Proactive VNF Redeployment and Traffic Routing for modern telco networks

Qiong Liu*, Tianzhu Zhang†‖, Walter Cerroni‡, Leonardo Linguaglossa*
*LTCI, Télécom Paris, Institut Polytechnique de Paris, 91120, Palaiseau, France
†Nokia Bell Labs, 91300, Massy, France
‡University of Bologna, Bologna, Italy

***Abstract*—The last decade has witnessed the rise of Network Function Virtualization (NFV). Despite its benefits, resource allocation and traffic scheduling are still challenging. A practical issue is where to place Virtual Network Functions (VNFs) in the network to sustain long-term optimal objectives and when to reallocate resources given the dynamics of the substrate network. Most prior works either consider static settings or work in reactive fashions. This paper proposes a dual-window algorithm for proactive service redeployment and traffic routing. Specifically, our algorithm employs an entropy measure to gauge the uncertainty in the substrate network and cognitively updates the service redeployment interval to avoid unnecessary data collection overhead. Our algorithm is lightweight, intuitive, requires no offline training, and achieves the best overall effectiveness and efficiency compared with three state-of-the-art solutions.**

***Index Terms*—Network Function Virtualization, Entropy, VNF placement, Service Deployment**

I. Introduction

In recent years, modern telco networks have undergone tremendous transformations with the emergence of Network softwarization, which embodies a large assemblage of concepts and technologies to drive traditional hardware-based network architectures toward software-based ones. One of the pivotal components is Network Function Virtualization (NFV). Unlike traditional networks that heavily rely on high-end, proprietary middleboxes, NFV decouples network functions from dedicated hardware and deploys them as Virtual Network Functions (VNFs) on commodity hardware. This paradigm transition enables agile and scalable service provisioning [1].

Despite the multitudinous benefits, applying NFV in practice still faces many problems, especially the deployment & management of network services and resources. One of the most challenging problems is VNF Placement and Traffic Routing (VPTR), which involves finding the optimal VNF placement and traffic routing schemes without violating the service and resource constraints [2]. As network traffic must traverse the VNFs in a specific order, which is implemented as Service Function Chains (SFCs), VPTR is inherently a more daunting undertaking than the traditional service placement problems [3]–[5]. The ever-increasing scale and complexity of modern telco infrastructures further compound the situation. Over the last decade, many research endeavors aim to address this problem [2], [6]. Although these solutions can optimize resource allocation in static settings, most struggle to sustain similar performance in dynamic environments, where the network and resource status continuously evolve [7]. Contemporary ISP and data center networks are permeated with intermittent evolvements and uncertainties, such as traffic fluctuation, performance contention, and component failure. The software nature of the NFV paradigm only expands the sphere of uncertainty. Some solutions aim to implement the dynamic SFC provisioning, defined as the VNF Redeployment and Traffic Routing (VRTR) problem. Still, their approaches are mostly reactive and can only be triggered by specific network conditions/events. Consequently, they fail to enable proactive adjustments, subjecting them to high setup latency and Quality of Service (QoS) degradation [8]. In essence, there is an urgent need for a time-aware, proactive resource allocation method capable of capturing network uncertainties and delivering long-term, high-quality services.

This work leverages the concept of *entropy* to capture network uncertainties and address the VRTR problem. Although entropy-based resource scheduling has shown great promise in point-to-point communication (e.g. [9]) and ad-hoc networks (e.g. [10]), its value is severely underappreciated in NFV domain [11]. We aim to optimize the service acceptance ratio by proactively adjusting the VNF placement and traffic routing schemes based on the measured network uncertainty. The contributions of this work are as follows:

- We formulate the VPTR problem with a compact matrix representation, which can be efficiently solved, even in large-scale networks with high traffic loads.
- We further model the VRTR problem based on VPTR. We define an entropy metric to inspect network stability and propose a novel dual-window algorithm to address the VRTR problem proactively.
- Our algorithm is evaluated against three state-of-the-art solutions, and It achieves the best overall performance in maximizing the service acceptance ratio while presenting the highest computation efficiency.

The rest of the paper is organized as follows: Sec. II and Sec. III present the related works and our system model, respectively. We formulate the VPTR and VRTR problems in Sec. IV.

‖ Corresponding Author

Then, we present our entropy-based algorithm to address the VRTR problem in Sec. V and evaluate its effectiveness in Sec. VI. Finally, we draw the conclusion and discuss the future directions in Sec. VII.

II. Related work

We devote this section to reviewing the previous research efforts toward solving the VRTR problem.

A. The VPTR problem

Since the inception of NFV, resource scheduling & allocation have always been a relevant topic [6]. In practical NFV settings, it is crucial to optimally decide where to deploy the involved VNFs and how to route the traffic to form the intended network services (as SFCs), which is commonly known as the VPTR problem [2], Compared to traditional service provisioning problems, VPTR is more challenging as the VNFs must be traversed in a predefined order without violating the service and resource constraints. Existing studies propose two variants to simplify the VPTR problem. The first direction is VNF Placement (VNFP), which seeks optimal VNF placement by assuming fixed routing schemes [6]. The second direction is Traffic Rerouting (TRR), which explores the best traffic steering schemes given fixed VNF placements [12]. Although these solutions can achieve optimal service provisioning in static networks, they have limitations in capturing substrate networks' time-varying dynamics and uncertainties [7].

B. The VRTR problem

Some works dig further and aim at solving the VNF Redeployment and Traffic Routing (VRTR) problem, which aims to dynamically redeploy VNFs and reconfigure the routing schemes in response to the dynamics of substrate networks. For instance, Liu et al. [7] employ the column generation technique to optimize the SFC deployment dynamically. Eramo et al. [13] propose three algorithms for VNF placement, SFC routing, and VNF migration to minimize the joint cost of energy consumption and QoS degradation. Ruiz et al. [14] and Nsga et al. [15] employ genetic algorithms for dynamic VNF provisioning to maximize the service acceptance ratio. Recently, researchers actively explored machine learning (ML) techniques, especially Deep Reinforcement Learning (DRL), to solve the VRTR problem [16]–[18]. Although these works account for the dynamic nature of NFV-enabled networks, they either over-simplify the routing process, which leads to sub-optimal outcomes, or operate reactively, resulting in long setup latencies and SLA violations [8].

To cope with modern networks' dynamic and uncertain nature, service providers require proactive service provision, which only a few prior works address. In particular, Wahab et al. [8] formulate the VRTR problem as an Integer Linear Programming (ILP) problem and propose a semi-supervised learning algorithm to accelerate the ILP solver by intelligently removing the redundant cost functions. Nonetheless, they omit the problem of traffic rerouting, which is essential for resource and QoS optimization. Pei et al. [19] propose a DRL-based algorithm to forecast the network conditions and adjust SFC deployment accordingly. While this algorithm can capture network uncertainties and make proactive adjustments, it requires massive agent-environment interactions to derive the optimal model, which is not always feasible. Also, the algorithm faces large convergence latency, which offsets the benefit of proactive SFC provisioning. Worse still, the actions derived by DRL cannot be readily interpreted by network operators, making its applicability questionable.

