently for the following reasons. i) Rule-based heuristics make one-shot decisions and fail to capture delayed rewards in dynamic environments brought by future arriving SFCs. Besides, future SFC requests are difficult to predict precisely. ii) In RSFCP problem, backup, deployment, and primary VNF selection decisions are coupled to determine resource cost and reliability due to heterogeneous factors, including hardware and software reliability, resource cost, and capacity. For such a complex problem, effective insights are difficult to obtain to guide the exploration of high-quality solutions, even for experts.

DRL: DRL is designed to handle complex dynamic environments, which can effectively capture delayed rewards, and extract hidden rules from complex environments using neural

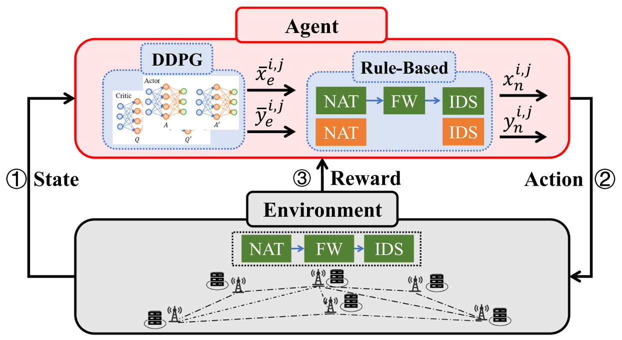


Fig. 3. Algorithm framework.

networks [47]. However, there are several challenges in applying DRL to RSFCP problem. i) DRL interacts with the environment in a trial-and-error manner to learn the rules hidden in the environment; this may lead to frequent constraint violations, including capacity, reliability, and SFC completeness constraints. This means service performance degradation and SLA (service-level agreement) violations, which are unacceptable for mission-critical services. ii) In our problem, the solution space is huge,[5] and random initialization usually leads to a poor initial solution, which in turn leads to long training times, trapping in poor local optima, or even failure to converge, as shown in Fig. 1.

To address these challenges, we next design RuleDRL, which leverages both strengths.

IV. Algorithm Design

This section designs RuleDRL algorithm, which combines the strengths of rule-based heuristics and DRL, where DRL is used to capture delayed rewards and extract hidden rules from complex environments, while the rule-based heuristic is used to make high-quality decisions without violating constraints. Specifically, we first introduce the algorithm framework, then introduce the details of the algorithm, and finally demonstrate its theoretical properties.

A. Algorithm Framework

To capture delayed rewards in dynamic environments, the reinforcement learning (RL) framework is adopted, as shown in Fig. 3. The framework consists of two entities, i.e., the agent and the environment, where the agent discovers the state from the environment, takes action based on the state, and acts on the environment. Then, the environment gives corresponding rewards according to the quality of the action. Finally, the agent updates the policy based on the feedback reward, aiming to maximize the expected cumulative reward.

In DRL, the agent explores the environment in a trial-and-error manner, and the neural network's output acts directly on the environment as an action, which can lead to frequent constraint violations, especially for our problem with many constraints. Inspired by the relaxation and rounding schemes [48], it solves the relaxed integer linear programming to produce a fractional solution and rounds this fractional solution based on expert insight to obtain a high-quality solution that satisfies the constraints.[6] We utilize a typical DRL algorithm for capturing delayed rewards, called DDPG, which can deal with our problem with continuous states and large-scale discrete actions (details in the next subsection). After capturing the delayed reward, DDPG outputs a fractional solution to guide the rule-based scheme to make decisions. Then, we make a backup, deployment, and primary VNF selection decision based on our insights and the fractional solution to obtain a high-quality solution that satisfies all constraints. Finally, this decision is applied to the environment as an action.

B. RL Model

The essential elements required by RL can be described by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$, referring to the state space, action space, and reward, respectively. We define these three elements as follows.

State: As shown in Fig. 3, the state in the environment includes information about that newly arrived SFC request and NFVI, i.e., $\mathcal{S} = \langle \Re_i, r^{i,j}, b^{i,j}, c^{i,j}, m^{i,j}, \kappa_e, \zeta_e, r_e, \tau_{e,e'}, B_{e,e'}, C_e, M_e, \forall j \in \mathcal{V}^i, e \in \mathcal{E}, (e,e') \in \mathcal{L} \rangle$. In this case, the size of the state is $4|\mathcal{V}^i| + 5|\mathcal{L}| + 2|\mathcal{E}| + 1$. Obviously, the change in SFC length leads to a change in state size, which leads to a mismatch with the fixed neural network input layer. To address this challenge, we express the state of the SFC in a redundant manner. According to [49], the length of SFC does not exceed 7, and we can use a fixed length to describe the state of SFC; if the length of SFC is less than 7, we fill it with zero. Then, $S = \langle \Re_i, r^{i,j}, b^{i,j}, c^{i,j}, m^{i,j}, \kappa_e, \zeta_e, r_e, \tau_{e,e'}, B_{e,e'}, C_e, M_e, \forall j \in \{1, \ldots, L\}, e \in \mathcal{E}, (e,e') \in \mathcal{L} \rangle$, where L denotes the maximum SFC length. Then, the state size is $4L + 5|\mathcal{L}| + 2|\mathcal{E}| + 1$.

Action: The action is output by the agent, which is responsible for making backup, deployment, and primary VNF instance selection decisions. In a dynamic environment, SFC requests arrive one by one. Once an SFC request S^i arrives, we need to determine whether to deploy instances on each ES for each VNF in the SFC, and which VNF instances are selected as the primary instance to forward traffic. Then, the action can be formulated as, $\mathcal{A} = \langle x_e^{i,j}, y_e^{i,j}, \forall j \in \mathcal{V}^i, e \in \mathcal{E}, (e,e') \in \mathcal{L} \rangle$. Similar to state representation, we respond to changing action sizes in a redundant manner. Then, $\mathcal{A} = \langle x_e^{i,j}, y_e^{i,j}, \forall j \in \{1, \ldots, L\}, e \in \mathcal{E} \rangle$. The state size is $2L|\mathcal{E}|$.

Reward: The reward is feedback from the environment that indicates the action's quality. We designed the reward function based on RSFCP problem. On the one hand, RSFCP problem aims to minimize the total resource cost, and the higher the total resource cost, the smaller the reward should be. On the other hand, once a decision violates the constraint in RSFCP problem, it may reject the SFC request because of the unacceptable performance degradation for mission-critical applications. This will cause economic losses, and it should be punished. Based

[5]For example, the solution space contains $2^{1000} = 10^{301}$ solutions when $|\mathcal{I}| = 10, |\mathcal{V}^i| = 5, |\mathcal{E}| = 10$.

[6]It should be noted that relaxation and rounding schemes fail to capture the delayed reward brought by the future SFC. The challenge is that the nonlinear integer programming has too many constraints, which is extremely time-consuming to solve.

on the above analysis, the reward function can be formalized as $\mathcal{R} = -\mathcal{C}_i - \mathcal{P}o^i$, where o^i indicates whether the SFC request S^i is satisfied, $\mathcal{P}$ is the penalty value.

