Computer Communications 224 (2024) 60–71

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Multi-objective optimization of SFC deployment using service aggregation and computing offload

Junbi Xiao [a], Jiaqi Zheng [a], Wu Wen [b], Mohsen Guizani [c], Peiying Zhang [a,d], Lizhuang Tan [d,*]

[a] Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China
[b] School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China
[c] Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi 999041, United Arab Emirates
[d] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China

ARTICLE INFO

Keywords:
Network function virtualization
Service aggregation
Computing offload
Service function chain
Cloud-fog-edge computing

ABSTRACT

New technologies such as virtualization and Software Defined Networking (SDN) have given traditional networks unprecedented flexibility. A Service Function Chain (SFC) formed by concatenating multiple Virtual Network Functions (VNFs) expands network functions to meet the growing personalized network requirements of applications. However, due to the decentralization of VNF instances and the tight cloud load, how to ensure Quality of Service (Qos) while deploying the SFC brings new challenges. Hence, a SFC deployment strategy based on multi-objective optimization, named MO-SACO is proposed, which regards delay optimization, reliability assurance and cost reduction as the main objectives. Specifically, we first design a VNF aggregation rule to handle raw SFC requests, which effectively reduces the SFC path latency and improves reliability. Then, VNF placement and traffic routing is modeled as a computation offloading decision based on cloud-fog-edge collaboration with the goal of target optimization according to functional, location, and resource constraints. Finally, we also take into account the performance changes of deployed SFCs and handle non-compliant requests in a timely manner. We conduct extensive simulations in networks of varying sizes and the results demonstrate that the proposed MO-SACO strategy not only achieves lower latency and cost, higher reliability compared with the state-of-the-art methods, but also has unexpected performance in terms of the deployment success rate and node load.

1. Introduction

5G mobile technology has brought disruptive changes to the communication architecture and enabled a wide range of services, including data center monitoring and measurement, network analysis, cloud fog edge network operation and maintenance management, massive bandwidth, etc. Furthermore, most existing network services have strict Quality of Service (QoS) requirements [1]. For example, intensive video streaming services have stringent requirements for end-to-end latency and real-time dynamics, broadband communication have to provide sufficient bandwidth but be relaxed about latency, and text streaming transmission such as email distribution and chat are expected to be given higher reliability. Software Defined Network (SDN) and Network Function Virtualization (NFV) [2] are expected to be key features of the 5G network architecture, and are considered to be the key guarantee for improving network flexibility and business performance. SDN separates the data plane and the control plane, uses controllers to realize centralized and unified management, which promotes the test and deployment of new services. NFV decouples software and hardware by abstracting network functions into software instances, which can running on the general server and called Virtual Network Functions (VNFs). VNFs can be deployed, migrated and deleted according to actual scenario requirements to significantly reduce Operating Expenditures (OPEX) and Capital Expenditures (CAPEX) [3]. Different types of VNFs can be combined into SFCs that provide various complex network functions [4,5]. As an important bearing form of network services, SFCs guide traffic routing sequentially so that improving the resource utilization potential of substrate network.

The SFC deployment mainly determines a feasible path by placing VNFs and mapping virtual links, where physical nodes and links meet the resource and bandwidth requests of VNFs, respectively. This process can be optimized according to business constraints to improve the QoS

* Corresponding author.
E-mail addresses: xiaojb@upc.edu.cn (J. Xiao), zhengjiaqi_0222@163.com (J. Zheng), wenwu@gzhu.edu.cn (W. Wen), mguizani@ieee.org (M. Guizani), zhangpeiying@upc.edu.cn (P. Zhang), tanlzh@sdas.org (L. Tan).

https://doi.org/10.1016/j.comcom.2024.05.017
Received 14 June 2023; Received in revised form 7 April 2024; Accepted 20 May 2024
Available online 3 June 2024

as well as network resource utilization, and reduce the CAPEX and OPEX.

The service latency has always been a key research issue in service deployment [6]. At present, the research on reducing service delay is mainly divided into two aspects. One is to reduce the delay of packet processing, queuing and forwarding, but this is subject to the capacity of the cloud network and the processing power [7], speed of the device itself. The other is to choose more reasonable physical paths for mapping the SFC to reduce the link resource consumption and the propagation delay so as to reduce the SFC's overall delay [8]. In recent years, there have also been some studies on offloading VNFs to access networks for deployment, but these efforts either ignore the cloud-to-access path, or do not consider the reliability and limited resources of edge devices. The emergence of fog computing [9] provides a reference for the above problems.

Fog computing [10] is defined as a highly virtualized computing paradigm, it migrates cloud centralized tasks to edge devices for execution. As a distributed computing structure, fog computing performs processing, storage and intelligent control around data devices. This elastic structure results in fog computing combining the superiorities of edge computing (closer to the user's location) and cloud computing (relatively sufficient resources) [11], so it has many advantages such as saving bandwidth resources and reducing latency. Apart from the above attempts, research about computation offloading [12–18] receives widespread concern, it complements fog computing technology. However, most researches about computation offloading consider offloading the computing services from resource-constrained devices to the cloud computing environment. Due to the above advantages, they should be an excellent solution for SFC deployment in terms of strict QoS and low cost. In fact, there are few studies that deeply investigate the rationality of service deploy strategies in mixed environments.

In critical-mission applications, reliability and server load are also significant SFC deployment issues. Most existing works presume that the infrastructure in an ideal state, but any physical or software failure will cause the SFC to be interrupted. The reliability guarantee of SFC is attributed to two types based on high reliability node selection and backup mechanism, the former will cause the resources of low reliability servers to be under-utilized, the latter will generate additional expenses. In addition, as the network architecture expands from single domain to multiples, high resource occupancy often leads to unbalanced network performance even service interruption. Maintaining network load balance is beneficial to reduce service delay and accommodate more requests.

Drawing from the analysis presented, this paper proposes a **M**ulti-**O**bjective SFC deployment optimization strategy based on **S**ervice **A**ggregation and **C**omputation **O**ffloading, named MO-SACO, which prioritizes delay, reliability, and cost. MO-SACO performs different functions in the SFC preprocessing phase and deployment phase. Preprocessing is usually a neglected part of research. At this stage, MO-SACO intends to aggregate VNFs with the same functions within a SFC and place them in the same node, which reduces the VNF count without affecting the normal function of the SFC. It effectively reduces queuing delay, mapping costs and improves reliability. In the deployment phase, MO-SACO leverages fog resources and edge devices to model VNF placement traffic routing as a computation offloading decision. We also unexpectedly discovered that VNF aggregation can jointly optimize SFC paths with cloud-fog-edge offloading to achieve additional optimization effects. Finally, we also take into account the performance changes of deployed SFCs and handle non-compliant requests in a timely manner. The main contributions of this paper is as follows:

- We express the multi-objective SFC deployment optimization as a Mixed Integer Linear Programming (MILP) model, which bridges latency, reliability, cost, and congestion that not fully considered in existing work.
- Different from the backup mechanism, we use service aggregation to aggregate VNFs that perform the same function in a SFC, so that requests are converted into SFCs with fewer VNFs. This is expected to save instantiation costs and reduce queuing delays while ensuring request reliability.
- Using the idea of cloud-fog-edge collaboration, taking into account both wired and wireless transmission, we select the best offloading decision of VNFs in different networks then route traffic, and service aggregation can appropriately make up for the single-node internal round-trip path faced by offloading.
- We conduct extensive simulations in resource-rich and constrained network environments. The results show that MO-SACO can not only adapt to different network environments and achieve excellent performance in terms of latency, reliability and cost, but also perform unexpectedly in cloud load relief.

