RQAP: Resource and QoS Aware Placement of Service Function Chains in NFV-enabled Networks

Haojun Huang, Jialin Tian, Hao Yin, Geyong Min, Dapeng Wu, Wang Miao

Abstract—Network Functions Virtualization (NFV), which decouples network functions from the underlying hardware, has been regarded as an emerging paradigm to provide flexible virtual resources for various applications through the ordered interconnection of Virtual Network Functions (VNFs), in the form of Service Function Chains (SFCs). In order to achieve the desired performance as dedicated hardware, how to efficiently deploy SFCs in NFV-enabled networks with limited resources is still a tremendous challenge. In this paper, RQAP, an effective Resource and Quality of Service (QoS) Aware SFC Placement approach is proposed to mitigate this issue with the QoS-guaranteed service provisioning at acceptable resource consumption. With the Markov property of VNFs, the resource and QoS aware placement of SFCs is modeled as a Markov-chain-based optimization problem, where the set of all possible placement states on diverse nodes is regarded as a state space in the Markov chain and each state is jointly determined by the initial state and transition matrices. Furthermore, the SFCs associated with traffic requests are re-sorted so as to efficiently instantiate VNFs of the same type in a resource-saving manner. On this basis, an efficient Backward-Viterbi-based heuristic mechanism is presented to conduct the optimal VNF placement in Markov chain space, with the aim of consumed-resource reduction, along with the QoS-based instantiation of virtual links between adjacent VNFs. Simulation results conducted in several scenarios demonstrate that RQAP can significantly achieve a trade-off between resource consumption optimization and QoS guarantee. Besides, the results show that our proposed approach can also effectively improve the SFC acceptance ratio and achieve desirable load balancing and scalability.

Index Terms—Network Function Virtualization, service function chains, Quality of Service, Markov chain.

1 Introduction

NETWORK Functions Virtualization (NFV) [1] [2], decoupling network functions from the underlying hardware, has been deemed as a promising network initiative to implement network functions and services in software in an efficient and effective manner. It allows network functions (or middle-boxes) to run in standard infrastructures rather than in customized hardware. Examples of network functions include Firewalls (FW), Load Balancers, Network Address Translation (NAT), Intrusion Detection Systems (IDS), Caches, and WAN-opt [3], [4], which are virtualized as Virtual Network Functions (VNFs) in NFV-enabled networks. Applications [1] have shown that NFV provides numerous benefits, ranging from centralized network management and programmability to reduced Operational expenses (OPEX) and Capital expenditure (CAPEX) for network operators. In reality, NFV receives strong support from many companies and standardization groups, including Google, Microsoft, IBM, Huawei, IETF and IRTF. Nowadays, the existing networks have unleashed a promising wave of innovations by introducing NFV and will evolve into NFV-enabled networks in the near future.

However, despite their benefits, NFV is still faced with some crucial considerable challenges to be tackled in networks [1], [5]. Specifically, how to deploy multiple VNFs in networks, in the form of Service Function Chains (SFCs), to ensure the Quality of Service (QoS) for different service provisioning with limited network resources. There already exist numerous efforts from the NFV community and academia in terms of the compromise between resource consumption and QoS guarantee for given NFV-based applications [6], for example, telecom services illustrated in Fig. 1. In such an application, diversified QoS performance with different resource consumption has been achieved with various SFC placement strategies. There are two SFCs to be deployed into the underlying networks, i.e., SFC$_1$: vFW$_1$ $\rightarrow$ vNAT$_1$, and SFC$_2$: vFW$_2$ $\rightarrow$ vNAT$_2$ $\rightarrow$ vIDS$_2$ $\rightarrow$ vWAN$_2$, which require bandwidth resources of 50 Mbps and 80 Mbps, respectively, and have the same ingress and egress. Furthermore, both FW and NAT are simultaneously included in such two SFCs, which can be placed on different VMs or the same one shown in Fig. 1. Besides, there are various virtual machines in the underlying networks with different resource capacities that can host different network functions. Taking Fig. 1(a) as en example, the instanced SFC$_1$ and SFC$_2$, which follows the same path as SFC$_1$ after embedding NAT$_2$, include FW$_1$-NAT$_1$ and FW$_2$-NAT$_2$-IDS$_1$-WAN$_1$, respectively. The total computing and bandwidth resources consumed are 15 cores and 650 Mbps, respectively. Fig. 1(b) makes an improvement in the

H. Huang is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. H. Huang is also with the Department of Computer Science, University of Exeter, Exeter, EX4 4QF, UK. Email: hjhuang@hust.edu.cn.
J. Tian and G. Min are with the Department of Computer Science, University of Exeter, Exeter, EX4 4QF, UK. Email: {jt816, g.min}@exeter.ac.uk.
H. Yin is with Tsinghua University, Beijing 100084, China. Email:h-yin@mail.tsinghua.edu.cn.
D. Wu is with the Department of Computer Science, City University of Hong Kong, Hong Kong. Email: dapengwu@cityu.edu.hk.
Wang Miao is with the Department of Computer Science, University of Plymouth, Plymouth, PL4 8AA, UK. Email: Wang.miao@plymouth.ac.uk.
Corresponding author: Haojun Huang and Jialin Tian.

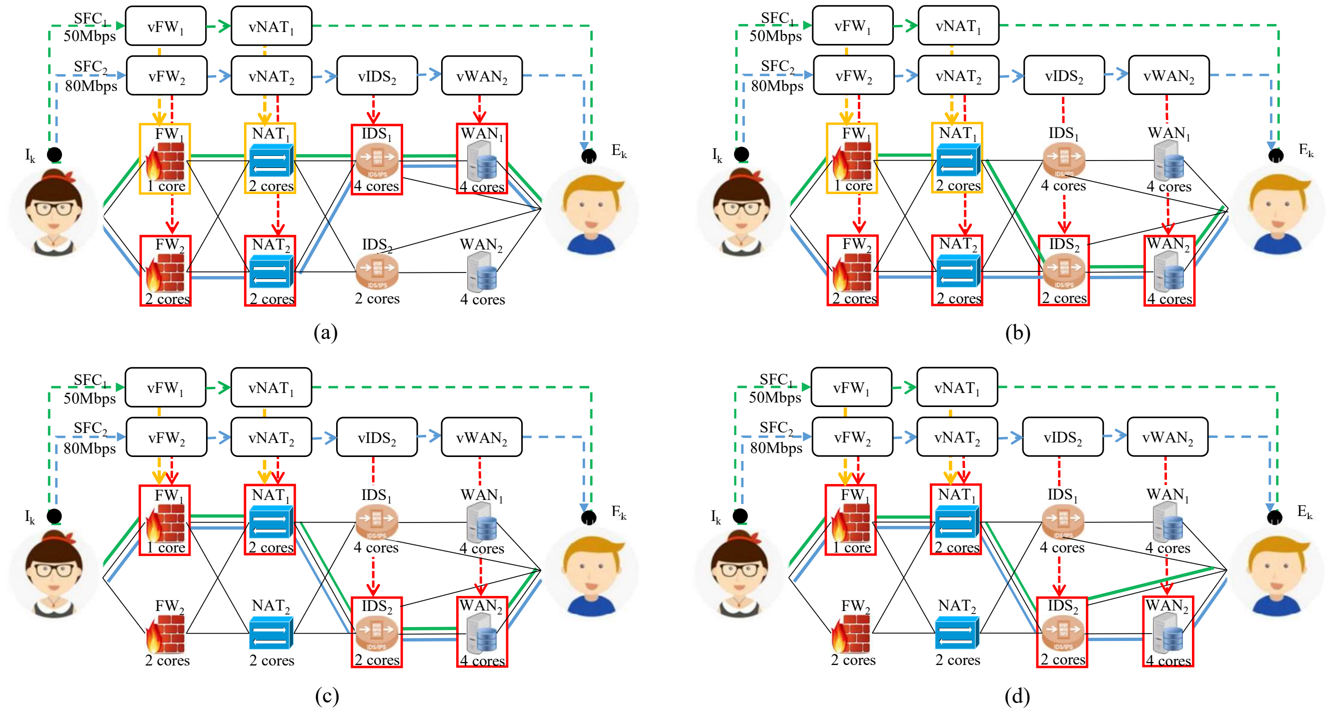


Fig. 1. SFC placement in NFV-enabled networks with different strategies, which have achieved diversified QoS guarantee with different resource expenditures.

computing resources assurance for vIDS$_2$ through IDS$_2$ and reduces it to 13 cores. Differently, Fig. 1(c) embeds the same VNFs of different SFCs into the same VM, which further significantly reduces computing resource consumption to 9 cores and also reduces resource fragmentation. As for Fig. 1(d), the mapping of VNFs remains unchanged, but there is a different path choice for SFC$_1$, which decreases the bandwidth resources by 50 Mbps and also changes the QoS on account of the variability of key metrics related to QoS loss for individual links.

It is clear to see from Fig.1 that NFV will offer different but not always the best performance for the same applications due to the unoptimized SFC placement. Intuitively, the resource consumption and the NFV performance in the above typical cases can be further improved through adjusting the ways of SFC placement and routing, but both of them often are not achieved simultaneously. In reality, a number of applications, such as network voice, multimedia sharing and high-performance storage, require optimized performance and QoS guarantee from VNFs in the virtualized infrastructure [1], [5]. Therefore, built on these investigations, this paper presents RQAP, an effective resource and QoS aware SFC placement approach, with the aim of reducing resource consumption and ensuring the QoS guarantee to boost NFV performance in NFV-enabled networks. In RQAP, the SFC placement is modeled as a Markov-chain-based optimization problem with the Markov property of its VNFs, where the set of all possible placement states on diverse nodes is regarded as a state space in the Markov chain and each state is jointly determined by the initial state and transition matrix. On this basis, the SFCs associated with traffic requests are re-sorted so as to efficiently instantiate VNFs of the same type in a resource-efficient manner. Besides, a Backward-Viterbi-based heuristic algorithm is presented to conduct the optimal VNF placement in Markov chain space to reduce resource consumption, followed by the QoS-based instantiation of virtual links between adjacent VNFs. The main contributions of this paper are summarized as follows.

- A novel SFC sort paradigm is proposed to ensure that SFCs with more types of VNFs can be deployed preferentially, so that as many kinds of VNFs as possible can be placed first and the same of them in later SFCs can be embedded into the shared hosting infrastructure, which effectively reduces resource fragmentation to obtain lower CAPEX.
- With the available resource in the underlying networks, the placement of a single SFC is innovatively devised as a Markov-chain-based model. The placement states of each VNF are determined by the initial state and transition matrix in the Markov chain, and the one with the maximum probability is selected as optimal VNF placement through a novel Backward-and-Viterbi-based heuristic iteration approach, which aims to reduce resource consumption.
- The specific routing between candidate nodes is designed to improve the QoS of SFCs. Instead of ordinary hop-based approaches, the hop, available bandwidth capacity, transmission delay and reliability of each link in RQAP are jointly considered to likely select routes with less QoS loss.
- Extensive experimental simulations are conducted in several scenarios, including multiple SFC amounts and varied throughput demands, to evaluate the performance of the proposed RQAP. The results show that RQAP significantly strikes a compromise between resource consumption and QoS guarantee.