To realize dynamic, proactive SFC provisioning, we address the VRTR problem with a dual-window approach, which accounts for the dynamic traffic patterns and network conditions by proactively collecting network states and assessing the real-time entropy. Entropy, originating from information theory, has been widely applied across various disciplines [20]. Entropy-based models, e.g., Von Neumann entropy [21] and Shannon entropy [22], can effectively capture the inherent network uncertainty (or unpredictability). Our algorithm utilizes entropy to quantify the uncertainty of the IP links' transient qualities and proactively 1) pruning the nodes and links with high uncertainties; 2) releasing the VNF instances with no corresponding SFC demanding; 3) instantiate SFC provisioning by constructing and solving binary matrices. Compared to state-of-the-art solutions, our algorithm is intuitive, efficient, and requires no interactions with the network.

III. System and traffic model

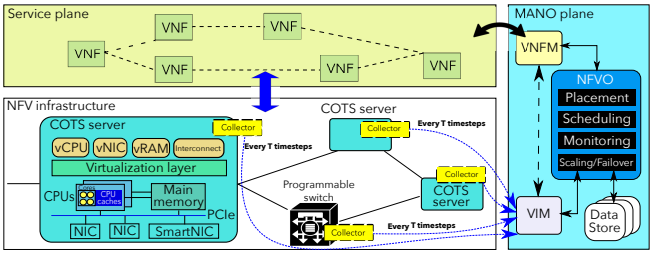


Fig. 1: VRTR problem in NFV-enabled networks

A. System model

Our work targets the standard NFV architecture, which consists of the NFV Infrastructure (NFVI), Service plane, and Management and Orchestration (MANO) plane, as illustrated in Fig. 1. The NFVI comprises heterogeneous network equipment, such as Commodity Off-The-Shelf (COTS) servers and programmable devices. VNFs are instantiated as SFCs in the Service Plane to form the desired network services. The MANO plane contains various modules for network management and service provisioning. The relevant modules for our work include monitoring, placement, and scheduling.

We propose a dual discrete time window algorithm to assess network stability and proactively adjust the SFC provisioning whenever necessary. Time is slotted equally as timesteps. We define M as the grand observation window and $T \ll M$ as the short, configurable window. Every T timesteps, each node updates the local network states to our algorithm residing in the

MANO plane via the Virtualized Infrastructure Manager (VIM). Our algorithm makes adjustments to optimize the M long-term service acceptance ratio. Our algorithm also designates the most suitable value of T whenever applicable.

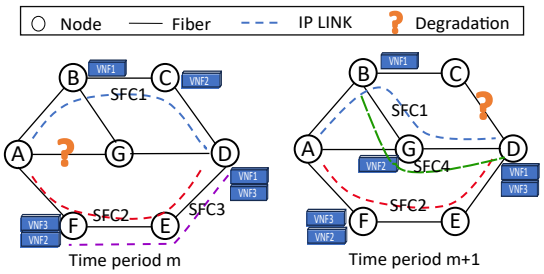


Fig. 2: An illustration of the VRTR problem.

A toy case of VRTR: Fig. 2 illustrates a simple scenario of the VRTR problem in a seven-node ISP network. The goal is to maximize the service acceptance ratio with dynamic traffic arrivals and unstable network conditions. We consider link quality degradations, which can occur independently due to component failures, software malfunctions, and resource contentions. In period m, to achieve the most suitable SFC provisioning, requests for SFC1, SFC2, and SFC3 are routed via path (A-B-C-D), (A-F-E-D), and (F-E-D), respectively. In the subsequent period $m+1$, due to the varied service requests and network conditions, a new round of SFC provisioning is required. Specifically, as the requests for SFC3 have terminated, its associated resources are released. Due to the degradation on (C-D), SFC1's path is rerouted via (A-B-G-D), with VNF2 redeployed at node G, to ensure transmission quality. SFC4 is also deployed along the path (B-G-D) to serve new requests.

B. Service traffic definitions

We model the network as a graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ stand for the sets of nodes and physical links. The parameter u, v represents two physical nodes, and $uv \in \mathcal{E}$ stands for a physical link between node $u \in \mathcal{V}$ and $v \in \mathcal{V}$. We use C_{uv}^{bw} to represent the capacity of link uv, C_u^{mem} the memory of node u, and C_u^{cpu} the units of available computing resource of node u. The set of network services is denoted as $\mathcal{S} = \{1, 2, \cdots, S\}$, where $s \in \mathcal{S}$ represents a specific type of service. The set of available VNFs is denoted by $\mathcal{F} = \{1, 2, \cdots, F\}$, where $f \in \mathcal{F}$ stands for a specific type of VNF. Each service s corresponds to a chain of ordered VNFs and is formalized as:

$$s := \left(F_s, \psi_s^{bw}, \psi_{sf}^{mem}, \psi_{sf}^{cpu}\right), \forall s \in \mathcal{S} \tag{1}$$

where $F_s = \{f_s^1, \cdots, f_s^n\}$ denotes the ordered set of its constituent VNFs, ψ_s^{bw} the required bandwidth, ψ_{sf}^{mem} the memory footprint, and ψ_{sf}^{cpu} the units of computing resources demanded by the VNF type f of service s.

Various factors, such as failures of network components, resource contention, or malfunctioning software, can impact the availability of an SFC, which is described as:

$$\prod_{p_s \in \mathcal{P}_s} \alpha_{p_s} \prod_{f_s \in \mathcal{F}_s} \alpha_{f_s} \leq \prod_{p_s \in \mathcal{P}_s} \alpha_{p_s} \tag{2}$$

where α_{f_s} is availability of the constituent VNF f_s, and α_{p_s} is the availability of the physical network component $p_s \in \mathcal{P}_s$.

Moreover, we assume every SFC can serve an aggregated set of users [23], for example, when multiple users require the same service between the two endpoints. We define the SFC request (SFCR) set as $\mathcal{R}$ and $r \in \mathcal{R}$ stands for an SFC request with specific ingress and egress. We have $r := (s, z_i^r, z_e^r)$, where s is defined in Eq. (1), and z_i^r, z_e^r are the ingress and egress nodes of SFCR r.

IV. VNF Replacement and Traffic Routing (VRTR)

This section provides problem formulations. We first formulate the VPTR problem and then address the VRTR problem.