The rest of the article is organized as follows. In Section 2, this paper describes the related literature on the service function chain's deployment. Section 3 defines the scenarios as well as models the network and the SFC request. Section 4 describes the strategies for optimizing SFC deployment. Section 5 details the MO-SACO algorithm and sub-algorithms proposed. Section 6 simulates the MO-SACO and analyzes the evaluation results. In Section 7, we summarize this paper and propose future directions.

2. Related work

Due to the difference of network environment, jointly placing VNFs and routing traffic will generate multiple sets of variables, making the solution of the problem more complex. Solving the multi-constraint SFC deployment problem using mathematical theory has been proven to be an NP-Hard problem [7,19]. This section divides the existing researches into four categories according to the optimization objectives.

2.1. Service latency

The network latency is an indicator worthy of attention in the SFC deployment. In a service request, the latency is the time required for traffic to pass through all VNFs in the current SFC from source to destination. Alleg et al. [20] assumed that the processing delay of a single function is positively related to the available resources of the server, and studied latency-aware VNF placement and chaining. But this work considered using the core servers to host all VNFs while ignoring the heterogeneous network characteristics. In fact, this is also what many jobs ignore. Subramanya et al. [21] modeled SFC placement as an Integer Linear Program (ILP) for minimizing latency in 5G network architectures, they employed the neural network to predict the number of instances, which facilitates scalable deployment. In [22], the authors proposed a general 5G network slicing framework to address VNF placement and set application-based latency limits for each SFC. However, it used a simple delay model that ignores the node processing and VNF queuing.

We also notice that offloading some services to the edge network or fog network for deployment have been considered by researchers, thereby reducing the overall service delay and the core network's load [17]. Huang et al. [23] regarded latency as a strict indicator and considered using soft actor-critic for optimization in the edge cloud. However, they focused on the DRL method and did not analyze SFC in depth enough. Jin et al. [14] used game theory and computing offload to propose a PSECO algorithm, which offloads VNFs to small base stations in fog network. PSECO can effectively handle delay-sensitive services and reduce the CAPEX and OPEX of internet service providers, but it ignored the limited resources in fog networks, which leads to a low deployment success ratio. Zamani et al. [16] considered the cloud-to-fog scenario in the Internet of Things network and proposed a SFC deployment solution, which aims at the bandwidth consumption of the IoT devices and minimizes the delay. They formulated the SFC deployment as an ILP problem and placed VNF in cloud or fog nodes, but the author did not point out the algorithm to solve the model.

2.2. Reliability

As a critical network functionality, SFCs are particularly vulnerable to disruptions and software failures. Therefore, devising a robust scheduling mechanism is essential for bolstering the reliability of SFCs. Kang et al. [24] proposed an optimization model based on VNF allocation of time slots, which can suppress service interruption caused by VM failure and VNF re-instantiation. The authors converted the SFC deployment problem into a MILP and come up with a heuristic method to maximize the SFC continuous available time. Zhai et al. [25] designed a VNF aggregation method to improve reliability and reduce resource consumption, but after the aggregated VNFs are placed, it is easy to cause the node overloaded result in entire SFC failed.

Redundant backup mechanisms are frequently used in actual works to deal with software failures. Herker et al. [26] proposed an algorithm to back up the entire SFC and embed it elastically into the data center, nonetheless, this work greatly increased the resource overhead. Zeng et al. [27]. A et al. did research on off-site backup of VNF. They backed up the same VNF and deployed it on different nodes, but did not give how to ensure the increase in delay and cost caused by enabling backup after the original VNF failed. Post-failure SFC functionality should be taken into account, and this increased reliability is still a cost sacrifice.

In addition to software failure, the reliability of SFC is also affected by hardware. As we mentioned above, most of the existing researches assume that the infrastructure of NFV is in an ideal state. Unlike [26, 27], Tang et al. [28] eschewed the backup redundancy approach. Instead, they leveraged the PageRank to ascertain node significance, while also taking into account both latency and reliability. However, the source and end points are ignored during VNF deployment, thus increasing the delay. Inspired by these works, we consider designing novel mechanisms to ensure service reliability without causing a surge in costs and delays.

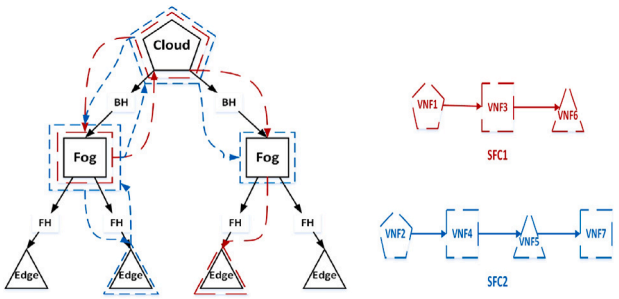
2.3. Cost

Most studies have addressed SFC costs starting from the flexible utilization of computing and network resources achieved by NFV. Costs can be divided into OPEX and CAPEX. The former is about the server and link resources, and the latter is related to costs energy consumption, license fees, VNF instantiation, etc. Cui et al. [29] proposed a SFC deployment approach based on user QoS and server resource-aware, it ensured users' multi-dimensional QoS requirements and operator costs in limited network resources by combining the simulated annealing algorithm and tabu search algorithm. However, the method did not perform very well in the scenario of large-scale networks. Beck et al. [30] proposed a CoordVNF algorithm, which can reasonably solve the VNF placement and resource allocation problems in large network scenarios, but the research only considered the operator's cost and ignored the user's QoS requirement. Chen et al. [31] took minimizing the constrained QoS cost as the main goal, and formulated the problem as ILP. Based on the hidden Markov model, they focused on the service deployment of large networks such as cloud computing and public cloud. There are also studies using reinforcement learning algorithms to optimize operating costs, mainly to optimize VNF placement costs. Luo et al. [32] utilized RNN and RL framework to reduce the overall cost of deploying SFC in distributed data center, here, RNN is tasked with traffic prediction, while RL is responsible for making deployment decisions.

2.4. Multi-objective optimization

Multi-objective optimization in SFC deployment frequently encounters the challenge of conflicting metrics. Zahedi et al. [33] proposed a method for reusing VNFs in cloud-fog computing to optimize latency and resource consumption. They reused preliminary VNF instances to facilitate flexible deployment of subsequent SFCs. However, they overlooked the fact that varying SFC lifecycles and the failure of reused instances or nodes could result in the unavailability of multiple SFCs. Xu et al. [34] proposed two strategies in their work. One is to deploy multiple backup instances of one or more VNFs in the SFC to ensure reliability, which is also a cost-effective solution. The other is to place multiple VNFs in a single SFC on the same node and reassemble the SFC, but it lacks restrictions on whether multiple VNFs can be placed on the same node, and it is prone to node overload and unevenness in small-scale networks. Han et al. [35] combined network state information to propose server and link balance factors to measure the degree of congestion, and embeds VNFs in the underlying network based on the proposed index. They also proposed a strategy in which SFCs of the same service back up each other. However, this solution did not take the reliability of the baseboard network into consideration. Shang et al. [36] considered optimization from a link perspective and proposed a Candidate Path Selection (OPS) algorithm to optimize the VNF placing and flow routing. It performed path selection and VNF placement through Randomized Rounding (RR), then merged the same VNF paths belonging to different SFCs as candidate paths, which effectively reduces the network congestion and cost-effectiveness. Nonetheless, their approach to assessing congestion primarily in cost terms limits its applicability for multi-objective joint optimization.