The rest of this paper is organized as follows. Section

2 provides an overview of the related work. Section 3 formulates the system and SFC placement model. Section 4 introduces in detail the proposed RQAP for SFC placement in NFV-enabled networks. Section 5 presents the simulation, results and performance comparisons. Finally, we conclude the paper in Section 6.

2 Related Work

The optimal SFC placement, also referred to as Virtualized Network Functions-Forwarding Graph (VNFFG) in ETSI [7], has drawn great attention. A large number of preliminary investigations mainly took it into account as a VNE problem, which embeds virtual network requests into the substrate graph as logical graphs of VMs. However, VNE concerns more about the whole network and multipoint-to-multipoint requests while SFC placement works more on point-to-point flow with a sequence order [8]. Therefore, SFC placement with chain composition has a higher complexity than VNE.

There are many investigations discussing SFC placement [9]. A common approach is to formulate it as an optimization problem with a particular objective. An Integer Linear Program (ILP) was put forward in [10], [11], [12], [13], [14], [15], [16] with multiple goals of minimizing the cost of mapping, end-to-end delay, network operating cost and total resource consumption, respectively. Furthermore, the Mixed ILP (MILP) was applied in [8], [17], [18], and [15]. However, these linear programming problems can only be optimized in a small range because the SFC placement is NP-hard.

Most of the recent publications above [11], [12], [13], [17], [19], [20], [21], [22] have further proposed heuristic algorithms in order to cope with large infrastructures and efficiently guide the LP to be solved in a limited time. A heuristic algorithm in [12] made a selection of the network function decompositions. The ILP optimization was broadened to linear programming in [13], and min-cost flow problems were adopted to solve it. Kariz was proposed as a local search heuristic for distributed SFC in [17], which routed traffic layer by layer. Maximizing the Accepted SFC Bandwidth (MASB) was used in [19] to minimize the SFC bandwidth rejected for performing SFC placement during the Peak Hour Interval. In addition to the heuristic algorithms mentioned above, reinforcement learning is widely used in [23], [24], and [25] to find the optimal policy for the mathematical optimization problem proposed for SFC placement.

Furthermore, approaches relying on graph theory were applied in [26], [27], [18], [28]. BACON in [18] is based on the betweenness centrality of nodes in the graph which contains the information of servers, data communication delay between them and logical link. The underlying network was transformed to a multilevel overlay-directed network in [27] and then the Dijkstra algorithm was applied to find the path with the maximum weight. The matrix-based optimization and a multi-stage graph method were brought forward in [28], where the matrix was used to seek out the optimal steering of the traffic flows for each VNF and the multi-stage graph worked on the placement of chain between vertices of the different stages.

TABLE 1
The Important Parameters and Notations

Notations	Descriptions
N^s/L^s	Set of underlying network nodes/links
$v(n_i)$	Set of already-deployed VNF types in $n_i \in N^s$
$s(n_i)/c(n_i)/b(n_i)$	Storage/computing/bandwidth capacity of n_i
l_{ij}	Set of optional paths between n_i and n_j
$hop(l_{ij})/b(l_{ij})$	Hop number/bandwidth capacity of the shortest path in l_{ij}
$l_{ij\|z}$	Specific path between n_i and n_j, $l_{ij\|z} \in l_{ij}$
$hop(l_{ij\|z})/b(l_{ij\|z})$	Hop number/bandwidth capacity of $l_{ij\|z}$
$t(l_{ij\|z})/r(l_{ij\|z})$	Transmission delay/reliability of $l_{ij\|z}$
G_k^v	The k-th SFC, $G_k^v = (I_k^v, E_k^v, F_k^v, L_k^v)$
I_k^v/E_k^v	Ingress/egress of G_k^v with IP $ip(I_k^v)/ip(E_k^v)$
F_k^v/L_k^v	Set of VNFs/virtual links on G_k^v
$v(f_{k\|u})$	Type of u-th VNF $f_{k\|u} \in F_k^v$
$s(f_{k\|u})/c(f_{k\|u})$	Storage/computing demands of $f_{k\|u}$
$l_{k\|u}$	Virtual link between $f_{k\|u-1}$ and $f_{k\|u}$
$b(f_{k\|u})/b(l_{k\|u})$	Bandwidth demands of $f_{k\|u}/l_{k\|u}$
$S_{n_i}^{f_{k\|u}}/S_{l_{ij\|z}}^{l_{k\|u}}$	State that $f_{k\|u}/l_{k\|u}$ is mapped to $n_i/l_{ij\|z}$
π	State probability distribution matrix for $f_{k\|1}$
X	State transition probability matrix for $f_{k\|u}$
Y	State distribution probability matrix for $l_{k\|u}$

However, most work except [11], [19], [20], [28] [29] leave routing between VNFs out of consideration. It was modeled as a LP in [20] aiming at throughput maximization and adopted with the primal-dual algorithm. Different from the existing work, RQAP focuses on the varying priorities associated with different SFC placements with novel consideration of shared VNF instances among multiple SFCs. Our objective is to achieve the efficient deployment of VNFs and the establishment of optimal routing between them, while innovatively concurrently optimizing communication resources, computing resources, and QoS.

3 Problem Statement and Formulation

This section provides preliminary knowledge including system model and formulations to facilitate the understanding of our RQAP which will be proposed in the next section. The important parameters and notations are listed in Table 1.

3.1 System Model

The underlying network is modeled as an undirected weighted graph $G^s = (N^s, L^s)$ with two sets of nodes N^s and physical links L^s. Each node $n_i \in N^s$ is associated with storage capacity $s(n_i)$, computing resource $c(n_i)$, bandwidth $b(n_i)$ and the types of VNFs $v(n_i)$ that have already been deployed in node n_i. $l_{ij} \in L^s$ is the set of paths between n_i and n_j, which is associated with the hop of the shortest path achievable $hop(l_{ij})$ and the available bandwidth capacity $b(l_{ij})$. There are several optional paths in l_{ij}, represented by $l_{ij|z} \in l_{ij}$, and the corresponding available bandwidth capacity, delay, reliability and hop are denoted as $b(l_{ij|z})$, $t(l_{ij|z})$, $r(l_{ij|z})$ and $hop(l_{ij|z})$, respectively.

As for SFC requests, $\mathbb{S}$ denotes a set of SFCs and the $k-th$ SFC is represented as $G_k^v = (I_k^v, E_k^v, F_k^v, L_k^v) \in \mathbb{S}$ $(1 \le k \le K)$.

I_k^v and E_k^v are the ingress and egress of G_k^v with their IP addresses $ip(I_k^v)$ and $ip(E_k^v)$, and they are fixed in the network. F_k^v and L_k^v are the sets of VNFs and virtual links, respectively, where the $u\text{-}th$ VNF is $f_{k|u}{\in}F_k^v$ and the logical link between it and $f_{k|u-1}$ is $l_{k|u}{\in}L_k^v$. Besides, U is denoted as the length of G_k^v, i.e., the number of all VNFs in G_k^v. Thus, there are $(U{+}2)$ nodes $\{I_k^v, E_k^v, f_{k|u}(u=1,2,...,U)\}$ and $(U{+}1)$ logical links $\{l_{k|u}|u=1,2,...,U,(U{+}1)\}$ in SFC G_k^v, where $l_{k|1}$ stands for the link between I_k^v and $f_{k|1}$, and $l_{k|(U+1)}$ is the link between $f_{k|U}$ and E_k^v. Each $f_{k|u}$ is associated with its function type $v(f_{k|u})$ and source requirements which contain storage resource $s(f_{k|u})$, computing resource $c(f_{k|u})$ and bandwidth resource $b(f_{k|u})$. The bandwidth resource requirement of $l_{k|u}{\in}L_k$ is $b(l_{k|u})$.

3.2 Problem Formulation

Given K SFCs $G_1^v, G_2^v, ..., G_K^v$ stemmed from different requests, there are multiple candidate nodes and links in $G^s{=}\ (N^s, L^s)$ for VNFs and virtual links in $G_k^v = (I_k^v, E_k^v, F_k^v, L_k^v)$ to be implemented in. The placement of SFCs can thus be formulated as mapping problems from (F_k^v, L_k^v) to (N^s, L^s), which are supported by two kinds of binary variables, i.e., the mapping state of VNFs $S_{n_i}^{f_{k|u}}$ and that of links $S_{l_{ij|z}}^{l_{k|u}}$, as

$$S_{n_i}^{f_{k|u}} = \begin{cases} 1, & f_{k|u} \text{ is embedded in } n_i, \\ 0, & \text{others}, \end{cases} \tag{1}$$

$$S_{l_{ij|z}}^{l_{k|u}} = \begin{cases} 1, & l_{k|u} \text{ is mapped to } l_{ij|z}, \\ 0, & \text{others}. \end{cases} \tag{2}$$

Through deciding $S_{n_i}^{f_{k|u}}$ for each $f_{k|u}$ and n_i, as well as $S_{l_{ij|z}}^{l_{k|u}}$ for each $l_{k|u}$ and $l_{ij|z}$, VNFs are served by chosen nodes and the oriented flow routing is assured in the network. On this basis, there will be correspondingly several resource consumption produced in terms of computing and bandwidth resources. Firstly, the consumption of computing resources mainly occurs when instantiating VNFs on nodes, which can be denoted as

$$R_n(S_{n_i}^{f_{k|u}}) = \sum_{k=1}^{K}\sum_{u=1}^{U}\sum_{i=1}^{N} S_{n_i}^{f_{k|u}} W_{n_i}^{f_{k|u}} c(f_{k|u}), \tag{3}$$

where $c(f_{k|u})$ is the resource requirement of $f_{k|u}$ that is related to its type, and $W_{n_i}^{f_{k|u}}$ is a binary variable to indicate whether the same type of VNFs as $f_{k|u}$ has been deployed on n_i. The latter can be denoted as

$$W_{n_i}^{f_{k|u}} = \begin{cases} 1, & v(f_{k|u}) \in v(n_i), \\ 0, & \text{others}. \end{cases} \tag{4}$$

where $v(n_i)$ is the set of already-deployed VNF types in node n_i. Eq. (4) indicates that deploying $f_{k|u}$ to nodes that have contained VNF instances related to themselves can greatly reduce computing resource consumption. At the same time, it can also avoid possible resource fragmentation on the nodes. Secondly, the consumption of bandwidth resources can be defined by

$$R_l(S_{l_{ij|z}}^{l_{k|u}}) = \sum_{k=1}^{K}\sum_{u=1}^{U+1}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{z=1}^{Z} S_{l_{ij|z}}^{l_{k|u}} hop(l_{ij|z}) b(l_{k|u}). \tag{5}$$

Therefore, the total resource consumption is

$$R(S_{n_i}^{f_{k|u}}, S_{l_{ij|z}}^{l_{k|u}}) = \alpha R_n(S_{n_i}^{f_{k|u}}) + \beta R_l(S_{l_{ij|z}}^{l_{k|u}}), \tag{6}$$

where α and β are the normalization weights of computing and bandwidth resources, respectively, which are used to balance the two variables and limit the influence of their respective magnitudes on the total resource consumption.