Our first assumption is that network states are updated every T timesteps, where the initial timestep is allocated for network reconfiguration, and the subsequent $T-1$ timesteps are dedicated to network service, as described in [10]. We assume a value of one for the entropy at the initial update timestep to capture the fact that no knowledge of network state is assumed during network state updates. Let R_m be the SFCRs set in the next time interval $[mT, (m+1)T)$. The objective is to devise an optimal solution based on R_m.

A. VPTR formulation

Upon each network state update, the MANO plane must release the impacted VNFs and decide where to place them. We now formulate our VPTR problem with the following variables:

- z_u^r: a binary variable which returns 1 if the SFCR r traversed node $u \in \mathcal{V}$, and 0 otherwise.
- z_{uv}^r: a binary variable which returns 1 if the SFCR r traversed link $uv \in \mathcal{E}$, and 0 otherwise.
- z_{uf}^r: a binary variable which returns 1 if the VNF f requested by r is placed on node $u \in \mathcal{V}$, and 0 otherwise.
- q_s^r: a binary variable which returns 1 if SFCR r belongs to service type s, and 0 otherwise. Noted that $\sum_s q_s^r = 1$.

For a given period $[mT, (m+1)T)$, the objective is to find a VNF placement and traffic routing scheme to jointly optimize the service acceptance ratio and the number of required nodes, which is formulated as follows:

$$\max_{z_u^r, z_{uv}^r, z_{uf}^r, k_{uf}^r} \alpha \frac{1}{|R_m|} \sum_{r \in R_m} A_r - \beta \sum_{r \in R_m} \sum_{u \in \mathcal{V}} z_u^r \tag{3}$$

where α and β are weighting parameters. A_r is the availability of SFCR r, which is formulated as

$$A_r = \mathbb{1}\left(\prod_{u \in V_r} z_u^r \prod_{uv \in \mathcal{E}_r} z_{uv}^r \left(\sum_{s \in \mathcal{S}} q_s^r \prod_{f_i \in F_s} z_{uf}^r\right)\right) \tag{4}$$

In (4), $\mathbb{1}(\cdot)$ is an indicative function, $\prod_{u \in V} z_u^r$ denotes the availability of all the intermediate nodes, $\prod_{uv \in \mathcal{E}} z_{uv}^r$ indicates the availability of all the traversed links. And $\prod_{f_i \in F_s} z_{uf}^r$ represents required VNFs availabilities, where z_{uf}^r indicates the availability of a specific VNF instance.

The objective of the second part of (3) is to minimize the number of required nodes and, whenever applicable, consolidate VNFs associated with the same SFCR on a single function node (server), which reduces the operational cost.

We consider several constraints. In particular, the SFCRs' resource footprints, e.g., memory and bandwidth, should not exceed the resource capacity of individual link ($\forall uv \in \mathcal{E}$) and node ($\forall u \in \mathcal{V}$), which is formulated as follows:

$$\sum_{s\in\mathcal{S}}\sum_{f\in\mathcal{F}}\sum_{r\in\mathcal{R}_m} q_s^r \psi_{sf}^{mem}(z_u^r - \hat{z}_u^r) \le w_u^{mem} C_u^{mem} \tag{5}$$

$$\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}_m} q_s^r \psi_s^{bw}(z_{uv}^r - \hat{z}_{uv}^r) \le w_{uv}^{bw} C_{uv}^{bw} \tag{6}$$

In Eqs. (5)-(6), $w_u^{men}, w_u^{bw} \in [0,1]$ represent the available memory share of node u and the available bandwidth share of link uv, respectively. $\hat{z}_{uv}^r$ and $\hat{z}_u^r$ represent the values of z_{uv}^r and z_u^r at the previous time interval. Thus, their differences indicate the node- and link-level resource variation.

The VNFs collectively guarantee the SFC's availability. However, there is a limitation in the number of VNFs due to the physical node's resource constraint. The aggregated CPU share of the VNFs can not exceed a node's CPU capacity:

$$\sum_{s\in\mathcal{S}}\sum_{f\in\mathcal{F}}\sum_{r\in R_m} q_s^r \psi_{sf}^{cpu} \le C_u^{cpu} \tag{7}$$

For all $r \in R_m, u \in \mathcal{V}$, single path flow balance constraints must be enforced:

$$\sum_{v:(u,v)\in\mathcal{E}} z_{uv}^r - \sum_{v:(u,v)\in\mathcal{E}} z_{vu}^r = \begin{cases} 1, & \text{if } u \text{ is ingress} \\ -1, & \text{if } u \text{ is egress} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Note that while physical links are undirected, SFCRs possess a specific direction. Consequently, $z_{uv}^r \neq z_{vu}^r$.

If a physical link is traversed, the nodes connected by this physical link should be traversed as well:

$$z_u^r z_v^r = \begin{cases} 1, \text{if } z_{uv}^r = 1, \forall u, v \in \mathcal{V}_r, uv \in \mathcal{E}_r \\ 0, \text{otherwise} \end{cases} \tag{9}$$

Each required VNF $f \in F_s$ can only be placed on one node:

$$\sum_{v\in\mathcal{V}} z_{uf}^r = 1, \forall r \in \mathcal{R}_m, f \in F_s \tag{10}$$

Each node hosting a required VNF $f \in F_s$ must be active:

$$z_u^r \ge z_{uf}^r, \forall r \in R_m, f \in \mathcal{F} \tag{11}$$

The total delay of each request does not exceed D. We consider the processing delay at nodes and propagation delay at links, $\forall r \in \mathcal{R}_m$:

$$\sum_{u\in\mathcal{V}}\sum_{f\in\mathcal{F}} q_s^r z_{uf}^r d_{suf}^{\text{processing}} + \sum_{uv\in\mathcal{E}_r} z_{uv}^r q_s^r d_{s,uv}^{\text{link}} \le D \tag{12}$$

where $d_{suf}^{\text{processing}}$ is the processing latency when service $s \in \mathcal{S}$ using VNF of type $f \in \mathcal{F}$ is hosted by node $u \in \mathcal{V}$; and $d_{s,uv}^{\text{link}}$ is the propagation delay of service s traversing link $uv \in \mathcal{E}$.

The established model is an integer linear programming with objective (3) and constraints (5)-(12).