In essence, the existing research on multi-objective SFC deployment optimization often glosses over the interplay of multiple metrics or fails to thoroughly address the intricate relationships within large volumes of SFCs and VNFs in complex networks, thereby constraining the effectiveness of the methods used.



(a) Three-tier network architecture. (b) SFC request.

Fig. 1. Different SFCs deployment example in Cloud-Fog-Edge. The VNFs in the red and blue SFCs are respectively placed on different types of nodes. The colored arrows indicate the routing direction between different nodes.

3. Problem description and network model

This paper considers the SFC deployment issues in the three-tier network architecture shown in Fig. 1(a). All three networks are equipped with antennas to serve users or devices, the edge network can be connected to the fog through the fronthaul (FH) link, and the fog network connected to the cloud core network through the backhaul (BH) link can serve multiple edge networks. Moreover, the closer the fog or edge is to the cloud core network, the more computing resources it has and the cheaper it is, but at the same time it may be accompanied by an increase in transmission delay. Fig. 1(b) represents an example of the SFCs deployed in Fig. 1(a). Detailed deployment is driven by different metrics such as latency, reliability, cost and network congestion.

This section next models the physical network consisting of cloud, fog and edge access networks, then presents the SFC request from users to users, and finally elaborates the qualification conditions for SFC deployment to the physical network. Notations relevant to this paper are summarized in Table 1.

Table 1
Notations.

Params	Description
G^p	Physical network graph.
N^p, E^p	Set of physical nodes and links in G^p.
R_p	Set of physical network resource constraints.
$c_{n_p}, c^{use}_{n_p}, c^{ava}_{n_p}$	Total,occupied,available CPU resources of server n_i.
$s_{n_p}, s^{use}_{n_p}, s^{ava}_{n_p}$	Total,occupied,available storage resources of n_i.
ϵ_{n_p}	Unit cost of server n_p resources.
ζ_{n_p}	Unit storage cost of n_p.
re_{n_p}	The reliability of the node n_p.
k_{n_p}	All VNF types that the node n_p can host.
d_{l_p}	Link l_p's delay.
b_{l_p}	Bandwidth capacity of link l_p.
ϵ_{l_p}	Unit cost of link resources.
re_{l_p}	The reliability of the link l_p.
G_R	SFC requests graph.
N_R, E_R	Set of all VNFs and links of SFC G_R.
R_{vnf}	SFC deployment constraint.
L_{from}, L_{to}	Source and destination of SFC, respectively.
C_N, S_N	Sets of CPU and storage resource.
R_N	Bandwidth resource requirements set of VNFs.
D_{vnf}	Tolerable overall delay of the SFC request.
T_{vnf}	lifetime of an SFC path.
M	Deployment strategy including M_N and M_E.
$M(f_i), M(e_i)$	Deploy position of f_i, e_i.
$L^{M(f_i)}_{f_i}$	Whether f_i can be placed on $M(f_i)$ by position.
$F^{M(f_i)}_{f_i}$	Whether f_i can be placed on $M(f_i)$ by function.
$L^{M(e_i)}_{e_i}$	Whether e_i can be mapped on $M(e_i)$.
$In(n_p)$	Set of incoming links of n_p.
$Out(n_p)$	Set of outgoing links of n_p.
Del	Single request latency.
$Del(n), Del(l)$	Latency of a VNF/virtual link mapped to a node/path.
Rel	Single request reliability.
Cos	Total request cost including CAPEX and OPEX.
$CAPEX$	Virtual machine instantiation cost.
τ	Instantiation cost per unit VNF resource.
$OPEX$	Including node and path resource cost.
Obj	Objective function.
α, β, γ	Weight coefficients of respective indicators.
N_{f_i}	Set of candidate placement nodes for f_i.
r_{wl}	Wireless date transmission rate.
W_{wl}	The bandwidth of the channel.
$g^{wl}_{n^{wl}_{i,j}}$	Transmission power of access network n_i.
$p_{n^{wl}_{i,j}}$	Channel gain between n_i and n_j.
σ^2_{wl}	Gaussian noise power.
T_{now}, T_{begin}	The time when SFC was deployed and current time.
$\mathcal{G}_R, \mathcal{G}_{set}$	SFC queue and deployed successfully SFC set.
$\mathcal{G}_{ind}$	Indicator set of SFC.
Num_{fail}	The number of deployed-failed SFC.
N^{clo}, N^{foe}	Set of available cloud and fog or edge nodes.

3.1. Physical network

The physical network consists of multiple distributed fog or edge wireless access networks and centralized cloud networks. We represent it as an undirected weighted graph $G^p = (N^p, E^p)$, $N^p = \{n_i | i = 1, 2, \ldots, |Np|\}$ represents a set of physical nodes. Each node can host the specific types of VNF. $E^p = \{l_i | i = 1, 2, \ldots, |Ep|\}$ indicates the set of physical links. Define $R_p = (C^N, C^E)$ as a set of physical network resource constraints, for any physical node $n_p \epsilon N^p$, where C^N indicates the properties of a physical node, e.g., $(c_{n_p}, c^{use}_{n_p}, c^{ava}_{n_p})$ and $(s_{n_p}, s^{use}_{n_p}, s^{ava}_{n_p})$ for the total, occupied, and available CPU and storage resources of the server, respectively. In addition, ϵ_{n_p} and ζ_{n_p} are the unit cost of CPU and storage resources, re_{n_p} represents the reliability of the node, and $k_{n_p} = \{k_i | i = 1, 2, \ldots, np\}$ represents all VNF types that n_p can host. C^E indicates the properties of a physical link, including link delay d_{l_p}, bandwidth capacity b_{l_p}, unit cost of link resources ϵ_{l_p} and the reliability of the link re_{l_p}.

3.2. SFC requests

The SFC request are modeled as undirected weighted graph $G_R = (N_R, E_R)$, where $N_R = \{f_i | i = 1, 2, \ldots, |Nr|\}$ means the collection of VNFs in a SFC request from a user to another user, and $E_R = \{e_i | i = 1, 2, \ldots, |Er|\}$ indicates the set of all SFC links. Define $R_{vnf} = (L_{from}, L_{to}, C_N, S_N, C_E, R_N, T_{vnf}, D_{vnf})$ as a deployment constraint, where L_{from} and L_{to} respectively denote the source and destination of request, $C_N = \{r(f_i) | i = 1, 2, \ldots, |Nr|\}$ and $S_N = \{s(f_i) | i = 1, 2, \ldots, |Nr|\}$ are the set of computing and storage resource requirements of current SFC request, $R_N = \{re(f_i) | i = 1, 2, \ldots, |Nr|\}$ is the reliability set of virtual machines instantiated by VNFs, $C_E = \{r(e_i) | i = 1, 2, \ldots, |Er|\}$ indicates the set of bandwidth resource requirements of all SFC links, D_{vnf} and T_{vnf} represent the tolerable overall transmission delay of the physical path hosting the request and the lifetime of an SFC path, respectively.