Furthermore, different placement of SFCs also results in diverse QoS for users, which is mainly influenced by the hop, available bandwidth resources, end-to-end latency, reliability, etc, of links. The QoS loss is thus defined by

$$\begin{aligned} Q(S_{l_{ij|z}}^{l_{k|u}}) = &\sum_{k=1}^{K}\sum_{u=1}^{U+1}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{z=1}^{Z} S_{l_{ij|z}}^{l_{k|u}}[\mu_1 hop(l_{ij|z}) \\ &+ \frac{\mu_2}{b(l_{ij|z})} + \mu_3 t(l_{ij|z}) + \frac{\mu_4}{r(l_{ij|z})}], \end{aligned} \tag{7}$$

where μ_1, μ_2, μ_3 and μ_4 are the normalization weights that are defined in a similar manner in Eq. (6) to balance the four variables.

Additionally, there are some constraints needed to be met during the placement of SFCs. Considering that each SFC request is not split into two different paths, each VNF is restricted to be deployed on only one node, which can be denoted as

$$\sum_{i=1}^{N} S_{n_i}^{f_{k|u}} = 1, \quad \forall f_{k|u} \in V_k^v, \quad \forall G_k^v \in \mathbb{S}. \tag{8}$$

The total computing and storage resource demands for VNFs deployed on the node n_i should not exceed its resource capacity as

$$\sum_{k=1}^{K}\sum_{u=1}^{U} S_{n_i}^{f_{k|u}} c(f_{k|u}) \le c(n_i), \quad \forall n_i \in N^S, \tag{9}$$

$$\sum_{k=1}^{K}\sum_{u=1}^{U} S_{n_i}^{f_{k|u}} s(f_{k|u}) \le s(n_i), \quad \forall n_i \in N^S. \tag{10}$$

Besides, the sum bandwidth of all SFCs passing through n_i should be less than its maximum throughput as

$$\sum_{k=1}^{K}\sum_{u=1}^{U} S_{n_i}^{f_{k|u}} b(l_{k|u}) \le b(n_i), \quad \forall n_i \in N^S. \tag{11}$$

The same constraint also exists for each link $l_{ij|z}$ as

$$\sum_{k=1}^{K}\sum_{u=1}^{U} S_{l_{ij|z}}^{l_{k|u}} b(l_{k|u}) \le b(l_{ij|z}), \quad \forall l_{ij|z} \in L^S. \tag{12}$$

In addition, the network-flows conservation constraints need to be satisfied in the SFC placement problem, which declares that the inflow at each node (except the ingress and the egress) is equal to its outflow. It can be further converted to that the forward and reverse flows between any two nodes are 0, considering that the underlying network is an undirected weight network, which is defined by

$$\begin{gathered} \sum_{k=1}^{K}\sum_{u=1}^{U} S_{n_i}^{f_{k|u}} S_{n_j}^{f_{k|u+1}} - \sum_{k=1}^{K}\sum_{u=1}^{U} S_{n_j}^{f_{k|u}} S_{n_i}^{f_{k|u+1}} = 0, \\ n_i, n_j \ne I_k, E_k, \quad \forall k \in [1,K], \quad \forall u \in [1,U]. \end{gathered} \tag{13}$$

Therefore, in order to provide desired performance for users, it is requisite to design an efficient service function orchestration paradigm to enable SFCs can be revised and further deployed to appropriate infrastructure in a resource-efficient and QoS-guaranteed manner. To be specific, the resource and QoS aware placement of SFCs which aims to minimize both resource assumption and QoS loss can be formulated as

$$\begin{aligned} \min \quad & R(S_{n_i}^{f_{k|u}}, S_{l_{ij|z}}^{l_{k|u}}) + Q(S_{l_{ij|z}}^{l_{k|u}}), \\ s.t. \quad & \text{Eq. (8) - Eq. (13)}, \end{aligned} \tag{14}$$

where the optimization objective is the normalized sum of $R(S_{n_i}^{f_{k|u}}, S_{l_{ij|z}}^{l_{k|u}})$ and $Q(S_{l_{ij|z}}^{l_{k|u}})$ that are defined in Eq. (6) and Eq. (7), respectively.

4 RQAP Design

This section presents in detail the proposed RQAP for SFC placement in NFV-enabled networks. Firstly, the RQAP architecture is provided, and then how to consolidate and make a reasonable order for SFCs is introduced. After that, the essential Markov-chain-based SFC placement model is illustrated, followed by resource-aware VNF placement and QoS-aware virtual link instantiation.

4.1 RQAP Architecture

The basic idea of RQAP is to integrate and re-sort given SFCs, and then design a Markov-chain-based SFC placement strategy, which further includes the placement of VNFs and the adjustment of corresponding virtual links mapping, with explicit considerations on QoS guarantee and resource consumption reductions. Because the placement of each VNF is only related to the state of its previous VNF, such memoryless characteristics exactly satisfy the Markov chain, that is, the probability distribution of the next state can only be determined by the current state while the events before it are irrelevant. Thus, the placement of VNFs in the SFC placement process is regarded as discrete states, and then an efficient resource-aware manner is proposed for the optimal VNF placement in the state space, followed by QoS-aware virtual link placement.

Fig. 2 shows the architecture of RQAP. The operation of it includes four phases: SFC sequencing, Markov-chain-based SFC placement modeling, resource-aware VNF placement and QoS-based virtual link instantiation. RQAP first clusters SFCs with the same or approximate IP into a set, and sorts them according to how many of the same VNFs they have as other SFCs for further implementation in the shared hosting infrastructures. Such sequencing determines the order of SFCs, so that more VNFs of the same type can be deployed on the same node in the subsequent process, to reduce resource consumption and resource fragmentation. Then for each SFC, RQAP establishes a Markov-chain-based SFC placement model to denote the state probabilities of VNFs and virtual links, where available underlying network resources, the requirements of SFCs and the characteristics of resource consumption and QoS are taken into consideration. On this basis, resource-aware VNF placement is conducted, based on Backward-Viterbi heuristic algorithm, to select underlying nodes in the Markov-chain space and

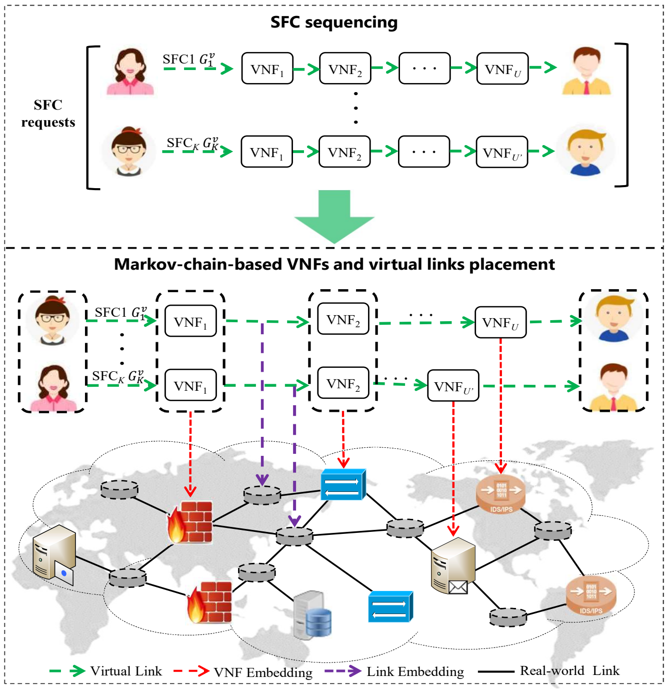


Fig. 2. The architecture of resource and QoS aware placement of SFCs.

orderly embed each VNF into them. Finally, RQAP makes routing decisions for virtual links between VNFs with the consideration of link state information and QoS control and then unicasts the packet to the established next-hop forwarder.

4.2 Service Function Chain Sequencing

It is a common case for various applications, for example, IDS and CDN, that many SFC requests share a public IP address to visit websites. Furthermore, most of them are always placed in adjacent access regions. That is to say, there is a certain amount of SFC similar in location and functions. Besides, most same functions can be implemented in expensive off-the-shelf programmable hardware in a parallel manner without effect to each other, which greatly reduces resource fragmentation on the nodes. Therefore, RQAP classifies all SFCs into a series of sets, built on their IP addresses, and then sorts them.

Given two SFCs $G_k^v = (I_k^v, E_k^v, F_k^v, L_k^v)$ and $G_j^v = (I_j^v, E_j^v, V_j^v, L_j^v)$, where I_k^v and I_j^v are their ingresses and E_k^v and E_j^v are their egresses if their IP addresses are the same, i.e.,

$$\begin{cases} ip(I_k^v) = ip(I_j^v), \\ ip(E_k^v) = ip(E_j^v), \end{cases} \tag{15}$$

they will be clustered into a set $S_{SF} = \{G_k^v | ip(I_1^v) = ip(I_2^v) = ..., ip(E_1^v) = ip(E_2^v) = ...\}$.

For G_k^v and G_j^v in S_{SF}, F_{V_k} and F_{V_j} are sets of VNFs' types contained, respectively. If one kind of VNF, f_i, is contained in G_k^v and G_j^v, i.e.,

$$f_i \in F_{V_j} \cap F_{V_k} (j \neq k), \tag{16}$$

Algorithm 1: Priority Determination for SFCs

```
Input: $\mathbb{S}=\{G_k^v|G_k^v=(I_k^v,E_k^v,F_k^v,L_k^v), k=\{1,...,K\}$
Output: Priority $F_k$ for $G_k^v$
1  for $G_k^v \in \mathbb{S}$ do
2      $S_{SF} \leftarrow \emptyset$;
3      $S_{SF} \leftarrow \{G_k^v\}$;
4      for $G_j^v \neq G_k^v \in \mathbb{S}$ do
5          if $ip(I_k^v)=ip(I_j^v)$ and $ip(E_k^v)=ip(E_j^v)$ then
6              $S_{SF} \leftarrow S_{SF} \cup \{G_j^v\}$;
7  for $G_k^v \in S_{SF}$ do
8      $F_k \leftarrow 0$;
9      for $f_i$ do
10         for $G_j^v \neq G_k^v \in S_{SF}$ do
11             if $f_i \in F_{V_k}$ and $f_i \in F_{V_j}$ then
12                 $F_k \leftarrow F_k+1$
13 return $F_k$
```

the binary variable $f_i(j,k)$ for G_k^v and G_j^v will be set to 1, otherwise, it will be 0. On this basis, F_k measures the amount of VNFs in G_k^v that are same as other SFCs, i.e.,

$$F_k=\sum_{j=1}^{K}\sum_{i=1}^{F} f_i(j,k), \quad j\neq k. \tag{17}$$

The determination process of priority F_k for each SFC is shown as Algorithm 1. The greater F_k is, the higher priority for G_k^v to be developed will be. In this way, the orchestration priorities of SFCs are that SFCs in any one set such as S_{SF} will be deployed together and SFCs with greater F_k in S_{SF} will be preferentially deployed.