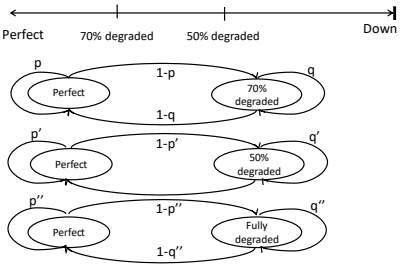


Fig. 3: Link states and their transition probability

B. VNF redeployment and traffic rerouting (VRTR)

We formalize the VRTR problem based on the preceding formulation of the VPTR problem. The crucial question is when to initiate SFC provisioning. To address this, we employ a dual discrete time window methodology, where M denotes a long-term observation window and T is the short time window. We update the network states for every T duration to adjust the SFC provisioning schemes and optimize the long-term objective. In a static network with constant traffic, link, and node conditions, the time interval for updating network states aligns with the long-term observation window, namely $T{=}M$. In a dynamic network with evolving topologies, traffic fluctuations, component failures, or resource contentions, the challenge of determining the optimal timing for resource reallocation necessitates strategic planning. Note that given $T \ll M$, the **long-term optimization goal** is formulated as:

$$\max_{T, z_u^r, z_{uv}^r, z_{rf}^u, k_{rf}^u} \alpha \frac{1}{M}\sum_{t=1}^{M} A_r - \beta \frac{1}{M}\sum_{t=1}^{M}\sum_{u\in\mathcal{V}} z_u^r \tag{13}$$

$$s.t.\ constraints\ (5) - (12)$$

V. Algorithm Design

This section presents our algorithm for dynamic, proactive SFC provisioning. Entropy, a metric for measuring network uncertainty, is central to our algorithm. By employing a dual time window strategy, our algorithm solves the VPTR problem under static network settings and addresses the VRTR problem under time-varying network conditions.

A. Network uncertainty quantification

We quantify the overall network uncertainty using two metrics, i.e., link entropy and network connectivity.

1) Link entropy: Link entropy can capture all the (nuanced) quality drift of a physical link in NFV-enabled systems. Specifically, component failures (hardware or software) can directly render a link unavailable. Traffic congestion, resource contentions, and malfunctioning software can severely degrade the quality of a connection, even if it is still available. We formulate link quality drifts using a two-state Markov model, which represents the quality of a physical link as either "perfect" (fully available) or "degraded (partially available)". The "degraded" state is predefined based on QoS level requirements, such as 70%, 50%, or full degradation, as shown in Fig. 3. We uniformly refer to various levels of degradation as "degraded".

At each timestep, the probability of a link remaining in the "perfect" state is denoted as p, and the probability of transition from a "degraded" to a "degraded" state is denoted as q.

To define an entropy-based measure of network uncertainty, let E_t^{perfect} be the set of perfect links at time t and let $E_{t+i}^{\text{perfect}} \subset E_t^{\text{perfect}}$ be the subset of those links remaining perfect at time $t+i$. Let l_{t+i} be a random variable representing whether an arbitrary link in E_t^{perfect} is perfect or degraded at time $t+i$. Then the probability that an arbitrary link in E_t^{perfect} is perfect at time $t+i$, is computed as

$$\mathbb{P}\left(l_{t+i} = \text{perfect} | l_t = \text{perfect}\right) = \frac{|E_{t+i}^{\text{perfect}}|}{|E_t^{\text{perfect}}|} \tag{14}$$

Based on (14), we collect and record each link's state (perfect or done) at regular intervals. Then, we count the number of transitions occurring from each state to the same and the other state. Subsequently, the transition probabilities p and q are derived by dividing the counts of "perfect-to-perfect" and "degraded-to-degraded" transitions by the total transitions originating from "perfect" and "degraded" states, respectively. Then, the link-perfect entropy is

$$H(l_{t+i}|l_t) = -\sum_{y=\{\text{perfect,degraded}\}} \mathbb{P}\left(l_{t+i}=y|l_t=\text{perfect}\right) \times \log\mathbb{P}\left(l_{t+i}=y|l_t=\text{perfect}\right)$$

The average link-perfect entropy h over T timesteps is:

$$h = \frac{1}{T}\sum_{j=1}^{T} H(l_{j+1}|l_0) \tag{15}$$

Where l_0 is the most recent timestep at which an entropy update occurred. We condition on the links found when network states are updated, i.e., $l_0 =$ perfect, because it is only those links that will be used in traffic routing. The links that are not consistently perfect are not useful for routing, so we do not include their entropy contribution in calculating h.

Further, we can also analytically evaluate h First, the t-step transition probability for a link (see, e.g., Ch.1 of [24]) as follows:

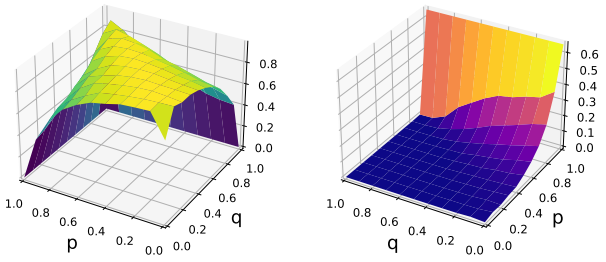
$$p_{uu}^t = \begin{cases} \frac{1-q}{2-p-q} + \frac{1-p}{2-p-q(p+q-1)^t} & \text{if } p+q<2 \\ 1 & \text{if } p+q=2 \end{cases}$$

$$p_{dd}^t = \begin{cases} \frac{1-p}{2-p-q} + \frac{1-q}{2-p-q(p+q-1)^t} & \text{if } p+q<2 \\ 1 & \text{if } p+q=2 \end{cases}$$

The notation p_{ud}^t is the probability that a link is in the state "degraded" after t transitions and having started in the state "up". We assume $p_{uu}^0 = p_{dd}^0 = 1$ and $p_{ud}^0 = p_{du}^0 = 0$. From the $(p+q-1)^t$ term, when $p+q-1<0$ links tend to oscillate. The network is thus stable when $p+q-1=1$ and bi-stable when $p+q-1=-1$. The link-up entropy $H(l_{t+i}|l_t)$ is then

$$H(l_{t+i}|l_t) = \sum_{y=\text{perfect,degraded}} p_{uy}^{t+i} \log_{uy}^{t+i} \tag{16}$$

As h decreases, the network becomes more predictable; as h goes to 1, the network becomes less predictable.



(a) Entropy (b) Algebraic connectivity

Fig. 4: Network measurements depend on p and q

2) Measure of network connectivity: To better capture the uncertainties, we further measure network connectivity using *algebraic connectivity*, defined as the second-smallest eigenvalue of the normalized Laplacian matrix of a graph [25]. The larger the value, the more well-connected is the network topology. The network is disconnected when *algebraic connectivity* equals 0.

Fig. 4 examine how p and q affect link entropy and connectivity in a random network topology with i.i.d links. The network connectivity is better when $p > 0.5$ and q is small and is a monotonically increasing (decreasing) function of p (q). Next, the link entropy, h, is minimized when q is very large, and p is very small, or q is very small, and p is very large: this is because links are very likely to stay degraded (or up), respectively, and so links are very predictable. Similarly, when p and q are both small, h is again low because links may be frequently breaking and recovered, links are predictably breaking. Finally, observe that the region where h is maximized corresponds to the region where $p = q$.