3.3. SFC deployment model

This paper models the deployment decision-making process of SFC as:

$$MO - SACO : (G^p, G_R) \rightarrow (N^{P^*}, C^{N^*}, E^{P^*}, C^{E^*}), \tag{1}$$

where $MO - SACO$ includes the placement strategy $MO - SACO_N$ for VNF and the mapping method $MO - SACO_E$ for virtual links. For convenience, in the following text, we will use M to simply represent strategy $MO - SACO$. Specifically, $M_N : (N^p, N_R) = \{M(f_i) | i = 1, 2, \ldots, |Nr|\}$ and $M_E : (E^p, E_R) = \{M(e_i) | i = 1, 2, \ldots, |Er|\}$, $M(f_i)$ and $M(e_i)$ respectively represent the deployment location of a certain VNF and virtual link in current request. $N^{P^*} \subset N^p$ and $E^{P^*} \subset E^p$ refer to a subset of physical nodes and physical paths, respectively. C^{N^*} and C^{E^*} represent the CPU resources of the physical server and the bandwidth resources allocated to a SFC request.

During the deployment process, both VNFs and virtual links should follow certain restrictions. Firstly, in terms of M, each VNF is placed on a node, and relevant virtual link is mapped to the underlying link, that is:

$$M(f_i) \in N^p, M(e_i) \in E^p, \forall f_i \in N_R, \forall e_i \in E_R. \tag{2}$$

We introduce two binary variables to indicate the location constraints and functional constraints of VNF placement, which is given by:

$$L^{M(f_i)}_{f_i} \in \{0, 1\}, F^{M(f_i)}_{f_i} \in \{0, 1\}, \forall f_i \in N_R. \tag{3}$$

The required resources of the VNF placed to the node should not exceed its remaining resources. Similarly, the bandwidth also have to meet this constraint, we have:

$$c(s)^{ava}_{M(f_i)} \geq \sum_{M(f_i)} L^{M(f_i)}_{f_i} F^{M(f_i)}_{f_i} r(s)(f_i), \forall f_i \in N_R. \tag{4}$$

$$b_{M(e_i)} \geq \sum_{M(e_i)} L^{M(e_i)}_{e_i} r(e_i), \forall e_i \in E_R. \tag{5}$$

where $L^{M(e_i)}_{e_i}$ is the binary constraint of the virtual link mapping. Then each VNF can only be placed on one substrate node and cannot be divided, we have:

$$\sum_{n \in N^p} |M(f_i)| = 1, \forall f_i \in N_R. \tag{6}$$

Traffic should be routed in order, we indicate $In(n_p)$ and $Out(n_p)$ as the set of incoming and outgoing links of node n_p, that is:

$$\sum_{l_p \in In(n_p)} L^{l_p}_{e_i} - \sum_{l_p \in Out(n_p)} L^{l_p}_{e_i} = L^{n_p}_{f_{i+1}} F^{n_p}_{f_{i+1}} - L^{n_p}_{f_i} F^{n_p}_{f_i}, \quad \forall n_p \in N^p, \forall l_p \in E^p \tag{7}$$

In our strategy, we define the evaluation indicators and calculation method of latency, reliability, cost and load, and they are calculated as follows:

(1) *Latency*: As for the user-to-user SFC deployment delay in decision, M, it can be divided into processing delay, queuing delay, propagation delay and sending delay. Since our working scenario is a distributed data center with a certain scale, after our estimation, compared with the propagation delay and queuing delay, the sending delay can be ignored, and the processing delay will not be differentiated by different strategies. Therefore the delay is defined as:

$$\begin{aligned} Del &= \sum_{f_i \in N_R} Del(f_i) \\ &= \sum_{f_i \in N_R} Del(M(f_i)) + \sum_{e_i \in E_R} Del(M(e_i)), \end{aligned} \tag{8}$$

where $Del(f_i)$ includes the queuing delay $Del(M(f_i))$ and all link delays associated with the previous VNF f_{i-1}. Since network scenarios involve cloud, fog, and edge devices, the specific calculation methods for latency $Del(M(e_i))$ may be different, their specific definitions are elaborated in slowromancapiv@-B.

(2) *Reliability*: In general, reliability is defined as the probability of a system providing standard services within a certain period of time without any failures. It is usually quantified in terms of time. But for SFC, in addition to nodes and links, the reliability of all VNFs that make up the SFC should also be considered. The reliability of a SFC can be given by:

$$Rel = \prod_{i}^{|N_r|} re_{f_i} = \prod_{i}^{|N_r|} re(f_i) \cdot re_{M(f_i)} \cdot re_{M(e_i)}, \tag{9}$$

where re_{f_i} is the probability of normal operation of a single VNF, calculated by the product of reliability of VNF, node and link.

(3) *Cost*: The cost is divided into CAPEX and OPEX. In this paper, we consider CAPEX, which is represented by the instantiation cost of the virtual machine, and OPEX, which includes the resources cost of server and link. Hence the cost is defined as:

$$\begin{aligned} &Cos = CAPEX + OPEX \\ &CAPEX = \sum_{f_i \in N_R} L_{f_i}^{M(f_i)} \cdot F_{f_i}^{M(f_i)} \cdot r(f_i) \cdot \tau \\ &OPEX = \sum_{f_i \in N_R} OPEX(M(f_i)) \\ &+ \sum_{e_i \in E_R} OPEX(M(e_i)) \\ &= \sum_{i=1}^{Nr} L_{f_i}^{M(f_i)} \cdot F_{f_i}^{M(f_i)} \cdot r(f_i)(\varepsilon_{M(f_i)} + \zeta_{M(f_i)}) \\ &+ \sum_{e_i \in E_R} L_{e_i}^{M(e_i)} \cdot r(e_i) \cdot \varepsilon_{M(e_i)}, \end{aligned} \tag{10}$$

where τ is the instantiation cost per unit VNF resource.

To compare cost, latency, and reliability, we normalize each metric as follows,

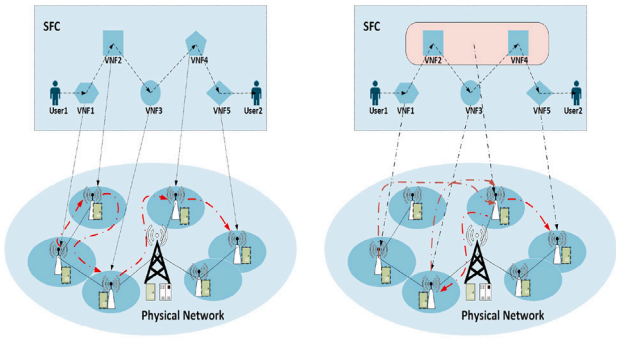
$$X' = \frac{X}{Base_X}, \tag{11}$$

where X is the indicator we need to calculate, and $Base_X$ is the corresponding result obtained through the baseline algorithm. With these definitions, the objective function can be defined as:

$$Obj = \alpha Del' - \beta Rel' + \gamma Cos' \tag{12}$$

where α, β, γ are the weight coefficients of respective indicators, which can be adjusted according to the actual QoS requirements of the business, $\alpha+\beta+\gamma = 1$. Hence, Our target is to find a policy that minimize the objective function:

$$M^* = \arg\min_{M} Obj. \tag{13}$$



(a) Traditional deployment. (b) VNF aggregation deployment.

Fig. 2. Comparison of traditional VNF deployment methods and VNF aggregation methods.

4. SFC deployment optimization

This study aims to solve the following problems: Jointly optimize the delay, reliability, cost and node load of SFC requests by placing VNFs and routing links reasonably. This section focuses on three steps to optimize the above performance.