4.3 Markov-chain-based SFC Placement Modeling

After the order for SFC placement has been determined in Section 4.2, each SFC $G_k^v=(I_k^v,E_k^v,F_k^v,L_k^v)$ will be mapped into the underlying networks $G^s=(N^s,L^s)$ in order. It is worth noting that the transition probability of deployment states for adjacent VNFs at any given time only depends on the placement state of the previous VNF. Considering such a Markov property, the Markov-chain-based probability matrix, denoted by $\lambda=(\pi,X,Y)$, is presented to model the SFC placement. π is the initial state probability distribution matrix, X is the state transition probability matrix for the placement of any two adjacent VNFs, which is determined by resource consumption, and Y is the state probability matrix for the instantiation of virtual links, which is determined by QoS guarantee.

$\pi=(\pi_1,\pi_2,\cdots,\pi_N)$ denotes the probabilities that 1-th VNF $f_{k|1}$ in G_k^v is embedded in potential nodes $\{n_1,n_2,\cdots,n_N\}$, where

$$\pi_i=P(S_{n_i}^{f_{k|1}}), \quad 1\leq i\leq N. \tag{18}$$

Since the deployment of each VNF is related to its resource requirement and the resource capacity of nodes, π_i is related to the resource consumption R_i incurred in node n_i and the corresponding link when n_i is selected for $f_{k|1}$, which is denoted as

$$\pi_i=\frac{\sum_{i=1}^{N}R_i-R_i}{(N-1)\sum_{i=1}^{N}R_i}, \tag{19}$$

where R_i further contains the computing and bandwidth resource consumption with the normalization weights α and β as

$$R_i=N_{n_i}^{f_{k|1}}B_{l_{0i}}^{l_{k|1}}D_{l_{0i}}[\alpha W_{n_i}^{f_{k|1}}c(f_{k|u})+\beta hop(l_{0i})b(l_{k|1})]. \tag{20}$$

It greatly combines one of the optimization objectives, i.e., $R(S_{n_i}^{f_{k|u}},S_{l_{ij|z}}^{l_{k|u}})$ in Eq. (6), and the constraints of Eq. (14) through two-value variables $N_{n_i}^{f_{k|1}}$, $B_{l_{0i}}^{l_{k|1}}$ and $D_{l_{0i}}$. Specifically, $N_{n_i}^{f_{k|1}}$ denotes the resource restrictions on optional nodes during the placement of $f_{k|u}$, which can be summarized as

$$N_{n_i}^{f_{k|u}}=W_{n_i}^{f_{k|u}}F_{n_i}^{f_{k|u}}+(1-W_{n_i}^{f_{k|u}})C_{n_i}^{f_{k|u}}, \tag{21}$$

where $W_{n_i}^{f_{k|u}}$ is a binary variable to indicate whether the same type of VNFs as $f_{k|u}$ has been deployed, as shown in Eq. (4), while $F_{n_i}^{f_{k|u}}$ and $C_{n_i}^{f_{k|u}}$ are two-value variables to indicate whether available resources $s(n_i)$ and $c(n_i)$ are greater than $s(f_{k|u})$ and $c(f_{k|u})$ and whether bandwidth resources on nodes are greater than $b(f_{k|u})$, respectively. $F_{n_i}^{f_{k|u}}$ and $C_{n_i}^{f_{k|u}}$ are thus denoted as

$$F_{n_i}^{f_{k|u}}=\begin{cases}1, & s(n_i)\geq s(f_{k|u})\ \&\ c(n_i)\geq c(f_{k|u}),\\ \infty, & \text{others},\end{cases} \tag{22}$$

$$C_{n_i}^{f_{k|u}}=\begin{cases}1, & b(n_i)\geq b(f_{k|u}),\\ \infty, & \text{others}.\end{cases} \tag{23}$$

Similarly to $C_{n_i}^{f_{k|u}}$, $B_{l_{ij}}^{l_{k|u}}$ is also a two-value variable that equals to 1 if and only if the available bandwidth resources $b(l_{ij})$ are greater than what $l_{k|u}$ demands. Therefore,

$$B_{l_{ij}}^{l_{k|u}}=\begin{cases}1, & b(l_{ij}),\geq b(l_{k|u})\\ \infty, & \text{others}.\end{cases} \tag{24}$$

To minimize resource consumption on links and shorten operation time, $D_{l_{ij}}$ is defined to limit the range of optional nodes, which equals to 1 if and only if $hop(l_{ij})$ is less than or equal to γ as

$$D_{l_{ij}}=\begin{cases}1, & hop(l_{ij})\leq\gamma,\\ \infty, & \text{others},\end{cases} \tag{25}$$

where γ is a threshold value, defined as the number of hops, to aid in the identification of whether the routing of l_{ij} involves an excessive number of hops.

Based on the above definitions of $N_{n_i}^{f_{k|1}}$, $B_{l_{0i}}^{l_{k|1}}$ and $D_{l_{0i}}$, R_i can be obtained from Eq. (20), and the initial state matrix π can thus be gained following Eq. (19).

X is the resource-related state transition probability matrix for VNFs. The relationship between the deployment states of any two adjacent VNFs can be considered as a transition of states, i.e., $S_{n_j}^{f_{k|u+1}}|S_{n_i}^{f_{k|u}}$ $(i,j\in[1,N])$, which describes the scenario that $f_{k|u+1}$ is embedded into n_j after its previous $f_{k|u}$ was deployed into n_i. Considering that n_i and n_j can be any optional nodes, the probability transition matrix X that contains $N\times N$ state transitions $x_{ij}=P(S_{n_j}^{f_{k|u+1}}|S_{n_i}^{f_{k|u}})$ can be defined as

$$X=\begin{bmatrix} P(S_{n_1}^{f_{k|u+1}}|S_{n_1}^{f_{k|u}}) & \cdots & P(S_{n_N}^{f_{k|u+1}}|S_{n_1}^{f_{k|u}})\\ \cdots & \cdots & \cdots\\ P(S_{n_1}^{f_{k|u+1}}|S_{n_N}^{f_{k|u}}) & \cdots & P(S_{n_N}^{f_{k|u+1}}|S_{n_N}^{f_{k|u}})\end{bmatrix}. \tag{26}$$

Similarly to the definition of π_i in Eq. (19), $x_{ij} \in X$ considers that n_i has embedded $f_{k|u}$ as a new ingress and is going to find an appropriate node n_j to map $f_{k|u+1}$. In this case, x_{ij} is decided by the resource consumption incurred in the transitions from $S_{n_i}^{f_{k|u}}$ to $S_{n_j}^{f_{k|u+1}}$ as

$$x_{ij} = P(S_{n_j}^{f_{k|u+1}} | S_{n_i}^{f_{k|u}}) = \frac{\sum_{i=1}^{N} R_{ij}^{'} - R_{ij}^{'}}{(N-1)\sum_{i=1}^{N} R_{ij}^{'}}. \tag{27}$$

$R_{ij}^{'}$ is the consumed computing and bandwidth resource for each n_j by deploying $f_{k|u+1}$ on it, which is defined in a similar way with R_i in Eq. (20) as

$$\begin{aligned} R_{ij}^{'} = & N_{n_j}^{f_{k|u+1}} B_{l_{ij}}^{l_{k|u+1}} D_{l_{ij}} \cdot \\ & [\alpha W_{n_j}^{f_{k|u+1}} c(f_{k|u+1}) + \beta hop(l_{ij}) b(l_{k|u+1})], \end{aligned} \tag{28}$$

where $N_{n_j}^{f_{k|u+1}}$, $B_{l_{ij}}^{l_{k|u+1}}$, $D_{l_{ij}}$ and $W_{n_j}^{f_{k|u+1}}$ are two-value variables that are defined in Eqs. (21), (24), (25) and (4). In this way, X is defined with the help of Eqs. (27) and (28).

It is worth noting that although X is defined for two adjacent VNFs, it varies with the requirements of each two VNFs instead of staying the same all the time. It is π and X (changing continuously with the order of VNFs) that jointly decide the deployment of VNFs.

Supposing there are node n_i with v_u and node n_j with v_{u+1}, y_{ij} is defined to describe the state that $l_{k|u+1}$ is mapped in one path that is belonged to the set l_{ij}. Let Y denote a QoS-related state probability matrix for virtual links. Then, we have

$$Y = \begin{bmatrix} y_{11} & \cdots & y_{1N} \\ \cdots & \cdots & \cdots \\ y_{N1} & \cdots & y_{NN} \end{bmatrix}, \tag{29}$$

where y_{ij} is described as

$$y_{ij} = [P(S_{l_{ij|1}}^{l_{k|u+1}}), \ldots, P(S_{l_{ij|z}}^{l_{k|u+1}}), \ldots, P(S_{l_{ij|L}}^{l_{k|u+1}})], \tag{30}$$

and $y_{ij}[z] = P(S_{l_{ij|z}}^{l_{k|u+1}})$ stands for the probability of choosing path $l_{ij|z}$, and L is the number of all optional paths between node n_i and n_j. Considering the key metrics of QoS guarantee, i.e., hop, available bandwidth resources, end-to-end latency and reliability, play important roles in the performance of link placement, as shown in Eq. (7), $y_{ij}[z]$ is defined as

$$y_{ij}[z] = P(S_{l_{ij|z}}^{l_{k|u+1}}) = \frac{\sum_{z=1}^{L} Q_{ij|z} - Q_{ij|z}}{(L-1)\sum_{z=1}^{L} Q_{ij|z}}, \tag{31}$$

where

$$Q_{ij|z} = \mu_1 hop(l_{ij|z}) + \frac{\mu_2}{b(l_{ij|z})} + \mu_3 t(l_{ij|z}) + \frac{\mu_4}{r(l_{ij|z})}. \tag{32}$$

The parameters $hop(l_{ij|z})$, $b(l_{ij|z})$, $t(l_{ij|z})$ and $r(l_{ij|z})$ are the hop, available bandwidth capacity, delay and reliability of $l_{ij|z}$, respectively, and μ_1, μ_2, μ_3 and μ_4 are their corresponding weight coefficients, defined in Eq. (7). The smaller $Q_{ij|z}$, the more likely its corresponding path $l_{ij|z}$ will be selected. In this way, the probability matrix of virtual links Y can be given with the help of Eqs. (31) and (32), and the probability of any path being chosen between any two points is thus determined.