B. Long-term updating scheduling for VRTR

The basic steps of the proposed algorithm are given in Algorithm. 1. The input parameters include the long-term optimization life-cycle M, topology $(\mathcal{V}, \mathcal{E})$, initial requests R_0 where $r : (s, z_i^r, z_e^r) \in R_0$. The output parameters include the optimized parameters of VNF redeployment and traffic routing strategies Addressing the placement of VNF ("where") and the corresponding transmission strategies, the algorithm incorporates a matrix-based method, as specified in lines 10-13 of Algorithm 1. Besides, the timing for VNF placement and routing ("when") is determined through the monitoring of link states and the computation of link entropy and connectivity and achieved using Algorithm 2. For each time interval $t \in [mT, (m+1)T]$, Step 10 involves adjusting the current topology by pruning nodes and links, particularly those with high link entropy or insufficient connectivity. In step 11, the algorithm updates the available resources at nodes and links. This update includes the release of VNF instances in cases where there are no longer corresponding SFCs within the current period. We keep the VPTR strategies for traffic belonging to the last interval, i.e., $R_m \cup R_{m-1}$, and only deploy for the new demand Based on 10-13, we model the current ILP problem as (3) and solve it using the CPLEX optimizer.

1) VPTR solver: To accelerate the solver, we construct a binary matrix with the following form:

$$\begin{bmatrix} z_1^1 & \cdots & z_{|\mathcal{V}|}^1 & z_{1,f_1}^1 & \cdots & z_{|\mathcal{V}|,f_{|\mathcal{F}|}}^1 & z_{edge1}^1 & \cdots & z_{edge_E}^1 \\ z_1^2 & \cdots & z_{|\mathcal{V}|}^2 & z_{1,f_1}^2 & \cdots & z_{|\mathcal{V}|,f_{|\mathcal{F}|}}^2 & z_{edge1}^2 & \cdots & z_{edge_E}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ z_1^r & \cdots & z_{|\mathcal{V}|}^r & z_{1,f_1}^r & \cdots & z_{|\mathcal{V}|,f_{|\mathcal{F}|}}^r & z_{edge1}^r & \cdots & z_{edge_E}^r \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ z_1^R & \cdots & z_{|\mathcal{V}|}^R & z_{1,f_1}^R & \cdots & z_{|\mathcal{V}|,f_{|\mathcal{F}|}}^R & z_{edge1}^R & \cdots & z_{edge_E}^R \end{bmatrix}$$

ALGORITHM 1: VNF Redeployment and Traffic Routing (VRTR)

Input: Life-cycle M, topology $(\mathcal{V}, \mathcal{E})$, initial requests R_0 where $r{:}(s, z_i^r, z_e^r) \in R_0$
Output: VNF redeployment and traffic routing strategies

for each $u \in \mathcal{V}$, $uv \in \mathcal{E}$ **do**
  (a) $C_{uv}^{bw} \leftarrow$ collect the bandwidth capacity of each link;
  (b) $C_u^{mem}, C_u^{cpu} \leftarrow$ collect the node memory capacity and CPU computing capacity of each node;
  (c) Initialize $\omega_u^{mem}, \omega_{uv}^{bw}, \omega_u^{cpu} = 1$ for all $u \in \mathcal{V}, uv \in \mathcal{E}$;
  (d) $m = 0$
end for
for $t = 1 \cdots M$ **do**
  for $t \in [mT, (m+1)T)$ **do**
    if $t = mT$ **then**
      (a) Update topology: $(\mathcal{V}, \mathcal{E}) \leftarrow$ pruning the nodes, links with high entropy and low connectivity;
      (b) Update $\omega_u^{men}, \omega_u^{bw}, \omega_u^{cpu}$: releasing VNF instances if there is no-more demand;
      (c) Update new resource demand based on R_m;
      (d) Model and solve current ILP Eq.(3);
    else
      (e) Transmit based on current VPTR strategy;
    end if
  end for
  $m \leftarrow m + 1$;
  Monitor and collect the average link entropy h;
  if h deviates **then**
    Apply update rules using Alg. 2 for $T \leftarrow T^*$;
  end if
end for

where each row represents a cluster of users with the same ingress/egress and SFC type. For a given SFCR r, z_1^r to $z_{|\mathcal{V}|}^r$ indicate the traversed nodes, z_{1,f_1}^r to $z_{|\mathcal{V}|,f_{|\mathcal{F}|}}^r$ indicate how VNFs are placed at function nodes, and z_{edge1}^r to $z_{edge_E}^r$ uniquely determine the routing strategy. We note that the size of binary matrix is $|\mathcal{R}|(|\mathcal{V}| + |\mathcal{F}||\mathcal{V}_{\text{function}}| + 2|\mathcal{E}|)$. We define a table for edges for notation convenience and reduce the Integer programming problem to a binary problem: `edges` {`edgeID : node_src, node_dst`}. Binary constraints can more efficiently prune the search space in the LP relaxation step and identify feasible solutions faster Once the matrix elements are solved, the placement of the VNFs and routing policy of the SFCRs are uniquely determined. As demonstrated in Sec. VI, our CPLEX solver can efficiently solve the VPTR problem, even with many requests and large-scale networks.

2) The optimal small window T^*: To find the best small window T^*, we devise a stochastic gradient descent (SGD) algorithm as in Alg. 2.

In SGD, the possible range of the small window T duration is $T \in (0, M]$. The initial values, T_0 and T_1, are set at 0 and an arbitrary value less than M, respectively. The objective function $W(T)$, related to T and derived from Eq. (13), represents the overall average service acceptance across the window M. The gradient for the optimization process is $\frac{1}{k}(W(T_k) - W(T_{k-1}))$. This iteration rate decreases as the number of iterations increases, guiding the algorithm towards convergence in its final stages. The update process for T is iterative: T_{k+1} is computed by step 4, and the iterations continue as long as $|T_{k+1} - T_k| \geq \epsilon$. The iteration counter k is incremented in each loop, and $W(T_k)$ is updated accordingly. Alg. 2 typically converges within 10 iterations. Upon convergence, the optimal small window duration will be the one that results in the best performance metric value within the grand time window M.

ALGORITHM 2: Stochastic Gradient Descent (SGD) for computation T^*.