4.1. VNFs aggregation decision

The common SFC deployment method is to place each VNF in the baseboard network and route traffic sequentially. Its focus is on the one-to-one relationship between VNFs and nodes, as shown in Fig. 2(a), which usually ignores the preprocessing of SFC and does not fully consider the deployment constraints of VNFs. Therefore, in order to make up for the shortcomings of existing work, we analyze the composition of SFCs and design a novel VNF aggregation rule to process SFCs waiting to be deployed. Specifically, for each VNF, We note that not all VNFs can be arbitrarily mapped to every physical node, due to functional differences between physical nodes, they can only carry a specific type of VNF, i.e. functional constraints. At the same time, each physical node may be distributed in multi-domains, thus the SFC deployment also should meet the location constraints of each VNF to avoid excessive transmission delay caused by too long links. We first ensure that it meets the placed constraints of resource, function and location, then the candidate node set of the current function can be obtained through

$$N_{f_i} = \{n_p | c_{n_p}^{ava} \geq r(f_i), F_{f_i}^{n_p} = L_{f_i}^{n_p} = 1\} \tag{14}$$

Secondly, we found that there are duplicate VNFs in a considerable number of existing SFCs, such as VOIP services (NAT-FW-TS-FW-NAT), Web services (FW-LB-NAT-LB-Webserver), etc. In this case, we consider aggregating VNFs that perform the same function inside the SFC and placing them at the same node, as shown in Fig. 2(b), assuming that VNF2 and VNF4 perform the same function, and there is intersection N_{f_i,f_j} in their respective candidate node sets, that is

$$\exists N_{f_i,f_j} = N_{f_i} \cap N_{f_j}. \tag{15}$$

Subsequently, the original request G_R can be converted into a temporary SFC G'_R with $N_R = \{f_1, f_{2,4}, f_3, f_5\}$. For $f_{2,4}$, since the VNF just executes the same function twice, we have $r(f_{2,4}) = max(r(f_2), r(f_4))$, $s(f_{2,4}) = max(s(f_2), s(f_4))$ to ensure that the necessary functions will not be missing.

Theoretically, according to (8)–(10), G'_R will have a VNF performance improvement of nearly 20% compared to G_R, reducing the number of VNFs in the same SFC will mean savings in virtual machine instantiation costs and node resource costs. Then, only links need to be considered when calculating the reliability of VNF4, the overall

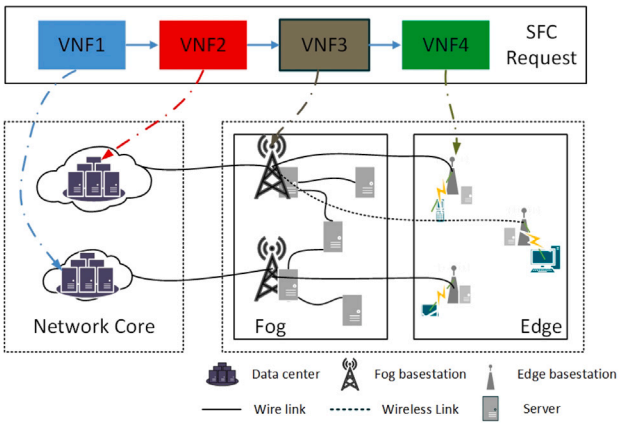


Fig. 3. VNFs choose a cloud network, a fog access network or an edge access network to deploy.

reliability will also be improved. However, the update of the request pattern means a change of deployment location. We cannot assert that all target QoS are improved but express VNF aggregation as a decision, that is, choosing the strategy with a smaller objective function between aggregation placement and dispersion placement. It is denoted as:

$$\begin{aligned} Obj_{agg} &= \alpha(Del_{agg} - Del(f_j)) - \beta Rel/(re(f_j) \cdot re_{M(f_i)}) \\ &\quad + \gamma(Cos_{agg} - Cos_N(M(f_j)) \\ Obj^* &= min(Obj, Obj_{agg}), \end{aligned} \tag{16}$$

where N_{f_i,f_j} represents the intersection of candidate nodes of VNF2 and 4, and Obj_{agg} is the new objective function calculated through the aggregation strategy.

It is worth mentioning that, E_R still maintains the original virtual link order, so the propagation delay and link cost do not necessarily increase due to the aggregation decision. Therefore, we further introduced fog node and edge devices for further optimization.

4.2. VNF offloading decision

Since the edge device and fog network are closer to the user's location, and there are a certain amount of CPU resources and forwarding resources, some VNFs in the SFC can be offloaded to the edge network or fog network for mapping and deploying, which shown in Fig. 3. Edge and fog serve as the complement to the clouds, can effectively reduce the end-to-end delay and the data center loading, first, it alleviates the high queuing delay caused by the high load of cloud nodes; second, when two or more VNFs are placed in the same cloud network domain, it also reduces the propagation delay between VNFs. The third and more wonderful point is that the offloading decision can have some subtle connections with our aggregation decision. In Fig. 2(b), after VNF2 and 4 are aggregated, the aggregated VNF changes the link shape of the SFC. There is a round-trip path between the new VNF2,4 and VNF3, and placing the VNF in the access network will also generate round-trip time, this just makes up for it. Subsequent experimental results also verified the feasibility of our strategy. However, due to the relative scarcity of fog resources and edge resources in the access network, this study only allows each SFC to request the use of fog/edge computing resources of its own user access network.

The arrival process of SFC on the node conforms to the Poisson distribution, and the service time conforms to the exponential distribution, so we utilize the M/M/1 model to calculate the processing delay of each VNF,

$$Del(M(f_i)) = \frac{1}{c^{ava}_{M(f_i)}/r(f_i) - L^{M(f_i)}_{f_i} F^{M(f_i)}_{f_i} \lambda + \rho}, \tag{17}$$

where λ represents the arrival rate of SFC, and ρ is an infinitesimal positive number to avoid the situation where the denominator of the formula is 0.

Since the fog edge network is included in the three-layer structure, both wired and wireless transmission methods should be considered [37]. Under the condition that the constraints are met, when the VNF is deployed in the cloud network, the deployment path $M(e_i)$ from the VNF f_i to the previous VNF f_{i-1} is found and its delay $Del(M(e_i))$ is obtained. Then the $Dijkstra$ algorithm is applied for tracing the path $p_{f_i,to}$ from the VNF f_i to user L_{to} and obtain its corresponding delay by:

$$Del(p_{f_i,to}) = \sum_{l_p \in p_{f_i,to}} d_{l_p}. \tag{18}$$

Therefore, the total delay of VNF f_i deployed in the cloud network is

$$Del(f_i)^{clo} = Del(M(e_i)) + Del(p_{f_i,to}) + Del(M(f_i))^c. \tag{19}$$

When the VNF is deployed in a fog or edge device, we also find the path $M(e_i)$, but unlike in the cloud network, in addition to considering the path $M(e_i)^{clo}$ between the cloud networks, the path $M(e_i)^{foe}$ between the fog/edge and the cloud network is also part of $M(e_i)$. The delay of path $M(e_i)^{clo}$ is $Del(M(e_i)^{clo})$, as for $M(e_i)^{foe}$, due to different placement locations, there will be differences in delay components. If it is a fog node, the delay $Del(M(e_i)^{c2f})$ is determined by the optical cable; if it is an edge device, in order to facilitate calculation, in addition to the cloud and fog, wireless transmission is used between edge devices and between devices and fog nodes, the data transmission rate between an access network is given as

$$r_{wl} = W_{wl} \cdot \log_2(1 + \frac{g_{n^{wl}_{i,j}} * p_{n^{wl}_{i,j}}}{\sigma^2_{wl}}), \tag{20}$$

where W_{wl} is the bandwidth of the channel, $p_{n^{wl}_{i,j}}$ is the access network n_i's transmission power, which is determined by data transmission power control mechanism. $g_{n^{wl}_{i,j}}$ denotes channel gain between access network n_i and n_j. σ^2_{wl} is the Gaussian noise power inside the channel. Hence, the propagation delay between cloud and fog or edge be obtained as

$$Del(M(e_i)^{foe}) = \begin{cases} Del(M(e_i)^{c2f}), on\ fog, \\ Del(M(e_i)^{c2f}) + \dfrac{r(f_i)}{r_{wl}}, edge\ device, \end{cases} \tag{21}$$

We can choose the communication method with lower latency. Then, when VNF f_i is deployed in a fog or an edge access network, its delay can be given as

$$\begin{aligned} Del(f_i)^{foe} &= Del(M(e_i)^{clo}) + Del(p_{f_i,to}) \\ &\quad + Del(M(e_i)^{foe}) + Del(M(f_i))^{foe} \end{aligned} \tag{22}$$

As a result, the VNF is deployed in different network environments, corresponding to different delay. Moreover, reliability and cost will also vary depending on the network environment, we then select the VNF deployment location by $min(Obj^{clo}, Obj^{foe})$.