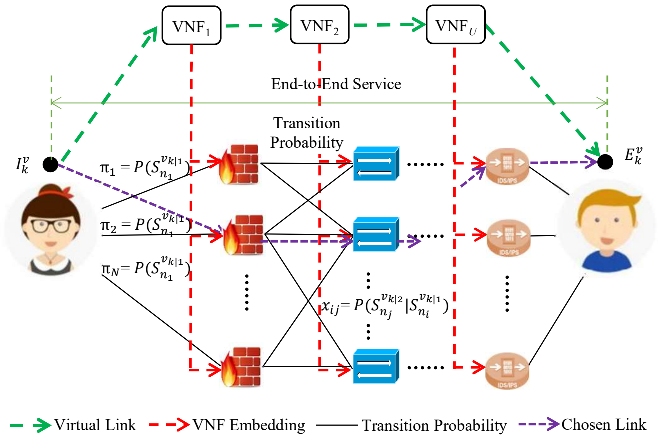

VNF$_1$
VNF$_2$
VNF$_U$
End-to-End Service
Transition Probability
I_k^v
E_k^v
$\pi_1 = P(S_{n_1}^{v_{k|1}})$
$\pi_2 = P(S_{n_1}^{v_{k|1}})$
$\pi_N = P(S_{n_1}^{v_{k|1}})$
$x_{ij} = P(S_{n_j}^{v_{k|2}} | S_{n_i}^{v_{k|1}})$
IDS/IPS
Virtual Link
VNF Embedding
Transition Probability
Chosen Link

Fig. 3. Markov-chain-based SFC placement in NFV infrastructure.

To sum up, RQAP conducts resource and QoS aware placement of each SFC based on $\lambda = (\pi, X, Y)$. Firstly, the resource-optimized VNF deployment is obtained by searching a maximum state probability list with the joint help of π and X. Secondly, the QoS-optimized path selection between the two mapping nodes is accomplished by finding the entry with the highest probability in Y. Sections 4.4 and 4.5 will introduce them in detail in the following.

4.4 Resource-aware VNF Placement

During the VNF placement, there are one ingress I_k^v, one egress E_k^v and U VNFs $V_k = \{f_{k|1}, \cdots, f_{k|u}, \cdots, f_{k|U}\}$ in G_k^v, where I_k^v and E_k^v are fixed in the underlying network as shown in Fig. 3. Therefore, the placement of U VNFs contains the mapping of 1-*th* VNF $f_{k|1}$ based on the ingress and π as Eq. (19), and the embedding of other VNFs $f_{k|u} (u \geq 2)$ based on Markov-chain transition probability matrix X as Eqs. (26) and (27).

It is assumed that the optimal deployment of VNFs is $\{n_{i_1^*}, n_{i_2^*}, \cdots, n_{i_U^*}\}$, which can be considered as a state list with maximum probability based on $\lambda = (\pi, X, Y)$. $n_{i_u^*}$ with the order i_u^* is the mapping node of VNF $f_{k|u}$, i.e., $S_{n_{i_u^*}}^{f_{k|u}} = 1$. However, directly searching the above state list is extremely computationally expensive. To solve this issue, RQAP takes advantage of dynamic programming to reduce the computing overhead of VNF placement. It is based on the idea that the result $n_{i_u^*}$ for $f_{k|u}$ must exist in the optimal deployment state sub-list from $f_{k|u}$ to $f_{k|U}$, i.e., $\{n_{i_u^*}, \cdots, n_{i_U^*}\}$, according to the principle of dynamic programming. If the above statement is not true, i.e., the state sub-list from $f_{k|u}$ to $f_{k|U}$ is $\{n_j, \cdots, n_{i_U^*}\}$, then the optimal deployment of $f_{k|u}$ should be updated to n_j rather than $n_{i_u^*}$, which is obviously contradictory to the previous hypothesis.

On this basis, Backward-Viterbi-based [30], [31] VNF placement is proposed in RQAP for resource optimization. For each optional node n_i of $f_{k|u}$, it only records the node n_j that has the maximum probability during the transition $S_{n_j}^{f_{k|u+1}} | S_{n_i}^{f_{k|u}}$ for $f_{k|u+1}$. That is to say, there are only N (the size of nodes) n_i-n_j pairs, rather than $N \times N$ pairs, recorded for each of the two adjacent VNFs. Such searching of n_i-n_j pairs for $f_{k|u}$-$f_{k|u+1}$ will be interactively executed from $u = U$ to $u = 1$, and there will be N state sub-lists

Algorithm 2: Resource-aware VNF Placement

```
   Input: $G^s = (N^s, L^s)$, $G_k^v = (I_k^v, E_k^v, F_k^v, L_k^v)$, $\pi$, $X$
   Output: $S_{n_i}^{f_{k|u}}$ for all nodes and VNFs
1  foreach $n_i \in N^s$ do
2      $\zeta_U(i)$, $\eta_U(i) \leftarrow$ Eq. (36) ;
3  foreach $f_{k|u} \in F_k^v$ $(u \neq U)$ do
4      foreach $n_i \in N^s$ do
5          $S_{n_i}^{f_{k|u}} \leftarrow 0$;
6          foreach $n_j \in N^s$ do
7              $x_{ij} \leftarrow$ Eq. (27);
8          $\zeta_u(i)$, $\eta_u(i) \leftarrow$ Eq. (37) ;
9  foreach $n_i \in N^s$ do
10     $\pi_i \leftarrow$ Eq. (19);
11     $P_i \leftarrow \zeta_1(i)\pi_i$;
12 $P^* \leftarrow \max_{1 \le i \le N} P_i$;
13 $i_1^* \leftarrow \arg\max_{1 \le i \le N} P_i$;
14 for $u = 2; u \le U$ do
15     $i_u^* \leftarrow \eta_{u-1}(i_{u-1}^*)$;
16     $S_{n_{i_u^*}}^{f_{k|u}} \leftarrow 1$;
17 return $\{n_{i_1^*}, n_{i_2^*}, \cdots, n_{i_U^*}\}$ and $S_{n_i}^{f_{k|u}}$
```

without considering the resource consumption between the ingress and $f_{k|1}$. Finally, through combining sub-lists with π, the list with the maximum probability will be selected, and the resource-aware placement of VNFs is accordingly solved.

For better description, RQAP defines a backward parameter $\zeta_u(i)$ for n_i and $f_{k|u}$ as

$$\zeta_u(i) = \max_{s_{u+1}, \cdots, s_U} P(s_u s_{u+1} \cdots s_U | \lambda), \tag{33}$$

where s_u is the deployment state of $f_{k|u}$, λ is the Markov-chain-based model, and $\zeta_u(i)$ represents the best weight of $f_{k|u+1}$ and all subsequent VNFs. Considering the feature of the Markov chain that the probability of a state transition at any given time only depends on its previous state, RQAP only considers the states between two adjacent VNFs. Therefore, $\zeta_u(i)$ can be simplified as

$$\zeta_u(i) = \max_{1 \le j \le N} \zeta_{u+1}(j) x_{ij}, \tag{34}$$

which is jointly decided by $\zeta_u(i)$ of $f_{k|u+1}$ and the transition probability x_{ij} as defined in Eq. (27). Besides, the node in the $(u+1)$-th layer that makes $\zeta_u(i)$ takes the maximum value is also recorded as

$$\eta_u(i) = \arg \max_{1 \le j \le N} \zeta_{u+1}(j) x_{ij}. \tag{35}$$

With the help of $\zeta_u(i)$ and $\eta_u(i)$, the iterative process is conducted from the egress E_k^v to the ingress I_k^v. The specific procedure is stated as follows:

1) **Initialization** $(1 \le i \le N)$:

$$\begin{cases} \zeta_U(i) = \dfrac{\sum_{i=1}^N hop(l_i) b(l_{k|U+1}) - hop(l_i) b(l_{k|U+1})}{(N-1)\sum_{i=1}^N hop(l_i) b(l_{k|U+1})}, \\ \eta_U(i) = 0, \end{cases} \tag{36}$$

where $\zeta_U(i)$ denotes the backward parameter of n_i for $f_{k|U}$, which is related to the shortest link l_i between n_i and E_k^v, $hop(l_i)$ is the hop of l_i, and $b(l_{k|U+1})$ means the bandwidth demands of the virtual link $l_{k|U+1}$.

2) **Recursion** $(1 \le i \le N)$:

$$\begin{cases} \zeta_u(i) = \max_{1 \le j \le N} \zeta_{u+1}(j) x_{ij}, & 1 \le u \le U-1, \\ \eta_u(i) = \arg \max_{1 \le j \le N} \zeta_{u+1}(j) x_{ij}, & 1 \le u \le U-1. \end{cases} \tag{37}$$

3) **Termination**:

$$\begin{cases} P_i = \zeta_1(i)\pi_i, \quad 1 \le i \le N, \\ P^* = \max_{1 \le i \le N} P_i, \\ i_1^* = \arg \max_{1 \le i \le N} P_i, \end{cases} \tag{38}$$

where P_i $(1 \le i \le N)$ is the probability of the optional list, which is decided by $\zeta_1(i)$ and π defined in Eq. (19), and the node $n_{i_1^*}$ with the highest probability value of P_i is selected as the deployment node for VNF $f_{k|1}$, i.e., $S_{n_{i_1^*}}^{f_{k|u}} = 1$.

4) **State tracking**:

$$i_u^* = \eta_{u-1}(i_{u-1}^*), \qquad 2 \le u \le U, \tag{39}$$

where i_u^* is the order of the placement node for $f_{k|u}$, i.e., $S_{n_{i_u^*}}^{f_{k|u}} = 1$, which is jointly determined by the deployment node of the previous VNF $f_{k|u-1}$ and the backtracking function $\eta_u(i)$.

Through the above four steps, the resource-aware VNF placement of RQAP is conducted following Algorithm 2. Specifically, the initialization and recursion processes are conducted in Lines 1-2 and Lines 3-8, respectively. Lines 9-13 describe the termination process and give the optimal deployment node $n_{i_1^*}$ for $f_{k|1}$, which is followed by the Markov-chain based state tracking in Lines 14-16 to obtain the optimal placement nodes $\{n_{i_1^*}, n_{i_2^*}, \cdots, n_{i_U^*}\}$.