C. RuleDRL Algorithm

1) DDPG Algorithm: As modeled above, the state is continuous, and the action is large-scale discrete. Therefore, we apply DDPG, which can deal with both continuous states and large-scale discrete states. After specifying the state, action, and reward, the DDPG is determined. However, as shown in Fig. 1, the result directly output by DDPG may fall into poor local optima and frequently violate constraints due to random exploration in the huge solution space. Therefore, in our algorithm, the output of DDPG is not directly used as an action. Instead, DRL is responsible for outputting a fractional solution that guides subsequent the rule-based heuristic to make decisions. The output of DDPG can be defined as $\mathcal{A}' = \langle \overline{x}_e^{i,j}, \overline{y}_e^{i,j}, \forall j \in \{1, \ldots, L\}, e \in \mathcal{E}\rangle$, where $\overline{x}_e^{i,j}$ and $\overline{y}_e^{i,j}$ are fractions, as a preference indicator, the larger the value, the more preferred.

Next, we briefly explain how DDPG captures the delayed reward as follows. In DDPG, there are two types of networks called actor networks and critic networks. The actor network with parameter θ^A is used to output an action based on the current state, and the critic network with parameter θ^A is used to evaluate the value of the state. The state value indicates the expectation of future rewards for that state and is used to capture the delayed reward. Then, the reward obtained after taking action can be expressed as $\Psi_i = \mathcal{R}_i + \gamma Q(\mathcal{S}_{i+1}, \theta^A, \theta^C)$, where $\mathcal{R}_i$ denotes its immediate reward, γ denotes its discount factor, and $Q(\mathcal{S}_{i+1}, \theta^A, \theta^C)$ denotes its state reward, which implies the expectation of its future reward. $Q(\mathcal{S}_{i+1}, \theta^A, \theta^C)$ is dependent on the state $\mathcal{S}_{i+1}$, the actor network, and the critic network.[7] In this way, both immediate and delayed rewards can be captured. The details of DDPG are omitted for brevity; refer to [24] for details.

2) Rule-Based Algorithm: The rule-based algorithm is responsible for finding a high-quality (low resource cost) solution that satisfies all constraints, including capacity constraints, SFC completeness constraints, and reliability constraints. This algorithm is designed to make SFC backup, deployment, and primary VNF instance selection decisions based on our insights and the fractional solution output by DDPG.

In RSBD problem, the key issue is satisfying the SFC reliability. To this end, we give the following result.

Lemma 1: The reliability of each SFC is limited by its component VNF with the lowest reliability, that is, $R^i \le \min_{j\in\mathcal{V}_i} R^{i,j}$.

Proof: According to (2), we have

$$R^i = \prod_{j\in\mathcal{V}_i} R^{i,j} \le \min_{j\in\mathcal{V}_i} R^{i,j}, \tag{12}$$

where the second inequality is due to $1 > R^{i,j} > 0$. □

[7]It should be noted that information about NFVI in state S_{t+1} can be obtained by taking action in a simulated environment, such as updating the corresponding resource capacity. The information about the next SFC request in $\mathcal{S}_{t+1}$ can be obtained from the request buffer in MANO [50]. If there is no SFC request in the buffer, we randomly select an SFC request from the historical requests and treat it as the next arriving request to get $Q(\mathcal{S}_{i+1}, \theta^A, \theta^C)$.

Algorithm 1: RuleDRL Algorithm.

```
Input:
  $\Re_i, r^{i,j}, b^{i,j}, c^{i,j}, m^{i,j}, \kappa_e, \zeta_e, r_e, \tau_{e,e'}, B_{e,e'}, C_e, M_e$.
Output: $o^i$, $x_e^{i,j}$, $y_e^{i,j}$.
1: $\overline{x}_e^{i,j}, \overline{y}_e^{i,j} \leftharpoondown$ call DDPG;
2: Initialize $x_e^{i,j} \leftharpoondown 0, y_e^{i,j} \leftharpoondown 0, \forall j \in \{1, \ldots, L\}, e \in \mathcal{E}$;
3: $o^i \leftharpoondown 1$; $R^i \leftharpoondown 0$;
4: $R^{i,j} \leftharpoondown 0, \forall j \in \mathcal{V}^i$;
5: $\Psi^j \leftharpoondown \{e, \forall e \in \mathcal{E}\}, \forall j \in \mathcal{V}^i$;
6: while $R^i < \Re^i$ do
7:   $\Psi^j \leftharpoondown \Psi^j - \{e, \forall e \in \Psi^j | c^{i,j} > C_e \vee m^{i,j} > M_e\}, \forall j \in \mathcal{V}^i$;
8:   $j' \leftharpoondown \arg\min_{j\in\mathcal{V}^i, \Psi^j != \emptyset} R^{i,j}$;
9:   if $j' = None$ then
10:    break;
11:  end if
12:  $e \leftharpoondown \arg\max_{e\in\Psi^{j'}} \overline{x}_e^{i,j'}$;
13:  $x_e^{i,j'} \leftharpoondown 1$;
14:  $\Psi^{j'} \leftharpoondown \Psi^{j'} - e$;
15:  Update reliability $R^i$, $R^{i,j}$, and update available resources $M_e$, $C_e$;
16: end while
17: if $R^i \ge \Re^i$ then
18:   for $j \in \mathcal{V}^i$ do
19:     $\mathcal{F} \leftharpoondown \{e', \forall e' \in \mathcal{E} | x_e^{i,j-1} = 1 \wedge x_{e'}^{i,j} = 1 \wedge B_{e,e'} \ge b^{i,j}, \forall (e, e') \in \mathcal{L}\}$;
20:     $y_e^{i,j} \leftharpoondown 1, e = \arg\max_{e\in\mathcal{F}} \overline{y}_e^{i,j}$;
21:     Update available resources $B_{e,e'}$;
22:   end for
23: end if
24: if $R^i < \Re^i$ or $\sum_{j\in\mathcal{V}^i} y_e^{i,j} < |\mathcal{V}^i|$ then
25:   $o^i \leftharpoondown 0$; $x_e^{i,j} \leftharpoondown 0$; $y_e^{i,j} \leftharpoondown 0, \forall j \in \mathcal{V}^i, e \in \mathcal{E}$;
26: end if
```

This lemma inspires us to add backups for the VNF in SFC with the lowest reliability (and available resources allow it to add additional backups), which can effectively improve SFC reliability. Its details are described in the next subsection.

3) Algorithm Description: The RuleDRL algorithm is shown in Algorithm 1, which is triggered by the arrival of a new SFC. First, we call DDPG to obtain a fractional solution and initialize decision variables and reliability. Then, we initialize the set of ESs that are candidates to add backups for each VNF. Moreover, we greedily add backups for the VNF with the lowest reliability on the candidate ES with the largest fractional solution. During this process, we exclude those candidate ESs that violate capacity constraints. This process terminates until the SFC reliability is satisfied or there is no candidate ESs for adding backups. Further, we select the VNF instance as the primary instance for forwarding traffic, which has sufficient bandwidth resources and the largest fractional solution. Finally, we reject the SFC request if the constraint is violated.