4.3. SFC uninstall and resource release

In order to prevent SFCs that have been deprecated by users or beyond their lifetime from occupying server resources continuously, we set a deployment daemon, which monitors the requests that meet the above criteria:

(1) The request normally exceeds its initial stated lifetime, i.e.

$$T_{now} - T_{begin} \geq T_{vnf}, \tag{23}$$

where T_{begin} and T_{now} represent the time when the SFC was successfully deployed and the time now, respectively.

(2) Processing latency will increase due to an increase in the number of underlying network services, causing the current SFC path delay to exceed the initial requirement, i.e.

$$\sum_{G_j \in \mathcal{G}_{set} \setminus G_i} \sum_{f_k \in G_j} \sum_{M(f_k) \in M(G_i)} Del(M(f_k)) + Del(G_i) \geq D_{vnf}, \tag{24}$$

where G_i represents the current request, $\mathcal{G}_{set}$ represents the set of successfully deployed requests, and $M(G_i)$ represents the set of nodes placed by all VNFs in G_i.

Then these requests are uninstalled and resources are released. Besides, in actual network environments, once the server load exceeds 80%, indicating that the resources about to be exhausted and cannot respond to user requests in a timely manner. For convenience, we scale the initial total resources of the node n_p to its 80%. i.e.

$$c_{n_p} \leftarrow c_{M(f_i)} \times 80\%, \quad c^{ava}_{M(f_i)} = c_{n_p} - c^{use}_{M(f_i)}, \quad \forall f_i \in N_R. \tag{25}$$

5. Algorithm design

For an SFC request with latency and reliability requirements, achieving efficient deployment of VNFs is an NP-Hard problem, which is difficult to solve in polynomial time. Hence, a service function chain orchestration strategy based on function aggregation and VNF offloading migration is proposed.

5.1. MO-SACO algorithm

In order to prove the superiority of our algorithm, for each SFC request G_i, we evaluate its performance according to the three indicators of Eqs. (8)–(10). Also, for requests that have already been processed, we also take the deployment success rate into account. A detailed description of the MO-SACO algorithm is shown in Algorithm 1. First, we take the information of the physical network G^p, R^p and SFC requests $\mathcal{G}_R$ coming in a Poisson process as input, and take the set of successfully deployed requests $\mathcal{G}_{Set}$, the set of indicators for each request $\mathcal{G}_{ind}$, and the number of failed deployment requests Num_{Fail} as output and initialize. At each time t, MO-SACO inspects the SFCs that have exceeded their lifecycles from $\mathcal{G}_{Set}$ and unloads them directly from the infrastructure, it also clears requests that do not meet the latency requirements, and then releases the corresponding physical network resources in time for new request deployment, as shown in lines 2–9 of Algorithm 1.

Algorithm 1 MO-SACO Algorithm.

Input: Physical network $G^p = (N^p, E^p)$, network constraints $R^p = (C^N, C^E)$, SFC requests queue $\mathcal{G}_R = \{G_i | i = 1, 2, ..., n\}$;
Output: $\mathcal{G}_{Set} = \{F_j | j = 1, 2, ..., m\}$, indicator set $\mathcal{G}_{ind}$, deployment-failed SFC number Num_{Fai};

1: Initialize $\mathcal{G}_{Set} = \emptyset$, $\mathcal{G}_{ind} = \emptyset$, Num_{fail} =0;
2: **for** t=1, ..., T **do**
3: **if** $\mathcal{G}_{Set}! = \emptyset$ **then**
4: **for** $G_R \in \mathcal{G}_{Set}$ **do**
5: Uninstall the SFC G_R if $t \geq T^{G_R}_{begin} + T_{vnf}$;
6: Clear delay exceeding request by Eq. (24);
7: Update network resources G^p;
8: **end for**
9: **end if**
10: **for** $G_R \in \mathcal{G}_R$ **do**
11: Initialize SFC candidate node set $N_{G_R} \leftarrow \emptyset$;
12: **for** $f_i \in G_R$ **do**
13: Find the candidate node set N_{f_i} by Eq. (14);
14: **end for**
15: **if** $\exists N_{f_i} \cap N_{f_j}$ and $F_{f_i} == F_{f_j}$ **then**
16: Update SFC shapes to get $G_{R_{agg}}$;
17: Virtual deploy $G_{R_{agg}}$;
18: Get the G^{Del}_{agg}, G^{Cos}_{agg} and G^{Rel}_{agg} respectively;
19: Calculate the Obj_{agg} by Eq. (16);
20: Compare with MILP deployment strategy;
21: Select aggregated or not by $min(Obj, Obj_{agg})$;
22: **end if**
23: Call the sub-algorithm 2;
24: Generate $G^{Del}, G^{Cos}, G^{Rel}, Obj^*_{G_R}$;
25: Get VNF offloading result M_{G_R};
26: **if** deploy successfully **then**
27: $\mathcal{G}_{set} \leftarrow \mathcal{G}_{set} \cup G_R$;
28: $\mathcal{G}_{ind} \leftarrow \mathcal{G}_{ind} \cup \{G^{Del}, G^{Cos}, G^{Rel}\}$;
29: **else**
30: $Num_{Fai} \leftarrow Num_{Fai} + 1$;
31: **end if**
32: $\mathcal{G}_R \leftarrow \mathcal{G}_R \setminus G_i$;
33: **end for**
34: **end for**
35: **return** $\mathcal{G}_{Set}, \mathcal{G}_{ind}, Num_{Fai}$;

Subsequently, lines 10–22 of Algorithm 1 are our VNF aggregation decision. MO-SACO processes each pending SFC request in $\mathcal{G}_R$ in order. For the current SFC G_R, it finds the candidate placement node set N_{G_R} for all VNF based on the rules we defined in slowromancapiv@-A, and determines whether there is an intersection among the node set N_{G_R} of VNFs that perform the same function. If so, it calculates the corresponding metrics for the aggregated request $G_{R_{agg}}$ and compares them with the MILP deployment to return a more optimal deployment scheme. Since the VNF offloading decision needs to continue to be executed, G_R is not actually placed in the substrate network at this time. It should be noted that as the aggregation decision is completed and selected, the shape of the SFC is also updated G_R by $G_{R_{agg}}$.

Then in line 23–34, the VNF offloading sub-algorithm, which is shown in Algorithm 2, is invoked to perform node-mapping and link-routing for each VNF f_i. If deploys successfully, the sub-algorithm will return a scheme M_G with delay G^{Del}, reliability G^{Rel} and cost G^{Cos}. Using the return value to update the set $\mathcal{G}_{Set}$ and $\mathcal{G}_{ind}$. Otherwise, the value of Num_{Fail} which represents the number of fail-deployed SFC is added with one.