Algorithm 3: QoS-aware Virtual Link Instantiation

```
   Input: $G^s = (N^s, L^s)$, $G_k^v = (I_k^v, E_k^v, F_k^v, L_k^v)$, $Y$
   Output: $S_{l_{ij|z}}^{l_{k|u}}$ for all links and virtual links
1  foreach $l_{k|u+1} \in L_k^v$ do
2      if $S_{n_i}^{f_{k|u}} = 1$ and $S_{n_j}^{f_{k|u+1}} = 1$ then
3          foreach $l_{ij|z} \in L_{ij}$ do
4              $S_{l_{ij|z}}^{l_{k|u}} \leftarrow 0$;
5              if $b(l_{ij|z}) \ge b(l_{k|u+1})$ then
6                  $y_{ij}[z] \leftarrow$ Eq. (31);
7                  $P(S_{l_{ij|z}}^{l_{k|u+1}}) \leftarrow y_{ij}[z]$;
8              else
9                  $P(S_{l_{ij|z}}^{l_{k|u+1}}) \leftarrow 0$;
10         $z^* \leftarrow \arg\max_z P(S_{l_{ij|z}}^{l_{k|u+1}})$ ;
11         $S_{l_{ij|z^*}}^{l_{k|u}} \leftarrow 1$;
12 return $S_{l_{ij|z}}^{l_{k|u}}$
```

4.5 QoS-aware Virtual Link Instantiation

After VNFs are embedded in nodes as guided in Section 4.4, the specific QoS-based virtual link instantiation of RQAP is conducted between the placement nodes in this section, based on the Markov-based deployment model with Markov properties $\lambda = (\pi, X, Y)$. Instead of adopting an ordinary hop-based routing approach, RQAP considers more indicators which cause loss and further have an effect on the QoS of each route, such as available bandwidth capacity, transmission delay and reliability. Routes with less QoS loss are more likely to be selected, and thus the QoS of RQAP is guaranteed.

Virtual links between VNFs on $G_k^v = (I_k^v, E_k^v, F_k^v, L_k^v)$ is denoted as $L_k = \{l_{k|1}, \cdots, l_{k|u+1}, \cdots, l_{k|U}, l_{k|U+1}\}$, where $l_{k|u+1}$ refers to the link between two adjacent VNFs $f_{k|u}$ and $f_{k|u+1}$. Assuming that node n_i and n_j have been selected to deploy $f_{k|u}$ and $f_{k|u+1}$, the QoS performance of the path between them, $l_{ij|z}$, is inversely proportional to its QoS loss. To quantify it, the QoS loss metric of path $l_{ij|z}$ in RQAP, as defined in Eq. (32), mainly considers length (represented as hop), bandwidth capacity, delay and reliability of the link, which is devised as $\mu_1 hop(l_{ij|z}) + \mu_2 / b(l_{ij|z}) + \mu_3 t(l_{ij|z}) + \mu_4 / r(l_{ij|z})$. μ_1, μ_2, μ_3 and μ_4 are the normalization weights of four components, respectively, with the help of their maximum and minimum values, so as to balance the four variables and eliminate the influence of their respective magnitudes on the total QoS loss. It can be seen that $hop(l_{ij|z})$ and $t(l_{ij|z})$ are directly proportional to QoS loss while $b(l_{ij|z})$ and $r(l_{ij|z})$ are inversely proportional to it. With these four components combined with normalization weights, the greater $Q_{l_{ij|z}}$, the worse QoS performance of $l_{ij|z}$ and accordingly the less likely it is to be chosen. As shown in Eqs. (30) and (31), state probability matrix Y is defined to describe the routing model of paths between any two nodes, where $y_{ij}[z] = P(S_{l_{ij|z}}^{l_{k|u+1}})$ denotes the probability of path $l_{ij|z}$ being chosen. Afterwards, the path with the highest probability value will be selected for the mapping of $l_{k|u+1}$ as shown in Algorithm 3.

5 Experimental Evaluation and Results

In this section, comprehensive simulations are implemented to evaluate the performance of our proposed RQAP using virtualization tools and services. The simulation settings and evaluation metrics employed in our experiments are first described, followed by the performance results obtained from the simulations. Finally, the comparative analysis between RQAP and other relevant solutions is conducted to assess the effectiveness and advantages.

5.1 Tested Environments and Settings

In our simulations, a substrate network comprising 50 physical nodes was generated using a randomization process. All nodes were assumed to be deployed within a cloud data center, with each node capable of both providing service functions and acting as forwarding nodes for their nearby connections to routers. Each pair of substrate nodes was randomly connected with a probability of 0.4. The resource capacities of each node, including CPU and storage, and the bandwidth of each link were subject to random distributions. Specifically, the CPU capacity ranged between 5 and 13 cores, the storage capacity ranged from 5 to 20 GB, and the link bandwidth varied from 100 to 200 Gbps. Moreover, the transmission delay and the link reliability were distributed within the range of [0, 30] ms and [0.997, 0.999], representing varying communication latencies and diverse quality levels of link connections, respectively.

TABLE 2
SFCs with different VNFs

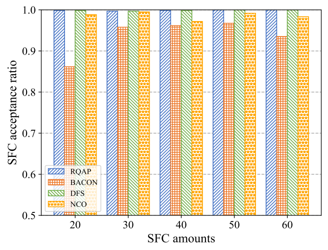
Len-1	Len-2	Len-3	Len-4
FW	FW-NAT	FW-NAT-IDS	FW-NAT-IDS-WAN
NAT	FW-IDS	FW-NAT-WAN	-
IDS	FW-WAN	NAT-IDS-WAN	-
WAN	NAT-IDS	-	-
-	NAT-WAN	-	-
-	IDS-WAN	-	-

There are four categories of network functions, i.e., FW, NAT, IDS and WAN-opt, included in the SFCs to be deployed, with their computing and storage resource requirements obeying the VNF instances provided by Riverbed [3] and Cisco [4]. To introduce diversity in the SFC compositions, the number of VNFs contained in each SFC is randomly selected from the range of [1, 4], while maintaining a consistent relative sequence between VNFs. This arrangement results in 14 alternative SFCs with varying lengths, as depicted in Table 2. For simulation purposes, the ingress and egress points of these SFCs are randomly chosen from nodes [0, 3, 4] and [1, 2, 5], respectively, to reflect realistic network scenarios. In addition, the throughput demands of virtual links within each SFC are generated randomly. Most importantly, the number of SFCs and the maximum throughput demands of each SFC are varied to simulate multiple experiment scenarios. First of all, the number of SFCs varies from 20 to 60, encompassing values [20, 30, 40, 50, 60], with a fixed bandwidth requirement of 100 Mbps. Secondly, the bandwidth requirement is gradually increased as [50, 75, 100, 125, 150], while maintaining a fixed number of SFCs at 20.

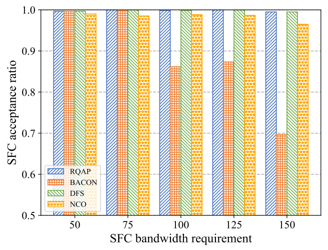
To provide a comprehensive visual evaluation of the performance of RQAP, three related solutions, i.e., BACON [18], DFS [32] and NCO [24], are also implemented in the experiment scenarios described earlier. BACON [18] is a heuristic solution for the component orchestration of the NFV platform, built on the Betweenness Centrality (BC) of nodes in the underlying network. It divides VNFs into subgroups according to their types and dependencies, calculates the BC of each node, and then deploys critical VNFs on nodes with the highest BC values, aiming to reduce resource consumption and latency during VNF placement. DFS [32] is a typical graph-based VNF placement algorithm that focuses on efficiently finding the shortest path between the ingress and egress points of each SFC. It selects nodes along this path to satisfy the resource requirements of VNFs, ensuring an optimized placement solution. NCO [24] is a Deep Reinforcement Learning (DRL)-based VNF placement scheme that leverages neural combinatorial optimization theory to learn an optimal placement policy. The policy is trained using LSTM and incorporated into the RL agent architecture with policy gradient methods.

5.2 Evaluation Metrics

Given SF_k which contains VNFs as $\{f_{k|1}, \cdots, f_{k|U}\}$, it is assumed that chosen nodes for VNFs are $\{n_{i_1^*}, \cdots, n_{i_U^*}\}$ and

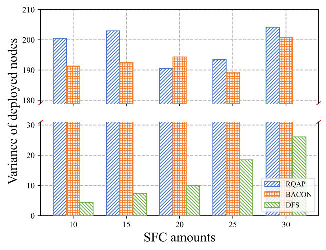


(a) Results with varied SFC amount

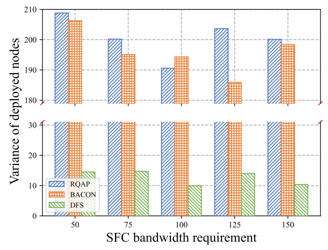


(b) Results with varied maximum bandwidth requirements

Fig. 4. The SFC acceptance ratios of RQAP, BACON, DFS and NCO when the number and the maximum bandwidth requirements of SFCs vary.



(a) Results with varied SFC amount



(b) Results with varied maximum bandwidth requirements

Fig. 5. The distribution variance of deployed nodes of RQAP, BACON and DFS when the number and the maximum bandwidth requirements of SFCs vary.

the routing paths between them are $\{l_{k|1},\cdots,l_{k|U},l_{k|U+1}\}$. To comprehensively evaluate the performance of SFC placement under various scenarios, we employ five evaluation metrics:

- SFC Acceptance Ratio: This metric, denoted as A_r, is the ratio of the number of successfully deployed SFCs to the total number of SFCs, i.e., $A_r = \frac{N_a}{K}$, which is also the probability of successfully placing SFCs within the network.
- Distribution Variance of Deployed Nodes: It calculates the variance of nodes where VNFs are embedded, which provides insight into the dispersion degree of VNF placement. A larger variance indicates a more dispersed distribution of VNFs across nodes, resulting in better overall load balancing.
- Consumed Resources: This metric refers to the weighted sum of consumed computing resources R_n and communication resources R_l per SFC on average. It is defined in Eq. (6) as $R = \alpha R_n + \beta R_l$.
- QoS Loss: The average QoS loss is determined by the packets traversing through G_k^v from its ingress to the egress on the selected path. This metric is quantified according to the definition provided in Eq. (7).
- Running Time: This metric quantifies the time taken to complete the placement of SFCs. It provides valuable information about the efficiency and computational performance of the algorithm.

5.3 Evaluation Results

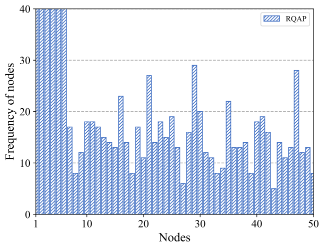
5.3.1 Comparisons in SFC Acceptance Ratio

The SFC acceptance ratios of RQAP, BACON, DFS and NCO are shown in Figs. 4(a) and 4(b), where the number and the maximum bandwidth requirements of SFCs vary from 20 to 60 and 50 to 150, respectively. It can be seen that the acceptance ratios of RQAP and DFS are stable at around 100.0% regardless of changes in the number and the maximum bandwidth requirements of SFCs, which are 99.8% and 99.8% on average, respectively, which greatly indicates that RQAP and DFS can deploy almost all SFCs. Additionally, although the SFC acceptance ratio of NCO dropped a little due to possible insufficient underlying network resources in Fig. 4, it is still greater than 98.0%, and the corresponding SFC placement can be regarded as a valid solution. In contrast, the SFC acceptance ratio of BACON is much lower, with an average value of 93.7% and 88.6% in Figs. 4(a) and 4(b), and decreases as the bandwidth requirements of SFCs increase. This is because BACON tends to deploy VNFs to a few critical nodes with high BC values, which causes SFC placement failure when the available resources in these nodes and relevant links are not sufficient. Taking Fig. 4(b) as an example, the SFC acceptance ratio of BACON is only 69.7% when deploying 20 SFCs with a maximum bandwidth requirement of 150 Mbps, while that of RQAP and DFS are both 99.5%. It shows well that RQAP proposed in this paper owns great robustness, which can effectively and stably deploy SFCs into the underlying networks and provide services in various scenarios.