D. Algorithm Analysis

1) Computational Complexity: In RuleDRL algorithm, the computational complexity in the offline training process is proportional to the size of the training data and the training period. After offline training, we can leverage the trained model to perform online inference, i.e., determine how to back up, deploy and select the primary VNF instance for newly arrived SFC requests. Since the offline training process is run offline, we mainly focus on the computational complexity in the online running process [19], [21].

Assume that there are N hidden layers in the neural network in RuleDRL, each layer contains M neurons.

Theorem 3: In RuleDRL algorithm, the online process runs in $O(NL|\mathcal{E}|+N|\mathcal{L}|+MN^2+L|\mathcal{E}|^2+L^2|\mathcal{E}|)$.

Proof: As shown in Algorithm 1, we first call DDPG to obtain the fractional solution, and its complexity depends on the structure of the neural network. The state is fed into the input layer, which has size $L+5|\mathcal{L}|+2|\mathcal{E}|+1$. The action is output by the output layer, which has size $2L|\mathcal{E}|$. Therefore, this process runs in $O(N(L|\mathcal{E}|+|\mathcal{L}|+MN))$. Then, the algorithm initializes the parameters in $O(L|\mathcal{E}|)$. Further, in the worst case, the algorithm greedily runs in $O(L|\mathcal{E}|^2+L^2|\mathcal{E}|)$ for adding backups to the VNF with the least reliability. Besides, the algorithm selects the primary VNF in $O(L|\mathcal{E}|)$. Finally, the algorithm resets the parameters in $O(L|\mathcal{E}|)$ when the constraints cannot be satisfied. Based on the above analysis, RuleDRL runs in $O(NL|\mathcal{E}|+N|\mathcal{L}|+MN^2+L|\mathcal{E}|^2+L^2|\mathcal{E}|)$. □

Notably, the complexity of our algorithm is acceptable, which is further verified by simulation results.

2) Algorithm Approximation: Next, we establish the approximation ratio of our proposed algorithm in general cases.[8] To facilitate the theoretical analysis, we define the following variables,

$$\begin{aligned}&\xi=\min\{r_e,\forall e\in\mathcal{E}\},\eta=\min\{r^{i,j},\forall j\in\mathcal{V}^i\},\\&\Re_{\max}=\max\{\Re^i,\forall i\in\mathcal{I}\},\kappa_{\min}=\min\{\kappa_e,\forall e\in\mathcal{E}\},\\&\kappa_{\max}=\max\{\kappa_e,\forall e\in\mathcal{E}\},\zeta_{\min}=\min\{\zeta_e,\forall e\in\mathcal{E}\},\\&\zeta_{\max}=\max\{\zeta_e,\forall e\in\mathcal{E}\},\tau_{\min}=\min\{\tau_{e,e'},\forall(e,e')\in\mathcal{L}\},\\&\tau_{\max}=\max\{\tau_{e,e'},\forall(e,e')\in\mathcal{L}\}.\end{aligned}\tag{13}$$

In the following theorem, we analyze the performance of RuleDRL in the general case, where all SFC requests are accepted.[9]

Theorem 4: Assuming that all SFC requests are accepted, RuleDRL algorithm is a $\max\{\lfloor\log_{1-\xi\eta}(1-(\Re_{\max})^{\frac{1}{L}})+1\rfloor\frac{\kappa_{\max}}{\kappa_{\min}},\lfloor\log_{1-\xi\eta}(1-(\Re_{\max})^{\frac{1}{L}})+1\rfloor\frac{\zeta_{\max}}{\zeta_{\min}},\frac{\tau_{\max}}{\tau_{\min}}\}$-approximation algorithm.

[8]The approximation ratio of an algorithm refers to the ratio between the result obtained by the algorithm and the optimal cost or profit [51]. In this article, the approximate ratio refers to the ratio between our algorithm and the optimal solution in terms of the total resource cost.

[9]In the real-world scenario, the service provider is obliged to accept all SFC requests for turnover. Although the resources of each ES are limited, the resources of massive distributed ESs have sufficient resources to support SFC backup, deployment, and routing.

Proof: According to (1), (8), (9), (11), the resource cost of SFC S^i in the optimal solution satisfies

$$\begin{aligned}\tilde{\mathcal{C}}^i&\geq\kappa_{\min}\sum_{j\in\mathcal{V}^i}(c^{i,j}\sum_{e\in\mathcal{E}}\tilde{x}_e^{i,j})+\zeta_{\min}\sum_{j\in\mathcal{V}^i}(m^{i,j}\sum_{e\in\mathcal{E}}\tilde{x}_e^{i,j})\\&\quad+\tau_{\min}\sum_{j\in\mathcal{V}^i}\sum_{e,e'\in\mathcal{L}}b^{i,j}\tilde{z}_{e,e'}^{i,j}\\&\geq\kappa_{\min}\sum_{j\in\mathcal{V}^i}c^{i,j}+\zeta_{\min}\sum_{j\in\mathcal{V}^i}m^{i,j}+\tau_{\min}\sum_{j\in\mathcal{V}^i}b^{i,j}.\end{aligned}\tag{14}$$

where $\tilde{x}_e^{i,j}$, $\tilde{z}_{e,e'}^{i,j}$ denote the optimal decision.

Since the accepted SFC request satisfies the constraints, based on (9), (11), the resource cost of SFC S^i in RuleDRL satisfies

$$\begin{aligned}\mathcal{C}^i&\leq\kappa_{\max}\sum_{j\in\mathcal{V}^i}(c^{i,j}\sum_{e\in\mathcal{E}}x_e^{i,j})+\zeta_{\max}\sum_{j\in\mathcal{V}^i}(m^{i,j}\sum_{e\in\mathcal{E}}x_e^{i,j})\\&\quad+\tau_{\max}\sum_{j\in\mathcal{V}^i}b^{i,j}.\end{aligned}\tag{15}$$

The reliability of the SFC can be improved by adding backups to its VNFs. Intuitively, a limited number of instances are needed to satisfy the specified SFC reliability. That is, $\sum_{e\in\mathcal{E}}x_e^{i,j}$ is bounded, it is proved as follows.

We first declare that once $\min_{j\in\mathcal{V}^i}\{R^{i,j}\}\geq\Re^i)^{\frac{1}{|\mathcal{V}^i|}}$, then the SFC reliability is satisfied ($R^i\geq\Re^i$). This can be formally proved as follows.