5.2. VNF offloading algorithm

The VNF offloading algorithm sequentially processes all VNFs in a SFC request. In line 2–5, for each VNF f_i, it divides the set of candidate nodes N_{f_i} into two collections, the available cloud network N^{clo} and the available fog network or edge device N^{foe}, and initializes the temporary variables measuring the delay G_i^{Del},cost G_i^{Cos}, reliability G_i^{Rel} of f_i.

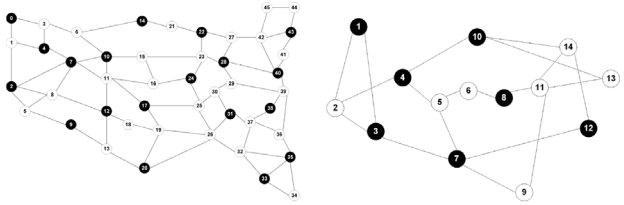
Algorithm 2 VNF Offloading Algorithm.

Input: Single SFC request G_R, constraints R_{vnf}.
Output: Deployment strategy M_{G_R} with delay G^{Del}, cost G^{Rel} ,reliability G^{Rel}.

1: Init $M_{G_R} = \emptyset, G^{Del} = G^{Cos} = 0, G^{Rel} = 1$;
2: **for** each $f_i \in G_R$ **do**
3: Divide N_{f_i} into N^{clo} and N^{foe};
4: Initialize $G_i^{Del} = G_i^{Cos} = \infty, G_i^{Rel} = 0$;
5: $Obj_i = \alpha \cdot G_i^{Del} - \beta \cdot G_i^{Rel} + \gamma \cdot G_i^{Cos}$;
6: **for** each $n_j \in N^{foe}$ **do**
7: Assumed place f_i onto fog/edge device n_j;
8: Get the queueing delay $Del(n_j)$ by Eq. (17);
9: Find the path $M(e_i)^{clo}$ and $M(e_i)^{foe}$;
10: Obtain $Del(M(e_i)^{clo}, Del(M(e_i)^{foe}$;
11: Use Dijkstra to get the path $p_{f_i,to}$;
12: Get $Del(f_i)^{n_j}$ by Eq. (19);
13: Calculate $Cos(f_i)^{n_j}$, $Rel(f_i)^{n_j}$ get Obj_i^{foe};
14: $Obj_i = min(Obj_i, Obj_i^{foe})$;
15: **end for**
16: **for** each $n_k \in N^{clo}$ **do**
17: Suppose place f_i onto cloud server n_k;
18: Get the corresponding delay by Eq. (22);
19: Calculate $Cos(f_i)^{n_k}$, $Rel(f_i)^{n_k}$ get Obj_i^{clo};
20: Decide to update $Obj_i = min(Obj_i, Obj_i^{clo})$;
21: **end for**
22: Update G_i^{Del}, G_i^{Cos} and G_i^{Rel}
23: $M_{G_R} \leftarrow M_{G_R} \cup \{f_i, M(f_i), M(e_i)\}$;
24: $G^{Del} \leftarrow G^{Del} + G_i^{Del}$;
25: $G^{Cos} \leftarrow G^{Del} + G_i^{Cos}$;
26: $G^{Rel} \leftarrow G^{Rel} \cdot G_i^{Rel}$;
27: Update resources of substrate networks;
28: **end for**
29: **return** $M_{G_R}, G^{Del}, G^{Cos}, G^{Rel}$;

Based on the current available fog or edge resources, the VNF f_i is mapped to the physical server n_j to obtain the process delay $Del(n_j)$ by Eq. (17). Then algorithm 2 finds the path $M(e_i)$, as can be seen from the above, the path $M(e_i)$ includes the cloud network path $M(e_i)^{clo}$ from the current VNF f_i to the previous VNF f_{i-1}, and the access network path $M(e_i)^{foe}$. If the current VNF is the first one, algorithm 2 finds the link from the current VNF f_1 to the original user L_{from}. Then we can get the path $p_{f_i,to}$, and calculate the propagation delay $Del(p_{f_i,to})$. If the path Obj_i^{foe} obtained by the VNF f_i deployed to the current server n_j are smaller than those of the previous server n_{j-1}, the algorithm updates the delay, cost, reliability. This procession is shown in line 6–15.

Similarly, in line 16–21, we assume that VNFs are deployed in cloud servers N^{clo} and communicate by wire totally. For server n_k, we can get paths $M(e_i)$ and $p_{f_i,to}$, then calculate the Obj_i^{clo}. Then in line 22–27, after traversing all cloud servers, the optimal VNF f_i deployment result can be obtained among all cloud servers, further the optimal strategy between cloud and access network to deploy the current VNF f_i can be found. We finally obtain the total optimal delay, cost and reliability of the current SFC deployment scheme M_{G_R}. The detailed algorithm is shown in Algorithm 2.



(a) USANET topology. (b) NSFNET topology.

Fig. 4. Network topology.

Table 2
Simulation scenario settings.

Parameter	cloud	fog	edge
c_{n_p}	U[80,120]	U[50,70]	U[10,30]
s_{n_p}	U[120,150]	U[60,90]	U[20,40]
b_{l_p}	U[60,80]	U[30,50]	U[10,20]
ε_{n_p}	U[0.6,1]	U[0.6,1]	U[0.6,1]
ζ_{n_p}	U[0.6,0.8]	U[0.3,0.5]	U[0.1,0.2]
ε_{l_p}	U[1,1.5]	U[0.3,0.5]	U[0.1,0.2]
re_{n_p}	U[0.98,0.99]		
re_{l_p}	U[0.96,0.98]		
d_{l_p}	U[2,4]		
$re(f)$	U[0.96,0.99]		
$\|Nr\|$	U[4,8]		
$r(f)$	U[5,10]		
$r(e)$	U[5,10]		
$s(f)$	U[3,6]		
τ	U[0.2,0.4]		
α, β, γ	0.33		

6. Simulation

6.1. Simulation environment settings

6.1.1. Simulation environment

We build the simulation environment on a server with CPU 12 × 2.10 GHz, RAM 128.0 GB, disk capacity 3.6 TB, and OS *ubuntu*20.04. The *Containernet*, which is a tool that integrates Mininet and Docker container, is used to deploy the physical network. The software environment is: *Python*3.6 and open source package *pyomo*2.5.0. Considering that the simulation environment needs to jointly schedule the resources of cloud network, fog network and edge device, the physical network presence point consists of multiple centralized cloud nodes and multiple distributed fog or edge nodes. Therefore, as shown in Fig. 4, We selected two representative topologies USANET and NSFNET to simulate network environments of different sizes. Assume that the black nodes in Fig. 4 are connected to the fog or edge access network like Fig. 1(a).

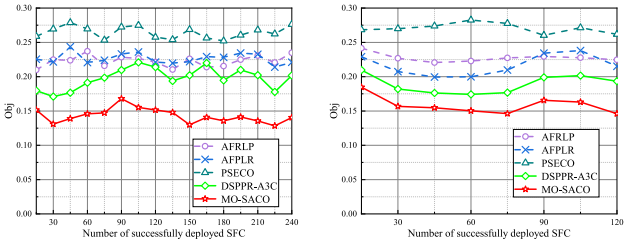
6.1.2. Parameter settings

Since three types of networks are involved in our research, we list the detailed parameters of the network and VNF in Table 2. Besides, within the access network, the wireless channel bandwidth $W = 20$ MHz, the transmission powers $p_{n_i,n_j} = 0.5$ W, the background noise $\sigma^2 = 2 \times 10^{-13} w$. According to the wireless channel model for radio environment, we set channel gain $g_{n_i,n_j} = 127 + 30 log d$, where d is the coverage area of the wireless signal.