Initialize $T_0 = 0$, $\epsilon < T_1 < M$, $k = 0$, $\epsilon \ll 1$
while $|T_{k+1} - T_k| \geq \epsilon$ **do**
  $k \leftarrow k + 1$
  $W(T_k) \leftarrow \alpha \frac{1}{M} \sum_{t=1}^{M} A_r - \beta \frac{1}{M} \sum_{t=1}^{M} \sum_{u \in \mathcal{V}} z_u^r$
  $T_{k+1} \leftarrow T_k + \frac{1}{k}(W(T_k) - W(T_{k-1}))$
end while
Return $T^* \leftarrow T_{k+1}$ and $W(T^*)$

VI. Experimental Evaluation

A. Simulation settings

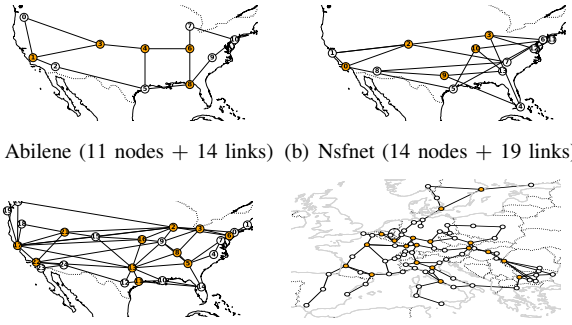
All the experiments are executed on a laptop equipped with an Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz and 4 GB memory. We consider two test scenarios to assess our algorithm: (1) static scenario with constant link quality; (2) dynamic scenarios with time-varying link quality. Following previous studies [26], we consider the real-world network topologies of the *Internet topology zoo* [27]. For space's sake, we only present the results for 4 out of the 250 topologies, namely Abilene (11

VNF type	Description	RAM	CPU (MHz)
Firewall	Click-based classifier (250 sequential rules)	1GB	2000
Auth/ftp	FTP Authentication services	3GB	3000
Billing	Click-based billing system	1GB	3500
Encoder	Compressing or encrypting data streams	2GB	1500
LPM	Click-based IP router pipeline	1GB	4000
STATS	Click-based flow stats collection	4GB	2000
IPsec	Click-based IPsec tunnel using IPsec	3GB	2000
MazuNAT	Click-based NAT pipeline by Mazu Networks	2GB	3000

TABLE I: The selected VNFs with their resource profiles

Type	Computational Resources
High-end server	36×Intel(R) Core(TM) i9 CPU @3.00GHz, RAM 263GB
Low-end server	32×Intel(R) Xeon(R) Silver CPU @2.10GHz, RAM 97GB

TABLE II: Node resource distribution



(a) Abilene (11 nodes + 14 links) (b) Nsfnet (14 nodes + 19 links)

(c) AttMples (25 nodes + 55 links) (d) Interoute (110 nodes + 145 links)

Fig. 5: Four of the ISP Network topologies used for evaluation.

nodes), Nesfnet (14 nodes), AttMples (25 nodes), and Interoute (110 nodes), as illustrated in Fig. 5. These topologies are chosen based on increasing scale and complexity. The yellow nodes can host VNFs, and the rest are switch nodes. We select eight prevalent VNFs from prior works [28], [29]. Their resource profiles are listed in Table I. The resource specification of each node is listed in Table. II by referring to the most commonly used servers in a small-scale enterprise data center. The link bandwidth is set to 100 Gbps to emulate the high-bandwidth settings. We optimally solve the VPTR model using CPLEX and iteratively execute the proposed Algorithm (Algorithm. 1) to address the VRTR problem over M timesteps.

B. Static network evaluation

We first evaluate our VPTR solver in the static network scenario. We generate 1000 trials for each experiment. In each run, we instantiate an SFC request between randomly selected ingress-egress pairs in the network. We construct two SFCs, each with three randomly selected VNFs from Table I. Users request either SFC at each timestep with a probability of 50%.

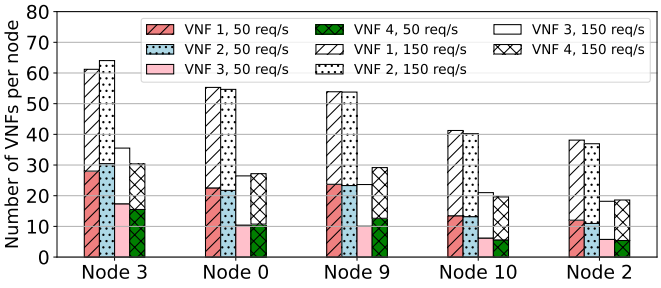


Fig. 6: VNF placement in AttMples topology

1) The number of VNFs placed on each node: We specifically select four VNFs, i.e., firewall (VNF1), FTP (VNF2), encoder (VNF3), and billing (VNF4), and fix the two SFCs to (VNF1 - VNF2 - VNF4) and (VNF1 - VNF3 - VNF4). Fig. 6 shows the average number of VNFs across the AttMples network under 50 and 150 requests/s. We sorted all the nodes based on their node degrees in descending order and selected the first five nodes as function nodes, which are denoted as $\mathcal{V}_{\text{function}} = \{3, 0, 9, 10, 2\}$. The number of VNFs per node is

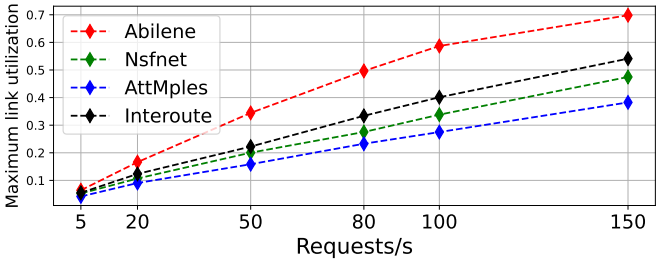


Fig. 7: Link utilization depend on traffic and topology

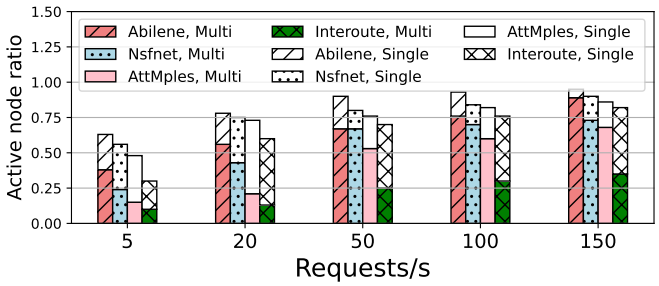


Fig. 8: The ratio of active nodes with varied traffic load

proportional to the traffic load. Notably, nodes 3 and 0, with the highest node degree of 9, necessitate more VNFs than other nodes due to their relatively heavier traffic load. This underscores the significance of deploying more VNFs to heavily loaded nodes. Besides, the results indicate that the requirement for different VNF types is proportional to the total service demand, achieving optimal SFC deployment.

2) The average resource utilization in different topologies: In Fig. 7, we compared the maximal link utilization under varied traffic loads. The results highlight that (i) the number of nodes does not fully indicate resource efficiency, and (ii) node degree is equally crucial. For instance, despite having fewer nodes and links, Abilene consistently showed high link utilization. In contrast, mesh-like topologies like AttMples distributed traffic more evenly. Based on our observation of all the topologies, radially structured networks with many central nodes generally exhibit higher link utilization.