Based on (2), we have

$$\begin{aligned}R^i&=\prod_{j\in\mathcal{V}^i}R^{i,j}\geq\prod_{j\in\mathcal{V}^i}\min_{j'\in\mathcal{V}^i}R^{i,j'}\\&\geq\prod_{v\in\mathcal{V}^i}(\Re^i)^{\frac{1}{|\mathcal{V}^i|}}=\Re^i,j\in\mathcal{V}^i.\end{aligned}\tag{16}$$

Define $\hat{R}^{i,j}$ as the reliability of VNF $v^{i,j}$ before its last instance is added. We also define $\hat{x}_e^{i,j}$ as the backup and deployment decision for VNF $v^{i,j}$ before its last instance is added. Then, we have

$$\sum_{e\in\mathcal{E}}\hat{x}_e^{i,j}+1=\sum_{e\in\mathcal{E}}x_e^{i,j},\forall i\in\mathcal{I},j\in\mathcal{V}^i.\tag{17}$$

In RSBD algorithm, we iteratively add backups to the VNF with the lowest reliability until the SFC reliability is satisfied. Obviously, once a VNF needs to be additionally added instances, its reliability is lower than $(\Re^i)^{\frac{1}{|\mathcal{V}^i|}}$. Then, we have

$$\hat{R}^{i,j}<(\Re^i)^{\frac{1}{|\mathcal{V}^i|}}<(\Re_{\max})^{\frac{1}{|\mathcal{V}^i|}},\forall i\in\mathcal{I},j\in\mathcal{V}^i.\tag{18}$$

Based on (3), (4), we have

$$\begin{aligned}\hat{R}^{i,j}&=1-\prod_{e\in\mathcal{E}}(1-r^{i,j}r_e\hat{x}_e^{i,j})\geq1-\prod_{e\in\mathcal{E}}(1-\xi\eta\hat{x}_e^{i,j})\\&=1-\prod_{e\in\mathcal{E}}(1-\xi\eta)^{\hat{x}_e^{i,j}}=1-(1-\xi\eta)^{\sum_{e\in\mathcal{E}}\hat{x}_e^{i,j}},\end{aligned}\tag{19}$$

where the third equation is obtained because the value of $\hat{x}_e^{i,j} \in \{0,1\}$.

Further, by taking the logarithm of (19), we have

$$\sum_{e\in\mathcal{E}} \hat{x}_e^{i,j} \le \log_{1-\xi\eta}(1-\hat{R}^{i,j}) \le \log_{1-\xi\eta}\left(1-(\Re^i)^{\frac{1}{|\mathcal{V}^i|}}\right), \quad (20)$$

where the second inequality is obtained by (18).

Obviously, based on (17), (20), we have

$$\sum_{e\in\mathcal{E}} x_e^{i,j} = \sum_{e\in\mathcal{E}} \hat{x}_e^{i,j} + 1 \le \log_{1-\xi\eta}\left(1-(\Re_{\max})^{\frac{1}{|\mathcal{V}^i|}}\right) + 1. \quad (21)$$

Since $x_e^{i,j} \in \{0,1\}$, $\sum_{e\in\mathcal{E}} x_e^{i,j}$ is also an integer. Then,

$$\sum_{e\in\mathcal{E}} x_e^{i,j} \le \left\lfloor \log_{1-\xi\eta}\left(1-(\Re_{\max})^{\frac{1}{|\mathcal{V}^i|}}\right)\right\rfloor + 1$$
$$\le \left\lfloor \log_{1-\xi\eta}\left(1-(\Re_{\max})^{\frac{1}{L}}\right)\right\rfloor + 1. \quad (22)$$

Then, based on (15), (22), we have

$$\mathcal{C}^i \le \left(\lfloor \log_{1-\xi\eta}(1-(\Re_{\max})^{\frac{1}{L}})\rfloor + 1\right)\cdot\left(\kappa_{\max}\sum_{j\in\mathcal{V}^i} c^{i,j} + \zeta_{\max}\sum_{j\in\mathcal{V}^i} m^{i,j}\right) + \tau_{\max}\sum_{j\in\mathcal{V}^i} b^{i,j}. \quad (23)$$

Finally, the approximate ratio of RuleDRL algorithm is

$$\alpha = \frac{\sum_{i\in\mathcal{I}}\mathcal{C}^i}{\sum_{i\in\mathcal{I}}\tilde{\mathcal{C}}^i} \le \max\left\{\lfloor \log_{1-\xi\eta}\left(1-(\Re_{\max})^{\frac{1}{L}}\right)+1\rfloor\frac{\kappa_{\max}}{\kappa_{\min}}, \lfloor \log_{1-\xi\eta}(1-(\Re_{\max})^{\frac{1}{L}})+1\rfloor\frac{\zeta_{\max}}{\zeta_{\min}}, \frac{\tau_{\max}}{\tau_{\min}}\right\}.$$

□

Specially, we discuss the approximate performance of RuleDRL algorithm in realistic cases.

Corollary 4.1: RuleDRL algorithm is a $\max\{5\frac{\kappa_{\max}}{\kappa_{\min}}, 5\frac{\zeta_{\max}}{\zeta_{\min}}, \frac{\tau_{\max}}{\tau_{\min}}\}$-approximate algorithm in the realistic case, where $\Re_{\max} \le 0.999999$, $\eta \ge 0.99, \xi \ge 0.99, L \le 7$.[10]

Proof: In the above realistic case, we have

$$\lfloor \log_{1-\xi\eta}(1-(\Re_{\max})^{\frac{1}{L}})\rfloor \le \lfloor \log_{1-0.99\cdot 0.99}(1-0.999999^{\frac{1}{7}})\rfloor$$
$$= \lfloor 4.03\rfloor = 4. \quad (24)$$

Then, based on Theorem 4, the approximate ratio satisfies

$$\alpha \le \max\left\{5\frac{\kappa_{\max}}{\kappa_{\min}}, 5\frac{\zeta_{\max}}{\zeta_{\min}}, \frac{\tau_{\max}}{\tau_{\min}}\right\}. \quad (25)$$

□

[10]As shown in [8], the reliability requirement of mission-critical services is distributed in [0.99-0.999999], where $\Re_{\max} \le 0.999999$. Besides, ES hardware downtime is about 29 hours per year [38], implying reliability of 0.9967, which is clearly in line with $\xi \ge 0.99$. Moreover, the reliability of service software is distributed in [0.99041-0.9999], which is also consistent with $\eta \ge 0.99$). Finally, the typical SFC length is less than 7, that is, $L \le 7$.

In real-world scenarios, the cost differences per unit of computing, memory, and bandwidth resources are bounded by constants [30], [31], [32]. That is to say, $\frac{\kappa_{\max}}{\kappa_{\min}} = \epsilon_1$, $\frac{\zeta_{\max}}{\zeta_{\min}} = \epsilon_2$, $\frac{\tau_{\max}}{\tau_{\min}} = \epsilon_3$, where ϵ_1, ϵ_2, ϵ_3 are constants. In this case, $\alpha = \max\{5\epsilon_1, 5\epsilon_2, \epsilon_3\}$. That is, in real-world scenarios, our algorithm has a constant approximation ratio.