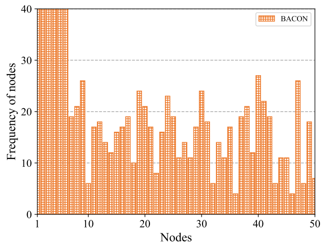
5.3.2 Comparisons in Distribution of Placement Nodes

In Fig. 5, the distribution variances of deployed nodes obtained by static algorithms (i.e., RQAP, BACON and DFS) are calculated when the number and the maximum bandwidth requirements of SFCs vary. It is clear that the variance of deployed nodes of DFS is small, revealing that the node utilization is centralized, which may cause heavy placement on a few nodes and traffic congestion. In contrast, the variances of RQAP and BACON are much larger and always greater than 185 regardless of the number and the maximum bandwidth requirements of SFCs, indicating that the nodes they selected for VNFs are more decentralized and there are better network loading balances. Furthermore, RQAP almost performs best, which improves the frequency of nodes by about 2.5% and 1800.0% on average compared to BACON and DFS, respectively. Therefore, RQAP is proven to achieve better load balancing when conducting VNF placement, compared to BACON and DFS, and maintain stability even as the number of SFCs increases.

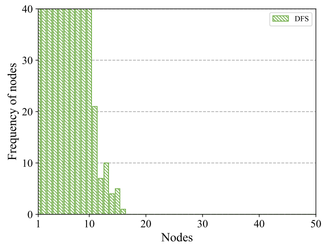
Taking the scenario where the number and the maximum bandwidth requirements of SFCs are 20 and 100 as an example, Fig. 6 illustrates the frequency of placement on each node of RQAP, BACON and DFS, and only the frequencies less than 40 are shown. As demonstrated in Fig. 6, RQAP and BACON make more extensive use of nodes to deploy VNFs, while DFS prefers to utilize the first 16 nodes which may lead to load imbalance in the overall system and causes congestion. In addition, all three methods have more than 40 occupancies on the first 6 nodes, because nodes [0, 3, 4] and [1, 2, 5] are randomly selected to be the ingress and egress of SFCs, respectively, so as to simulate the scenarios where a large number of SFC requests are applied from the same IP in reality.



(a) RQAP

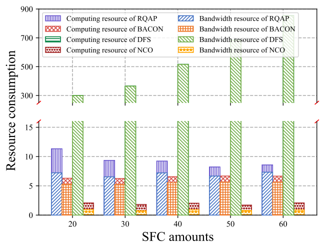


(b) BACON

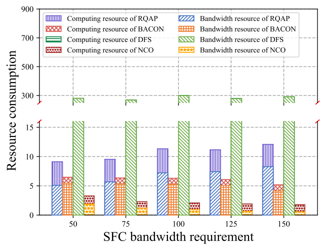


(c) DFS

Fig. 6. The frequency of placement on each node of RQAP, BACON and DFS when the number and the maximum bandwidth requirements of SFCs are 20 and 100.

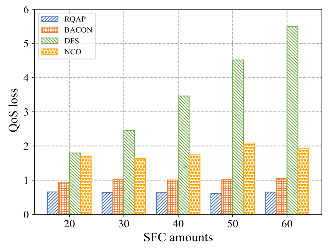


(a) Results with varied SFC amount

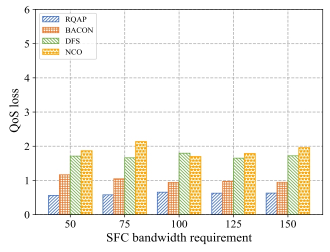


(b) Results with varied maximum bandwidth requirements

Fig. 7. The average resource consumption of RQAP, BACON, DFS and NCO when the number and the maximum bandwidth requirements of SFCs vary.



(a) Results with varied SFC amount



(b) Results with varied maximum bandwidth requirements

Fig. 8. The average QoS loss of RQAP, BACON, DFS and NCO when the number and the maximum bandwidth requirements of SFCs vary.

5.3.3 Comparisons in Consumed Resources

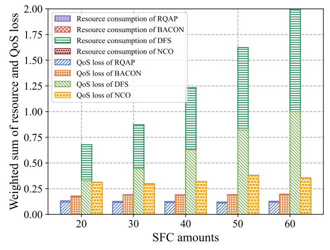
The average resource consumption of RQAP, BACON, DFS and NCO, including computing and bandwidth resources, is shown in Fig. 7, where the number and the maximum bandwidth requirements of SFCs vary. As shown in Fig. 7, the average resource consumption of RQAP, BACON and NCO is much less than that of DFS, no matter how many the number and bandwidth resource requirements of SFCs are, which is just 2.4%, 1.5% and 0.5% of the consumption of DFS on average, respectively. Besides, bandwidth resource consumption is the main part of DFS, which is because that DFS tends to start searching from fixed nodes during SFC placement and ignores the path distance between them, leading to excessive consumption of bandwidth resources. This phenomenon becomes more pronounced along with the increase in the number of SFCs, as shown in Fig. 7(a).

In contrast, the average resource consumption of RQAP decreases as there are more SFCs, which is attributed to the reutilization of nodes that have completed VNF instantiation before. Such reuse not only reduces the consumption of computing resources but also improves the scalability of RQAP. Moreover, when the number of SFCs remains the same and the maximum bandwidth requirements increase in Fig. 7(b), the average resource consumption of RQAP increases, especially in terms of bandwidth resources. It is because that RQAP will adjust the placement of virtual links to obtain better QoS, which results in more bandwidth resources than others, and the increased parts are proportional to the bandwidth resource requirements of SFCs.

Meanwhile, it is obvious that NCO achieves the minimal resource consumption, which is attributed to its resource-related reward design. Positive incentives for rewards in the RL agent guide NCO to take deployment actions that consume the least amount of resources. In addition, although the average resource consumption of BACON is slightly less than RQAP in Figs. 7(a) and 7(b), it may be achieved at the expense of discarding the placement of some SFCs as confirmed in the SFC acceptance ratio of Fig. 4. It indicates that the scalability of BACON is worse than RQAP. Therefore, considering the average resource consumption of RQAP is still relatively small and within a reasonable range, it is believed that RQAP trades acceptable resource consumption for better SFC acceptance ratio, load balancing and QoS (according to the next section for details).

5.3.4 Comparisons in QoS Loss

Figs. 8(a) and 8(b) evaluate the average QoS loss per SFC for RQAP, BACON, DFS and NCO as the number and the maximum bandwidth requirements of SFCs vary from 20 to 60 and 50 to 150, respectively. It is worth noting that RQAP achieves the lowest average QoS loss regardless of the number and the maximum bandwidth requirements, which is reduced by about 36.5%, 78.8% and 64.6% on average in Fig. 8(a), and 39.2%, 64.3% and 67.5% in Fig. 8(b), compared with BACON, DFS and NCO. It greatly indicates that RQAP effectively guarantees QoS through QoS-based virtual link placement during SFC placement. Furthermore, the average QoS loss of RQAP, BACON and NCO can maintain stability when the requirements of SFCs change, while that of DFS raises gradually as the number of SFCs increases. The latter is because DFS only focuses on the successful placement of SFCs, does not take into account the delay that closely relates to QoS and neglects QoS optimization. In addition, the performance of NCO in terms of QoS loss is not as good as it is in terms of resource consumption in Fig. 7. It is because that the optimization for QoS is ignored by the reward design of NCO, which makes it focus on selecting paths with fewer hops while ignoring important QoS metrics such as latency and reliability of the paths. Combined with the analysis of resource consumption in the aforementioned Fig. 7, it can be seen that the slightly


(a) Results with varied SFC amount

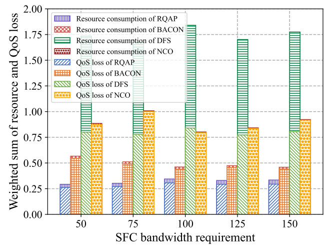

(b) Results with varied maximum bandwidth requirements

Fig. 9. Balance between resource consumption and QoS loss of RQAP, BACON, DFS and NCO when the number and the maximum bandwidth requirements of SFCs vary.

more resource consumption of RQAP is traded off for an effective improvement in QoS.

5.3.5 Comparisons in Balance between Resource Consumption and QoS Loss

In order to conduct a more visualized evaluation of each approach on their joint performance of resource consumption and QoS loss, the weighted sum of two metrics is introduced as

$$P = \rho R + \varphi Q, \tag{40}$$

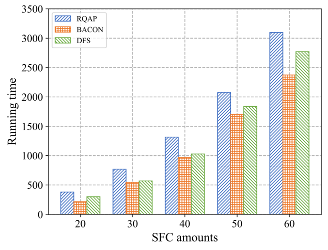
where the normalization coefficients ρ and φ are associated with the maximum value of resource consumption R and QoS loss Q in the corresponding experimental scenarios, respectively. Considering that less QoS loss is generally achieved at the cost of more resource consumption, especially in terms of bandwidth resources, the smaller P, the better balance between resource consumption and QoS loss.

As a consequence, Fig. 9 shows the weighted sum of resource consumption and QoS loss of RQAP, BACON, DFS and NCO when the number and maximum bandwidth requirements of SFCs vary. It can be observed from Fig. 9 that RQAP achieves the best performance as desired, where the weighted sum of RQAP is reduced by 33.2% and 34.5% on average compared with BACON, 88.3% and 81.6% compared with DFS, as well as 61.5% and 63.7% compared with NCO in Figs. 9(a) and 9(b), respectively. It greatly indicates that the slightly more resource consumption of RQAP, as shown in Fig. 7, has been offset by the effective improvement in QoS, as shown in Fig. 8, and RQAP delivers the best solutions with the balance between resource consumption and QoS loss than the other three algorithms.

Furthermore, when the number of SFCs increases in Fig. 9(a), it is clear that the weighted sum of resource consumption and QoS loss of RQAP, for each SFC, is the most stable one, and the growth rate is -0.7% on average, while the average growth rates of BACON, DFS and NCO are 2.6%, 31.2% and 3.8%, respectively. This is mainly thanks to that RQAP takes into account the optimisation of both resource consumption and QoS, and the QoS-based virtual link instance helps it keep QoS loss at a relatively low level even when there are a large number of SFCs to be deployed.