3) The benefit of VNF consolidation: In Fig. 8, we explore the ratio of active nodes, calculated as the average active nodes over total nodes under different traffic loads. We consider two modes for our VPTR solver: "Single" - only preserving the first part of Equation (3), and "Multi" - the involvement of the second part. We specifically consider scenarios where at least 95% service requests are satisfied. We observe that introducing the second objective of active node minimization reduces resource utilization by about 40% while maintaining service quality. This efficiency stems from avoiding unnecessary traversals of redundant nodes and links, solving the VPTR problem. Additionally, Fig. 8 reveals that small-scale networks (e.g., Abilene) exhibit a higher active node ratio compared to larger networks (e.g., Interoute). This is attributed to the more distributed traffic in larger networks, leading to fewer active nodes to support equivalent traffic levels.

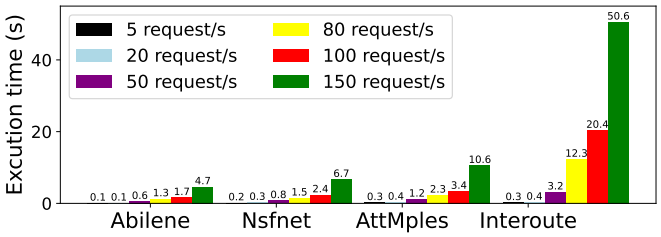


Fig. 9: Comparison of execution time under different loads

4) The execution efficiency under different network scales: We evaluate the network scales as the size of the binary matrix $|\mathcal{R}|(|\mathcal{V}| + |\mathcal{F}||\mathcal{V}_{\text{function}}| + 2|\mathcal{E}|)$. Note that the time needed to solve the ILP problem in different scenarios varies from 1 to 50 seconds. The formulated ILP problem in large-scale networks contains up to 74,400 variables and 5,000 constraints. Several factors contributed to the observed high efficiency. First, employing matrix operations greatly boosts performance by reducing nested loops and leveraging the speed of binary matrix calculations. Second, by integrating the traffic routing problem and the VNF placement problem in a single matrix, our approach avoids the problem of obtaining two local sub-optimalities in two different phases in the conventional approach, thus ensuring a globally optimal solution while maintaining execution efficiency.

C. Dynamic network evaluation

This part evaluates our algorithm's effectiveness in solving the VRTR problem under dynamic network settings.

1) The benefit of dynamic resource rescheduling: We first demonstrate the importance of dynamic resource scheduling. For the two-state Markov model proposed in Sec. V-A, starting from $p = 1$, we decrease it by 0.1 per 100 timesteps till 0 and fix q at 0.1, 0.5, 0.8 and 0.9, respectively. T is fixed to 100. We compare our algorithm with a baseline algorithm that excludes the part of dynamic resource rescheduling. The results are shown in Fig. 10. First, we observe a maximal 20% improvement in the service acceptance ratio compared to non-scheduling. Second, higher q (lower p) indicates lower link connectivity, hence lower service acceptance ratio. Besides, as p decreases, the gap between scheduling and non-scheduling initially increases and decreases to zero. Notably, the link entropy shows the same trend as t increases. This phenomenon can be interpreted as reallocating resources does not make much difference in a stable network (p is high). When there is a high link uncertainty (link entropy is high), resource reallocation can improve performance by up to 20%. If p drops significantly and network connectivity declines, reallocating resources does not improve performance either.

2) The impact of the small time window size T: Fig. 11 shows the results obtained by fixing $q = 0.1$ and varying p from $[0, 1]$. The network unpredictability increases initially, then decreases, reaching its maximum at $p = 0.5$. Network connectivity monotonically increases with p. We let $M = 100$ and update T every 5, and 10, respectively. As p increases, the

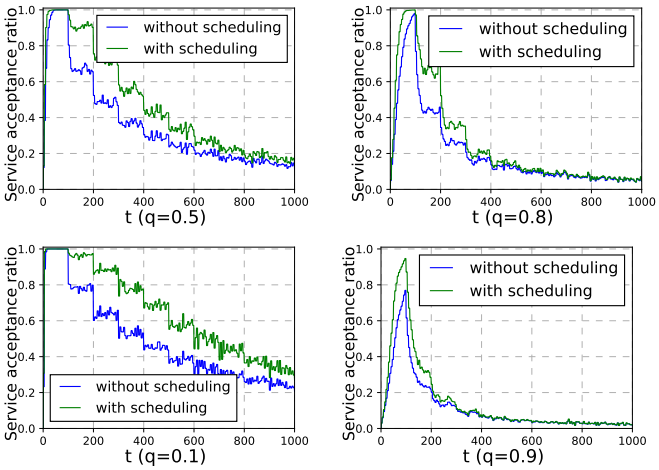


Fig. 10: Service acceptance ratio depending on link entropy

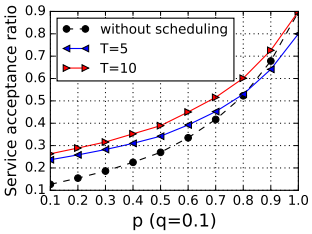


Fig. 11: Service acceptance ratio depending on T

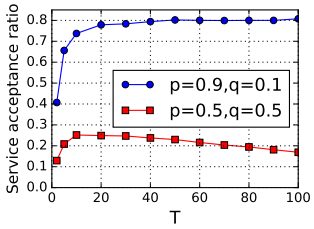


Fig. 12: Optimal updating time interval T^*

service acceptance ratio gradually improves. When $p=1$, the network is fully connected and highly predictable, enabling maximum transmission rates. When $T = 10$, periodic states updates improve the service acceptance ratio for $p<0.7$, while without scheduling is more effective for $p>0.7$. This suggests that frequent updates of the network states allow network operators to reallocate network resources promptly for better SFC provisioning.

3) The optimal small time window T: In Fig. 12, we depict two fixed settings for p and q, namely ($p = 0.9$, $q = 0.1$) and ($p=0.5$, $q=0.5$). The former exhibits high connectivity and low unpredictability, whereas the latter demonstrates high unpredictability. When $p=0.9$ and $q=0.1$, the optimal network states update time exceeds 80 timesteps, i.e., $T^*=80$, whereas for $p=q=0.5$, $T^*=10$, which implies that high network state update frequency enhances the service acceptance ratio when the network is less predictable.

D. Comparison with state-of-the-art algorithms

We compare our algorithm with three state-of-the-art solutions: *Genetic Algorithm (GA) [14], NSGA-II [15]*, and *Linear-multi-objective deep reinforcement Learning (MODRL) [19]* The MODRL algorithm solves weight uniformly distributed sub-objectives, i.e., the weight factor for each objective is set to 1; GA and NSGA have 100 populations for 100 generations. These two algorithms are customized for our problem formulation, and their initial populations are equally generated by randomly placing VNFs at function nodes. We

only present the results on specific topologies for space's sake.