We further consider a special case where ESs are isomorphic in computation cost, memory cost, and links are isomorphic in bandwidth cost. That is, $\kappa_e = \kappa_{e'}, \forall e \in \mathcal{E}, e' \in \mathcal{E}$; $\zeta_e = \zeta_e, \forall e \in \mathcal{E}, e' \in \mathcal{E}$; $\tau_{e,e'} = \tau_{e'',e'''}, \forall (e,e') \in \mathcal{L}, (e'',e''') \in \mathcal{L}$. In this case, $\kappa_{\min} = \kappa_{\min}$, $\zeta_{\max} = \zeta_{\min}$, $\tau_{\max} = \tau_{\min}$. Based on Corollary 4.1, we have $\alpha \le \max\{5,5,1\} = 5$. This means that the resource cost of our algorithm does not exceed 5 times the optimal solution. In the evaluation, we further verify with the results that our algorithm can well approach the optimal solution.

V. Evaluation

In this section, we first present the evaluation settings, then analyze and discuss the simulation results.

A. Settings

Evaluation Environment: We implement a Python-based simulator and use PyTorch to build the machine learning framework. All simulations are conducted on a computer with 40-Core Intel(R) Xeon(R) CPU @ 2.40 GHz, 125G RAM, and an NVIDIA Geforce RTX 3090 GPU.

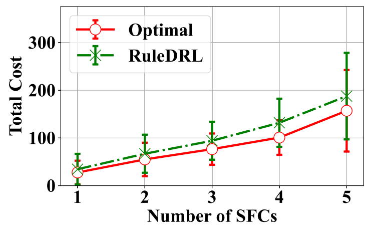
Service Function Chain: The SFC consists of 1-7 VNFs [49]. The computational, memory, and bandwidth resource costs required by each VNF are derived from the Alibaba *cluster-trace-v2018* trace data [52]. Besides, the software reliability of VNFs is distributed in [0.999,0.99999] [7]. Moreover, the reliability requirements of SFCs are distributed in [0.99, 0.999,0.9999, 0.99999, 0.999999] [8].

NFVI: In NFVI, there are 30 ESs, which are interconnected by overlay links. The bandwidth capacity of each link, as well as the computing resource and memory resource capacity of each ES are also derived from the Alibaba *cluster-trace-v2018* trace data. The hardware reliability is distributed in $[0.999 - 0.99999]$, which is obtained from typical infrastructure providers [53]. The cost per unit of computing and memory resources on each ES is distributed between 1-10 units, and the cost of bandwidth resources on each link is distributed between 1-10 units [30], [31], [32].

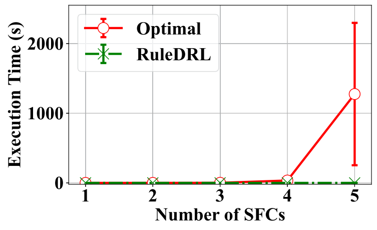
Parameters: Each neural network has three layers, and each hidden layer contains 1024 neurons [54]. The learning rates for Actor and Critic are 0.001 and 0.003 [55], respectively. The discount factor is 0.99. The penalty parameter $\mathcal{P}$ is set as 1000, which is based on the resource cost. The above parameters are adopted as default settings unless otherwise specified. All the data points are collected from 100 runs.

Algorithms Comparison: We evaluate RuleDRL with the following algorithms.

- *Rule-Based:* This is a simplified version of our algorithm; the rule-based scheme is used to add backups to the VNFs with the lowest reliability and deploy these instances to



(a) Total cost



(b) Execution time

Fig. 4. Compared with the optimal solution.

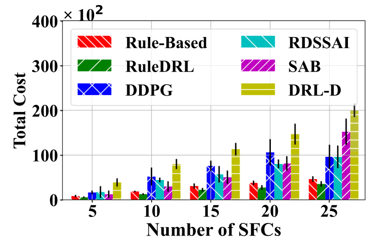
the ESs with the greatest reliability improvement, with no capacity violations.

- *DDPG:* This algorithm is a simplified version of our algorithm, where DDPG [24] outputs an integer solution.
- *RDSSAI:* This is an improved version of RDSSA [23], where DRL is used to make deployment decisions.
- *SAB [13]:* The algorithm first provides a static backup for each VNF in the SFC, and dynamically adds backups for failed VNFs.
- *DRL-D [22]:* The algorithm utilizes a DRL-based approach to deploy each VNF in the SFC without violating capacity constraints in a resource-efficient manner.
- *Optimal solution:* The optimal solution is obtained by exploring the solution space via the branch and bound method.

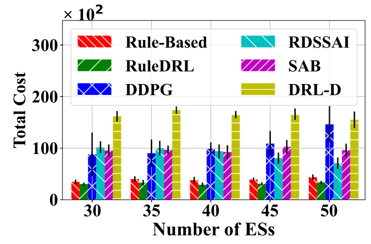
B. Simulation Result

1) Optimality: We evaluated the optimality of our algorithm by comparing it with the optimal solution in total cost and execution time under different numbers of SFCs. Since solving the optimal solution is time-consuming, we set $|\mathcal{E}| = 5, |\mathcal{V}^i| = 1, |\mathcal{I}| \in [1, 5]$, which is a small scale special case. The simulation results are shown in Fig. 4.

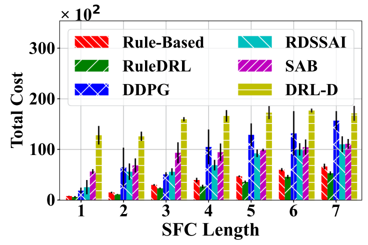
As shown in Fig. 4(a), the total cost of RuleDRL is close to the optimal solution. For example, when there are 5 SFCs, the total cost of RuleDRL is 187.72, and the optimal solution is 157.08. That is, our algorithm is 16.31% away from the optimal solution. It should be noted that the optimal solution is obtained by exploring the solution space, which has known information about all future SFCs. However, in the real environment, information about the future SFC is difficult to obtain accurately. Therefore, we believe that RuleDRL is good in optimality. Besides, solving the optimal solution exhibits exponential growth as the number of SFC requests increases, as shown in Fig. 4(b). This is because as the number of SFC requests increases, the solution space



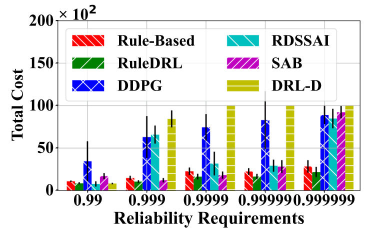
(a) Different numbers of SFCs



(b) Different numbers of ESs



(c) Different SFC lengths



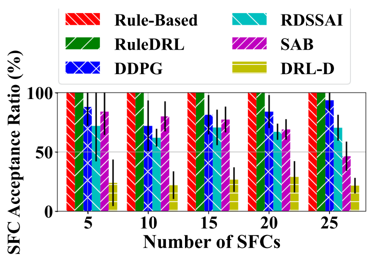
(d) Different reliability requirements

Fig. 5. Compared with other solutions in total cost.