5.3.6 Comparisons in Running Time

The running time of static algorithms (i.e., RQAP, BACON and DFS) are illustrated in Fig. 10 by varying the number and the maximum bandwidth requirements of SFCs. The running time of NCO is not shown here because it takes a longer training time for converging to an effective deployment policy, and it is far greater than the running time of the static algorithms. In Fig. 10, the running time of all three static methods is below 3500ms, showing their great availability in SFC placement. Besides, it can be seen that the running time basically does not vary with the bandwidth requirements, but increases with the number of SFCs, because the latter directly affects the size of the problem. What's more, although the running time of RQAP is indeed larger than BACON and DFS, it is still within an acceptable range and used in exchange for better performance on the SFC acceptance ratio, load balancing, QoS loss and system scalability.


(a) Results with varied SFC amount

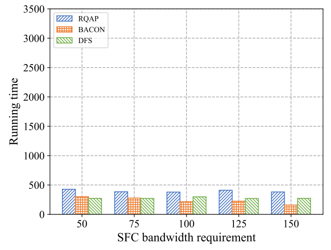

(b) Results with varied maximum bandwidth requirements

Fig. 10. The running time of RQAP, BACON and DFS when the number and the maximum bandwidth requirements of SFCs vary.

TABLE 3
The time complexity

Solutions	Time Complexity
RQAP	$O(UN^2)$
Exhaustion	$O(N^U)$
BACON	$O((N^3 - N^2)/2)$
DFS	$O(N^2)$
NCO	$O(I(U(\lvert N\rvert + \lvert L\rvert)(\lvert N\rvert + \lvert L\rvert + U)) + U^2(\lvert N\rvert + \lvert L\rvert)^2)$

5.3.7 Comparisons in Time Complexity

The time complexities of RQAP, the default exhaustion method for solving the Markov chain model, BACON, DFS, and NCO are elaborated in Table 3. For RQAP, the highest order of magnitude is the subroutine that conducts the resource-aware VNF placement as shown in Algorithm 2. Its time complexity is $O(UN^2)$, where U represents the number of VNFs across all given SFCs and N denotes the number of nodes available in the substrate network. Meanwhile, the default exhaustion method for solving the Markov chain model, which exhaustively enumerates all potential state sequences and compares their probabilities to identify the maximum probability, exhibits a significantly higher time complexity of $O(N^U)$. It is evident that the Backward-Viterbi algorithm employed in RQAP results in a substantial reduction in time complexity when compared to the default exhaustion method. As for other existing methods, BACON exhibits a computational complexity of $O((N^3 - N^2)/2)$ as analyzed in [18], while DFS demonstrates a complexity of $O(N^2)$. The time complexity of NCO can be approximated as $O(I(U(|N| + |L|)(|N| + |L| + U)) + U^2(|N| + |L|)^2)$, where $|L|$ is the number of links and I denotes the training rounds. The time complexity of NCO's training process is notably extensive, often necessitating a prolonged period for convergence. In contrast, heuristic algorithms such as RQAP, BACON, and DFS exhibit significantly smaller time complexities, which facilitates their swift and efficient deployment in NFV-enabled networks.

6 Conclusions

In this paper, RQAP, a resource and QoS aware solution of SFC placement in NFV-enabled Networks, has been proposed, which jointly aims at efficient resource consumption reductions and QoS guarantees. Inspired by the Markov property, the placement of an SFC is devised as a Markov-chain-based model which refers to VNF placement and link instances. After sequencing SFCs to be deployed by the IP address and VNFs contained, a heuristic algorithm based on Backward and Viterbi algorithms is proposed to find the best VNF placement manner for resource optimization, and the virtual link instance algorithm is also arisen for ensuring better QoS. Simulation results show that RQAP achieves effective SFC placement with a high and stable SFC acceptance ratio and great load balancing over a variety of network scenarios. Compared with the other related solutions, RQAP performs a better compromise between optimal resource consumption and better QoS, which is coincident with our expected objective. In the real-world networks, RQAP can be seamlessly integrated into the controller of NFV-enabled networks to guide the instantiation of VNFs on servers and the routing of traffic flows to fulfill specific service requirements for end-users.

Acknowledgment

This work is partially supported by the Natural Science Foundation of China (No.62372192 and No.92067206), the National Key Research and Development Program of China (No.2022YFB2702801), and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement (No.101030505). This article reflects only the authors' view. The European Union Commission is not responsible for any use that may be made of the information it contains.

References

[1] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surv. Tutor.*, vol. 21, no. 2, pp. 1409–1434, 2019.

[2] A. Nouruzi, A. Zakeri, M. R. Javan, N. Mokari, R. Hussain, and S. M. A. Kazmi, "Online service provisioning in nfv-enabled networks using deep reinforcement learning," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 3, pp. 3276–3289, 2022.

[3] Riverbed, "Riverbed virtual steelHead series," https://www.wansolutionworks.com/Virtual-SteelHead.asp, 2022.

[4] Cisco, "Cisco virtual wireless controller data sheet," https://www.cisco.com/c/en/us/products/collateral/wireless/virtual-wireless-controller/datasheetc78-714543.html, 2022.

[5] 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. Surv. Tutor.*, vol. 18, no. 1, pp. 236–262, 2015.

[6] X. Lin, D. Guo, Y. Shen, G. Tang, B. Ren, and M. Xu, "Sft-box: An online approach for minimizing the embedding cost of multiple hybrid sfcs," *IEEE/ACM Trans. Networking*, pp. 1–15, 2022.

[7] N. ETSI, GS, "Network functions virtualisation (NFV): Architectural framework," https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf, 2013.

[8] M. H. Gao, B. Addis, M. Bouet, and S. Secci, "Optimal orchestration of virtual network functions," *Comput. Networks*, vol. 142, pp. 108–127, 2018.

[9] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service function chaining in next generation networks: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 216–223, 2017.

[10] H. Huang, C. Zeng, Y. Zhao, G. Min, Y. Zhu, W. Miao, and J. Hu, "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, 2021.

[11] T. Li and Z. Zhu, "Qos-aware management reconfiguration of vnf service trees with heterogeneous nfv platforms," *IEEE Trans. Netw. Serv. Manage.*, pp. 1–1, 2022.

[12] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Comput. Networks*, vol. 93, pp. 492–505, 2015.

[13] J. B. Zhang, W. J. Wu, and J. C. S. Lui, "On the theory of function placement and chaining for network function virtualization," *Proceedings of the 2018 the Nineteenth International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '18)*, pp. 91–100, 2018.

[14] D. F. Li, P. L. Hong, K. P. Xue, and J. N. Pei, "Virtual network function placement and resource optimization in NFV and edge computing enabled networks," *Comput. Networks*, vol. 152, pp. 12–24, 2019.

[15] M. A. T. Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "vSPACE: VNF simultaneous placement, admission control and embedding," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 542–557, 2018.

[16] Y. Yue, B. Cheng, M. Wang, B. Li, X. Liu, and J. Chen, "Throughput optimization and delay guarantee VNF placement for mapping SFC requests in NFV-enabled networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 18, no. 4, pp. 4247–4262, 2021.

[17] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2479–2489, 2017.

[18] H. Hawilo, M. Jammal, and A. Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 643–655, 2019.

[19] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Trans. Networking*, vol. 25, no. 4, pp. 2008–2025, 2017.

[20] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 486–494.

[21] J. Liu, H. Xu, G. Zhao, C. Qian, X. Fan, X. Yang, and H. Huang, "Incremental server deployment for software-defined NFV-enabled networks," *IEEE/ACM Trans. Networking*, vol. 29, no. 1, pp. 248–261, 2021.

[22] J. Zheng, C. Tian, H. Dai, Q. Ma, W. Zhang, G. Chen, and G. Zhang, "Optimizing NFV chain deployment in software-defined cellular core," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 248–262, 2020.

[23] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang, and J. Zhang, "NFVdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[24] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 292–303, 2020.

[25] 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, 2020.

[26] Y. Xie, L. Huang, Y. Kong, S. Wang, S. Xu, X. Wang, and J. Ren, "Virtualized network function forwarding graph placing in sdn and nfv-enabled iot networks: A graph neural network assisted deep reinforcement learning method," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 1, pp. 524–537, 2022.

[27] B. B. Ren, D. K. Guo, Y. L. Shen, G. M. Tang, and X. Lin, "Embedding service function tree with minimum cost for NFV-enabled multicast," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1085–1097, 2019.

[28] S. Khebbache, M. Hadji, and D. Zeghlache, "Virtualized network functions chaining and routing algorithms," *Comput. Networks*, vol. 114, pp. 95–110, 2017.

[29] O. Alhussein and W. Zhuang, "Robust online composition, routing and NF placement for NFV-enabled services," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1089–1101, 2020.

[30] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[31] G. D. Forney, "The viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[32] J. Martín-Pérez and C. J. Bernardos, "Multi-domain VNF mapping algorithms," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2018, pp. 1–6.

Haojun Huang is an Associate Professor in the School of Electronic Information and Communications at Huazhong University of Science and Technology, China. He received his PhD degree in Communication and Information Engineering from the University of Electronic Science and Technology of China in 2012, and the BS degree in Computer Science from Wuhan University of Technology in 2005. His current research interests include Internet of Things, Network Function Virtualization, Wireless Communications, and Artificial Intelligence for networking.

Jialin Tian is currently a PhD student in Computer Science at the University of Exeter. She received the ME degree in Information and Communication Engineering from Huazhong University of Science and Technology, China in 2022, and the BS degree in Communication Engineering from Northeastern University, China, in 2019. Her research interests include Network Function Virtualization and Artificial Intelligence for networks.

Hao Yin is a Professor with the Research Institute of Information Technology, Tsinghua University, Beijing, China. His research interests span broad aspects of Multimedia Communications and Computer Networks. He received the BS, ME and PhD degrees in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 1996, 1999 and 2002, respectively. Prof. Yin was awarded the State Natural Science Award (Second Prize) in 2011 and the State Technological Invention Award (Second Prize) in 2012, and was appointed as a Yangtze River Scholar in 2021.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the BS degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems, Modelling and Performance Engineering.

Dapeng Oliver Wu is currently a Chair Professor at the Department of Computer Science, City University of Hong Kong. He received the PhD degree in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003, and BE degree in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 1990. His research interests are in the areas of Networking, Communications, Signal Processing, Computer Vision, Machine Learning, Smart Grid, and Information and Network Security. He has served as an Editor in Chief of IEEE TNSE, Associate Editor for IEEE TCC, ToC, TWC and TVT, and a Guest-Editor for IEEE JSAC. He is an IEEE Fellow.

Wang Miao is currently a lecturer in the Computer Science Department of the University of Plymouth, United Kingdom. He received his PhD degree in Computer Science from the University of Exeter, United Kingdom in 2017. His research interests focus on Vehicle Edge Computing, Unmanned Aerial Networks, Wireless Communication Networks, Software Defined Networking, Network Function Virtualisation, Applied Machine Learning, and Stochastic Performance Modelling and Analysis.