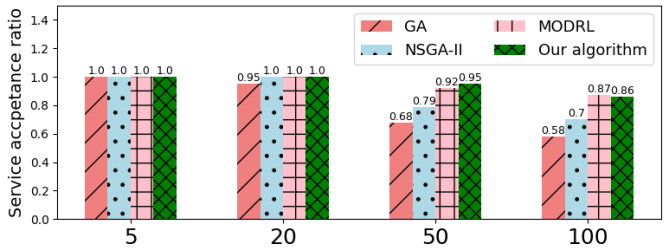


Fig. 13: Performance comparison on the AttMples Topology

Topology	GA	NSGA-II	DRL	Our algorithm
Abilene (11-node)	320s	240s	20s	4.6s
AttMples (25-node)	860s	760s	60s	10.6s

TABLE III: Execution efficiency of different algorithms

1) Performance: Fig. 13 shows their service acceptance ratio under different request rates on the AttMples topology. All algorithms achieve good results when low traffic loads (e.g., 5 or 20 request/s). GA and NSGA-II fail to sustain good performance as the traffic load increases because they heavily rely on the initial feasible population to find the optimum. For instance, GA performs the worst in all cases, as it can only find neighbor solutions from the initial population. Also, they only consider VNF placement and neglect the importance of traffic routing. Our algorithm and MODRL consistently achieve high acceptance ratios because both can proactively capture the network uncertainty and reallocate the resources. MODRL employs a sophisticated neural network and requires adequate interactions with the substrate network to attain optimal performance. Unlike MODRL, our algorithm is lightweight, transparent, and requires no offline training.

2) Efficiency: Table III shows the execution time for all the algorithms on Abilene and AttMples. In both scenarios, GA and NSGA-II take a similar time to run 100 iterations, and MODRL only takes $< 10\%$ of time than them. When the network scale increases from Abilene to AttMples, the execution time for all the algorithms also rises. In particular, when running on AttMples, the execution times of GA, NSGA-II, and MODRL increase by ≥ 3 times. Our algorithm takes only 10.6 seconds to find the best solution.

VII. Conclusion

Despite the proliferation of NFV in recent years, resource allocation remains daunting. While the VPTR problem has been well-studied in the literature, network operators still require a lightweight, proactive approach to address the more complicated VRTR problem. This paper proposes a novel algorithm that employs real-time network entropy to capture unstable network conditions and proactively address the VNF redeployment & traffic routing. Based on extensive evaluation, our algorithm outperforms two of three state-of-the-art solutions and achieves the best computation efficiency. In future work, we plan to deploy our algorithm in a real testbed for proactive VNF and traffic orchestration. We will also extend our algorithm to support state migration for stateful VNFs.

Acknowledgements

This work has been carried out in the context of the Beyond5G project, as part of the economic recovery plan from the French government, namely "France Relance" and the investments for the future program.

References

[1] B. Yi et al., "A comprehensive survey of network function virtualization," Computer Networks, 2018.
[2] S. Yang et al., "Recent advances of resource allocation in network function virtualization," IEEE TPDS, 2021.
[3] Q. Zhang et al., "Dynamic service placement in geographically distributed clouds," IEEE JSAC, 2013.
[4] A. Dalvandi et al., "Time-aware vm-placement and routing with bandwidth guarantees in green cloud data centers," in IEEE CloudCom, 2013.
[5] F. Song et al., "An optimization-based scheme for efficient virtual machine placement," International Journal of Parallel Programming, 2014.
[6] J. G. Herrera et al., "Resource allocation in nfv: A comprehensive survey," IEEE TNSM, 2016.
[7] J. Liu et al., "On dynamic service function chain deployment and readjustment," IEEE TNSM, 2017.
[8] O. A. Wahab et al., "Maple: A machine learning approach for efficient placement and adjustment of virtual network functions," Journal of Network and Computer Applications, 2019.
[9] P. Ji, Z. Ge, J. Kurose, and D. Towsley, "A comparison of hard-state and soft-state signaling protocols," IEEE/ACM TON, 2007.
[10] V. Manfredi et al., "Understanding stateful vs stateless communication strategies for ad hoc networks," in MobiCom, 2011.
[11] A. Gharehgoli et al., "Ai-based resource allocation in end-to-end network slicing under demand and csi uncertainties," IEEE TNSM, 2023.
[12] T. Wang et al., "Multi-resource load balancing for virtual network functions," in IEEE ICDCS, 2017.
[13] V. Eramo et al., "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," IEEE/ACM TON, 2017.
[14] L. Ruiz et al., "A genetic algorithm for VNF provisioning in NFV-enabled cloud/mec ran architectures," Applied Sciences (Switzerland), 2018.
[15] S. Khebbache et al., "A multi-objective non-dominated sorting genetic algorithm for vnf chains placement," in IEEE CCNC, 2018.
[16] Y. Xiao et al., "Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning," in IEEE/ACM IWQoS, 2019.
[17] A. Nouruzi et al., "Online service provisioning in NFV-enabled networks using deep reinforcement learning," IEEE TNSM, 2022.
[18] Y. Bi et al., "Multi-objective deep reinforcement learning assisted service function chains placement," IEEE TNSM, 2021.
[19] J. Pei et al., "Optimal VNF placement via deep reinforcement learning in sdn/nfv-enabled networks," IEEE JSAC, 2020.
[20] A. Halu et al., "Emergence of overlap in ensembles of spatial multiplexes and statistical mechanics of spatial interacting network ensembles," Physical Review E, 2014.
[21] F. Passerini and S. Severini, "The von neumann entropy of networks," arXiv preprint arXiv:0812.2597, 2008.
[22] K. Anand et al., "Entropy measures for networks: Toward an information theory of complex topologies," Physical Review E, 2009.
[23] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in IEEE NFV-SDN, 2015.
[24] J. R. Norris, Markov chains. Cambridge university press, 1998, no. 2.
[25] A. Jamakovic et al., "On the relationship between the algebraic connectivity and graph's robustness to node and link failures," in IEEE Next Generation Internet Networks, 2007.
[26] T. Zhang et al., "The role of the inter-controller consensus in the placement of distributed sdn controllers," Computer Communications, pp. 1–13, 2017.
[27] "Topology-zoo," http://www.topology-zoo.org/dataset.html/.
[28] T. Mahboob et al., "Dynamic VNF placement to manage user traffic flow in software-defined wireless networks," JNSM, 2020.
[29] A. Tootoonchian et al., "{ResQ}: Enabling {SLOs} in network function virtualization," in USENIX NSDI, 2018.