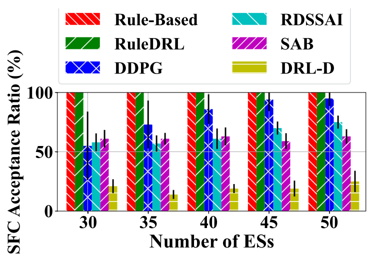
grows exponentially, and exploring optimal solutions in such a large solution space is extremely time-consuming. When there are 5 SFCs, it takes 1275.5 s to explore the optimal solution, while RuleDRL only takes 5.8 ms. It should be noted that we do not provide the optimal solution in the following simulations because it is extremely time-consuming.[11] The above results verify that our algorithm has good optimality and is significantly better than the optimal solution in terms of execution time.

2) Superiority: We evaluate the superiority of our algorithm by comparing it with existing works in terms of total cost and SFC acceptance ratio, and the results are shown in Figs. 5 and 6. As shown in Fig. 5(a), our algorithm consistently outperforms other algorithms under different numbers of SFCs. For

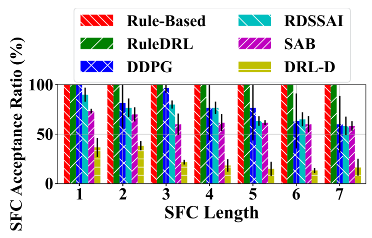
[11] It should be noted that the solution space also exhibits exponential growth as the SFC length and the number of ESs increase. In the following evaluation, $|\mathcal{E}| \geq 30, |\mathcal{V}^i| \in [1, 7], |\mathcal{I}| \geq 5$, which leads to unacceptable time to solve the optimal solution even when $|\mathcal{I}| = 5$ (no optimal solution was obtained after 10 hours of execution).



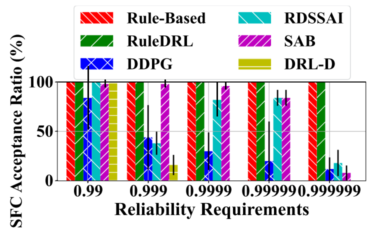
(a) Different number of SFCs



(b) Different number of ESs



(c) Different SFC lengths



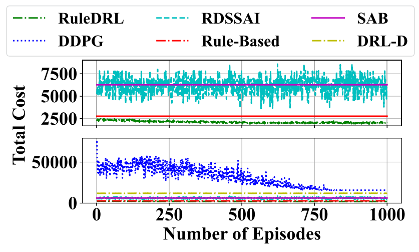
(d) Different reliability requirements

Fig. 6. Compared with other solutions in SFC acceptance ratio.

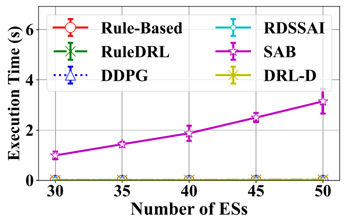
example, when there are 25 SFCs, the total cost of Rule-Based, RuleDRL, DDPG, RDSSAI, SAB, and DRL-D are 3806, 2756, 10612, 8005, 8127, and 14689, respectively. This means that our algorithm can save the total cost by 27.589%, 74.03%, 65.67%, 66.09%, and 81.24%, compared with Rule-Based, DDPG, RDSSAI, SAB, and DRL-D. That is, our algorithm can save the total cost by up to 65.67% compared to existing solutions. The reason is that compared with Rule-Based, RuleDRL can capture delayed rewards and has a stronger ability to extract the rules behind RSFCP problem. Compared with DDPG, RDSSAI, SAB, and DRL-D, RuleDRL can better cater to constraints and avoid punishment caused by violating constraints.[12] Besides, DRL-D does not provide any backups, which leads to frequent violations of reliability requirements and the highest total cost due to penalties. And SAB provides a static backup for each VNF in each SFC, resulting in resource waste for SFCs with low-reliability requirements. Moreover, the total cost of these algorithms increases with the number of SFC requests since more SFCs lead to more resource consumption. Similar results can be found in Fig. 5(b), (c), and (d), where our algorithm always outperforms other solutions in terms of total cost under different numbers of ESs, SFC length and reliability requirements. As shown in Fig. 5(b), with the increase in the number of ESs, the total cost of DDPG gradually increases, as random exploration in DDPG leads to more backups to be deployed. As shown in Fig. 5(c), as the length of SFC increases, the total cost of our algorithm gradually increases, which is due to more VNFs that need to be deployed, resulting in more resource consumption. As shown in Fig. 5(d), as the reliability requirements of SFC increase, the total cost of our algorithm gradually increases, which is because more backups need to be deployed to meet the SFC reliability requirement, resulting in more resource costs. The above results verify that our algorithm significantly outperforms existing algorithms in total cost.

As shown in Fig. 6(a), RuleDRL and Rule-Based can always accept all requests, while DDPG, RDSSAI, SAB, and DRL-D may reject requests frequently. For example, when there are 25 SFCs, the SFC acceptance ratios of DDPG, RDSSAI, SAB, and DRL-D are only 93.6%, 70.4%, 46.4%, and 21.6%. This is because, in RDSSAI and DDPG, the DRL agent randomly explores the solution space, which contains massive solutions that violate constraints, including SFC completeness, capacity, and reliability constraints. Frequent constraint violations lead to frequent SFC requests to be rejected. In Rule-Based and RuleDRL, constraints are carefully checked before making decisions, and they exploit rules to effectively prevent constraints from being violated. Besides, SAB provides a static backup for each VNF and dynamically increases the backup when the VNF instance fails. This is sufficient for requests with low-reliability requirements but not for those with high-reliability requirements. When an SFC request arrives, and both the primary and backup instances of one of its VNFs fail, but the dynamically increased backup has not yet been activated, it fails. Similar results can be found in Fig. 6(b), (c), and (d), where our algorithm always outperforms other solutions in terms of SFC acceptance ratio under different number of ESs, SFC length and reliability requirements. As shown in Fig. 6(d), RuleDRL and Rule-Based can still accept all SFC requests under different reliability requirements. This is because our algorithm dynamically allocates more backups for SFC requests when its reliability requirements are higher. When the SFC reliability requirement is 0.999999, RuleDRL can improve the SFC acceptance ratio by 88%, 82%, 92%, and 100% compared to DDPG, RDSSAI, SAB, and DRL-D. Besides, the SFC acceptance ratios of DDPG, RDSSAI, SAB, and DRL-D gradually decreased as the reliability requirements increased. This is because software reliability is ignored in RDSSAI, which leads to more and more SFC requests being rejected due to reliability violations when SFC reliability requirements become higher. Besides, in DDPG, higher reliability requirements lead to more solutions in the solution space violating reliability constraints, which in turn leads to random exploration exploring more solutions violating the constraints. Moreover, SAB and

[12]It should be noted that once the requirements of an SFC request are violated, it will be rejected, resulting in lost revenue and penalty costs.

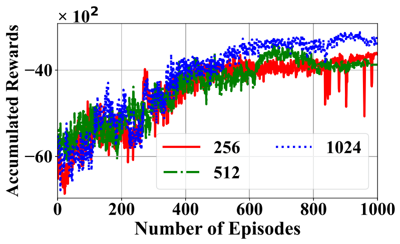


(a) Convergence

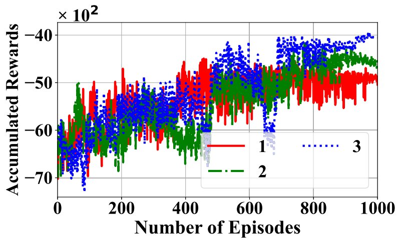


(b) Execution time

Fig. 7. Compared to other solutions in convergence and execution time.



(a) Varying hidden layer width



(b) Varying hidden layer depth

Fig. 8. Accumulated rewards under different parameter settings.

DRL-D fail to adapt to the differentiated reliability requirements of SFCs, which leads to a decrease in their SFC acceptance ratio as the reliability requirements become higher. The above results verify that our algorithm is also superior to the existing algorithms in SFC acceptance ratio.

3) Practicality: We evaluate the practicality of RuleDRL in convergence and execution time during training and inference, respectively, and the results are shown in Fig. 7. As shown in Fig. 7(a), the figure above is a partial expansion of the figure below. RuleDRL consistently outperforms the other five algorithms, as guided by the rule-based scheme, starting with a high-quality solution and converging to a better solution. This benefits from the rule-based scheme, which prevents DRL from exploring low-quality solutions. Besides, the total cost of RDSSAI is jittery with increasing training episodes due to frequent penalties introduced by constraint violations. Moreover, the total cost of DDPG is significantly higher than the other algorithms, since its random initialization may produce a decision with a large number of unnecessary, redundant backups, which is quite costly. Besides, it gradually converges to a poor local optimum since it needs to explore an extremely large solution space, which is inefficient and prone to getting stuck in a poor local optimum. Moreover, the total cost of DRL-D does not change significantly, since it does not provide backup for any VNF in each SFC, resulting in reliability violations and stable penalties. As shown in Fig. 7(b), the execution time of SAB is significantly longer than other algorithms, while our algorithm has a similar execution time with RDSSAI, Rule-Based, DDPG, and DRL-D. This is because SAB needs to utilize an optimizer to solve a linear programming, which is time-consuming. When there are 50 ESs, SAB algorithm requires 3.15 s to obtain a service provision decision for an SFC request, while our algorithm only requires 0.029 s, which is acceptable for service provision. The above results verify that our algorithm has good practicality in convergence and execution time.

4) Robustness: We evaluate the performance of RuleDRL under different parameter settings, as shown in Fig. 8. As shown in Fig. 8(a), RuleDRL tends to converge to a larger reward under different hidden layer neuron settings. Besides, RuleDRL with more hidden layer neurons tends to converge faster because larger neural networks have a stronger fitting ability. As shown in Fig. 8(b), RuleDRL still converges to a larger reward under the different number of hidden layer settings. Moreover, RuleDRL with more neural network layers tends to converge to larger rewards, also due to the stronger fitting ability of larger neural networks. Despite the above advantages, larger neural networks result in longer training and inference times and may underfit insufficient training sets. The above results show that our algorithm can converge with different parameters over the training epochs, showing good robustness.

VI. Conclusion

Achieving highly reliable SFCs in dynamic edge environments is challenging due to the delayed rewards brought by future SFCs, limited infrastructure resources, and heterogeneous hardware and software reliability. To capture these challenges, we formulate the reliable SFC provision problem as a nonlinear integer programming problem, prove its NP-hardness, and reveal its pain points. To address this problem, we propose a RuleDRL algorithm combining the advantages of both DRL and rule-based schemes, where DRL is used to capture delayed rewards in dynamic environments. The rule-based method is based on our findings on the formulation of SFC reliability and explores cost-efficient solutions to satisfy all constraints. This algorithm is rigorously proved to output solutions with bounded approximation guarantees in general cases. Extensive trace-driven simulations verify that our algorithm significantly outperforms the state-of-the-art solution in terms of resource cost and SFC acceptance ratio.

References

[1] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing–A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[2] S. Yang, F. Li, R. Yahyapour, and X. Fu, "Delay-sensitive and availability-aware virtual network function scheduling for NFV," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 188–201, Jan./Feb. 2022.

[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tut.*, vol. 18, no. 1, pp. 236–262, First Quarter 2016.

[4] J. Pei, P. Hong, K. Xue, and D. Li, "Resource aware routing for service function chains in SDN and NFV-enabled network," *IEEE Trans. Serv. Comput.*, vol. 14, no. 4, pp. 985–997, Jul./Aug. 2021.

[5] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in NFV: Models and algorithms," *IEEE Trans. Serv. Comput.*, vol. 10, no. 4, pp. 534–546, Jul./Aug. 2017.

[6] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[7] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: A field study of middlebox failures in datacenters," in *Proc. ACM Internet Meas. Conf.*, 2013, pp. 9–22.

[8] 3GPP, "Service requirements for the 5G system (release 18)," Tech. Specification Group Serv. Syst. Aspects, Tech. Rep. TS 22.261, 2021. [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/22series/22.261/22261-i11.zip

[9] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "RABA: Resource-aware backup allocation for a chain of virtual network functions," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1918–1926.

[10] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[11] W. Liang, Y. Ma, W. Xu, Z. Xu, X. Jia, and W. Zhou, "Request reliability augmentation with service function chain requirements in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4541–4554, Dec. 2022.

[12] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2096–2105.

[13] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2994–3008, Aug. 2022.

[14] M. A. Jamshed et al., "Kargus: A Highly-scalable Software-based Intrusion Detection System," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 317–328.

[15] G. Papathanail, A. Pentelas, and P. Papadimitriou, "Towards fine-grained resource allocation in NFV infrastructures," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.

[16] A. Gupta, M. F. Habib, U. Mandal, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service-chaining strategies using virtual network functions in operator networks," *Comput. Netw.*, vol. 133, pp. 1–16, 2018.

[17] D. Gedia and L. Perigo, "Performance evaluation of SDN-VNF in virtual machine and container," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2018, pp. 1–7.

[18] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[19] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 263–278, Feb. 2020.

[20] J. S. P. 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, Feb. 2020.

[21] J. Pei, P. Hong, K. Xue, D. Li, D. S. Wei, and F. Wu, "Two-phase virtual network function selection and chaining algorithm based on deep learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1102–1117, Jun. 2020.

[22] Q. Fan, P. Pan, X. Li, S. Wang, J. Li, and J. Wen, "DRL-D: Revenue-aware online service function chain deployment via deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4531–4545, Dec. 2022.

[23] J. Jia, L. Yang, and J. Cao, "Reliability-aware dynamic service chain scheduling in 5G networks based on reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[24] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1–14.

[25] J. Li, W. Liang, M. Huang, and X. Jia, "Providing reliability-aware virtualized network function services for mobile edge computing," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 732–741.

[26] J. Li, W. Liang, M. Huang, and X. Jia, "Reliability-aware network service provisioning in mobile edge-cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1545–1558, Jul. 2020.

[27] 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 Glob. Commun. Conf.*, 2019, pp. 1–6.

[28] NFV, GS and others, "Network functions virtualisation (NFV); Architectural framework," *Ind. Specification Group Netw. Functions Virtualization*, vol. 2, no. 2, 2013, Art. no. V1.

[29] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[30] Compute pricing, 2022. [Online]. Available: https://www.oracle.com/cloud/compute/pricing.html

[31] Memory Pricing, 2022. [Online]. Available: https://cloud.ibm.com/docs/ComposeForElasticsearch?topic=ComposeForElasticsearch-pricing

[32] Bandwidth Pricing, 2022. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/bandwidth/

[33] P. Berde et al., "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Top. Softw. Defined Netw.*, 2014, pp. 1–6.

[34] N. Katta, H. Zhang, M. Freedman, and J. Rexford, "Ravana: Controller Fault-Tolerance in Software-Defined Networking," in *Proc. 1st ACM SIGCOMM Softw. Defined Netw. Res.*, 2015, pp. 1–12.

[35] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, and S. Shenker, "SCL: Simplifying distributed SDN control planes," in *Proc. USENIX Conf. Netw. Syst. Des. Implementation*, 2017, pp. 329–345.

[36] 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, Nov. 2020.

[37] H. Huang et al., "Scalable orchestration of service function chains in NFV-enabled networks: A federated reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2558–2571, Aug. 2021.

[38] J. Covitz, "Resilience at edge computing sites is resilience for the whole IT environment," 2019. [Online]. Available: https://www.networkworld.com/article/3356439/resilience-at-edge-computing-sites-is-resilience-for-the-whole-it-environment.html

[39] Reliability definition, 2022. [Online]. Available: https://reliabilityweb.com/en/articles/what-is-reliability

[40] 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, Feb. 2020.

[41] J. Li et al., "Budget-aware user satisfaction maximization on service provisioning in mobile edge computing," *IEEE Trans. Mobile Comput.*, early access, Sep. 9, 2022, doi: 10.1109/TMC.2022.3205427.

[42] J. Meza, T. Xu, K. Veeraraghavan, and O. Mutlu, "A large scale study of data center network reliability," in *Proc. Internet Meas. Conf.*, 2018, pp. 393–407.

[43] X. Jin et al., "Dynamic scheduling of network updates," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 539–550, 2014.

[44] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[45] Knapsack Problem, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Knapsack_problem#0-1_knapsack_problem

[46] D. Pisinger and P. Toth, "Knapsack Problems," in *Handbook of Combinatorial Optimization*. Berlin, Germany: Springer, 1998, pp. 299–428.

[47] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.

[48] H. Q. Ngo, "Approximation algorithms based on LP relaxation," 2005. [Online]. Available: https://cse.buffalo.edu/∼hungngo/classes/2005/594/notes/relaxation-rounding.pdf

[49] W. Haeffner, J. Napper, M. Stiemerling, D. Lopez, and J. Uttaro, “Service function chaining use cases in mobile networks,” Internet Eng. Task Force, Tech. Rep. draft-ietf-sfc-use-case-mobility-09, 2019. [Online]. Available: https://datatracker.ietf.org/doc/pdf/draft-ietf-sfc-use-case-mobility-09
[50] Y. Kim, J. Park, D.-H. Kwon, and H. Lim, “Buffer management of virtualized network slices for quality-of-service satisfaction,” in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2018, pp. 1–4.
[51] Approximation Algorithms, 2023. [Online]. Available: https://www.cs.yale.edu/homes/aspnes/pinewiki/ApproximationAlgorithms.html?highlight=%8CategoryAlgorithmNotes%29
[52] K. W. J. Guo, Z. Chang, and G. Xu, “Cluster-trace-V2018,” 2022. [Online]. Available: https://github.com/alibaba/clusterdata
[53] C. Cérin et al., “Downtime statistics of current cloud solutions,” Int. Work. Group Cloud Comput. Resiliency, Tech. Rep. iwgcr2014, 2014. [Online]. Available: https://lipn.univ-paris13.fr/∼coti/papiers/iwgcr2014.pdf
[54] H. Zhu, V. Gupta, S. S. Ahuja, Y. Tian, Y. Zhang, and X. Jin, “Network planning with deep reinforcement learning,” in *Proc. ACM SIGCOMM Conf.*, 2021, pp. 258–271.
[55] 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, Feb. 2020.

Yue Zeng received the MS degree from the department of electronic information engineering from Southwest University, Chongqing, China, in 2019. He is currently working toward the PhD degree with the department of computer science and technology in Nanjing University, China. His research interests include network functions virtualization, software defined networking, machine learning for networking, distributed computing, and edge computing.

Zhihao Qu (Member, IEEE) received the BS and PhD degree in computer science from Nanjing University, Nanjing, China, in 2009, and 2018, respectively. He is currently an associate professor with the College of Computer and Information at Hohai University. His research interests are mainly in the areas of wireless networks, edge computing, and distributed machine learning.

Song Guo (Fellow, IEEE) is currently a full professor with the Department of Computing, The Hong Kong Polytechnic University. He is also the Changjiang chair professor awarded by the Ministry of Education of China. His research interests include edge AI, mobile computing, and distributed systems. He has been recognized as a highly cited researcher (Web of Science) and was the recipient of more than 14 best paper awards from IEEE/ACM conferences, journals, and technical committees. He is the editor-in-chief of the IEEE Open Journal of the Computer Society. He was on the IEEE Communications Society Board of Governors, IEEE Computer Society Fellow Evaluation Committee, and editorial board of a number of prestigious international journals, including *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, and *IEEE Internet of Things Journal*.

Bin Tang (Member, IEEE) received the BS and PhD degrees in computer science from Nanjing University, Nanjing, China, in 2007, and 2014, respectively. He was an assistant researcher with Nanjing University from 2014 to 2020, and also a research fellow with The Hong Kong Polytechnic University in 2019. He is currently a professor with Hohai University. His research interests lie in the area of communications, network coding, and distributed computing.

Baoliu Ye (Member, IEEE) received the PhD degree in computer science from Nanjing University, China in 2004. He is a full professor with the Department of Computer Science and Technology, Nanjing University. He served as a visiting researcher of the University of Aizu, Japan from 2005 to 2006, and the dean of School of Computer and Information, Hohai University since 2018. His current research interests mainly include distributed systems, cloud computing, wireless networks with more than 70 papers published in major conferences and journals. He served as the TPC co-chair of HotPOST12, Hot-POST11, P2PNet10. He is the regent of CCF, the secretary-general of CCF Technical Committee of Distributed Computing and Systems.

Jing Li (Member, IEEE) received the BSc and PhD degrees with the first class Honours from The Australian National University in 2022 and 2018, respectively. He is currently a postdoctoral fellow with The Hong Kong Polytechnic University. His research interests include mobile edge computing, Internet of Things, network function virtualization, and combinatorial optimization. He has published papers in top journals and conferences such as *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, *ACM Transactions on Sensor Networks*, and IEEE ICDCS.

Jie Zhang (Member, IEEE) is currently working toward the PhD degree with the Department of Computing, The Hong Kong Polytechnic University. Her current research interests include edge computing, federated learning, and deep reinforcement learning.