For SFC settings, suppose each node carries three types of VNF randomly, and the VNF type that can be carried by the fog or edge server connected to the cloud network is the same as that of the cloud server. We assume that there are 10 types of VNFs in total. Assuming that each SFC request arrives dynamically according to Poisson process with parameter $\lambda = 1/20$ and its lifetime subject to exponential distribution with parameter $\mu = 1000$. The simulation time is 10 000 time units.

Table 3
Description of five methods.

Method	Description
MILP-MUTI	Baseline follow the greedy strategy, it first places the VNF on nodes that minimizes the objective function, then finds the shortest path to the previous VNF and route traffic.
AFRLP	AFRLP first completes the aggregation decision, and then all paths that satisfy the delay are found. Finally it selects the optimal reliability path and places the VNFs in order.
AFPLR	At the beginning, AFPLR completes the aggregation decision and finds all delay-allowed mapping paths. Then, it deploys VNFs according to the criterion with the maximum node reliability and the path with the highest node reliability.
DSPPR-A3C	DSPPR-A3C provides SFC with information about VNFs that need to be backed up and multiple VNFs that need to be deployed on the same node through the VNF queue network, which is configured by the queue memory and LSTM. It also utilize A3C to accelerate the training process.
PSECO	PSECO first embeds the VNFs in the SFC into the substrate network in sequence and routes traffic. Then, when it encounters VNFs that cannot meet resource constraints, it offloads the VNFs that occupy the most physical resources among the deployed requests to the fog.



(a) Optimization object in USANET. (b) Optimization object in NSFNET.

Fig. 5. Object value result.

6.1.3. Comparison algorithm

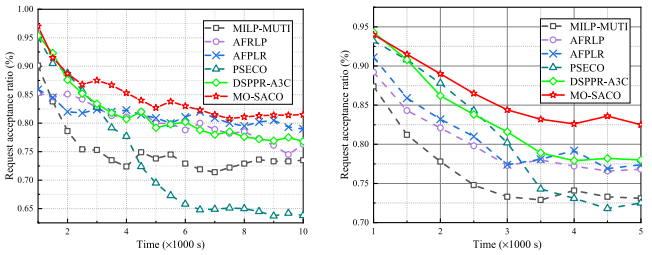
In order to prove the effectiveness of the algorithm proposed, We use a greedy algorithm MILP-MUTI, which regard the objective function defined in slowromancapiii@-C as a baseline and compare it with the following four strategies: two improved algorithms [33] with our aggregation strategy, namely Aggregate and First Routing Last Placement (AFRLP), Aggregate and First Placement Last Routing (AFPLR), DSPPR-A3C algorithm [34], and PSECO algorithm [14]. The description of the five algorithms is shown in Table 3. In order to reduce the interference of random factors, we arrived at the final result by repeating the experiment many times.

6.2. Evaluation results and discussion

Fig. 5 illustrates the Obj results of our MO-SACO and four contrasting algorithms relative to baseline MILP-MUTI. The results show that the MO-SACO algorithm stands out among them and achieves excellent results in networks of different sizes. The specific results of request acceptance rate, delay, cost and reliability will be shown in Figs. 6–9.

Fig. 6 illustrates the relationship of the request acceptance ratios over time for the six methods in USANET and NSFNET. The figure shows that, under the condition of sufficient network resources and bandwidth in the initial stage, the deployment success rate of the six methods is maintained at a high level, but as the number of incoming SFC requests increases, the acceptance rate shows a clear downward trend due to saturation of infrastructure.

AFRLP algorithm completes the placement of VNFs in all paths that meet the conditions, while AFPLR algorithm first places VNFs and then routes traffic to obtain deployment paths. Therefore, server



(a) SFC acceptance ratio in USANET. (b) SFC acceptance ratio in NSFNET.

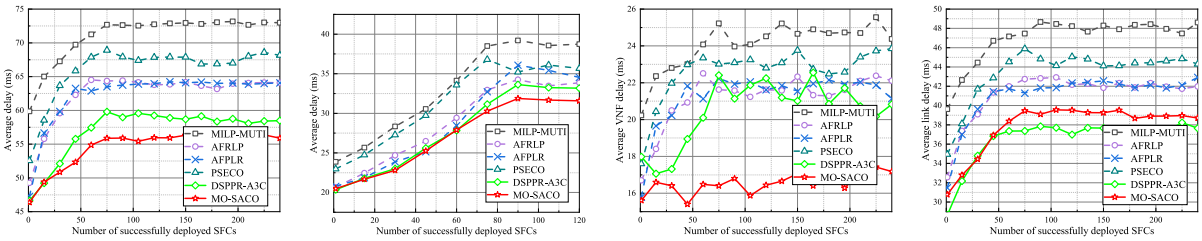
Fig. 6. Acceptance ratios.

resources and bandwidth resources become the bottlenecks of the request acceptance rate of algorithms AFRLP and AFPLR, respectively. PSECO algorithm offloads all SFCs with the largest proportion to the fog network after the cloud network cannot carry new requests. However, in the face of large-scale SFCs, PSECO cannot fully deploy requests in the cloud or fog. As a result, its performance is slightly worse than the baseline algorithm MILP. Due to backup mechanism and immature joint VNF placement scheme of DSPPR-A3C, it may increase the risk of node overload and uneven load when network resources are relatively limited. Our MO-SACO algorithm jointly utilizes the resources of cloud, fog and edge devices to increase the possibility of requests being placed. In addition, with the help of VNF aggregation and offloading decisions, MO-SACO exerts the superiority of cloud-fog collaboration and is slightly better than USA in the relatively resource-constrained NSF, which can be seen from Fig. 6(a) and (b).

Fig. 7(a) and (b) show the relationship between latency and the number of successfully deployed SFCs in USA and NSF for the six methods, respectively. With the requests number increases, a considerable part of CPU resources and bandwidth resources are occupied, and the number of hops of the shortest delay path between VNFs also rises, which indirectly causes the increase of queueing delay and propagation delay, and gradually reaches a saturated state. But in relatively large-scale networks, the effect of MO-SACO is better than other methods. Sufficient network resources help it choose the optimal decision. We also calculated the delay of VNF and link, as shown in Fig. 7(c) and (d).

In Fig. 7(c), it is worth mentioning that the advantages of AFPLR, AFRLP over MILP-MUTI, PSECO prove that the VNF aggregation decision does play a role in reducing node queuing delay. MO-SACO reduces the number of VNFs and node load compared to the DSPPR-A3C, and reflected in the results. In Fig. 7(d), MO-SACO does not achieve optimal performance because the joint placement of DSPPR-A3C results in a reduction in the number of SFC links. However, Our method still has advantages over others, which proves that the VNF aggregation decision appropriately compensates for the round-trip delay within the hybrid network.

Fig. 8 shows the effects of six SFC deployment strategies in two network topologies. After being improved by our aggregation strategy, the performance of the AFPLR and AFRLP algorithms is significantly better than that of the MILP-MILP and PSECO algorithms. When placing VNFs, PSECO method does not affect the number of placed nodes, and as a result, the node reliability is not significantly improved. AFRLP places a higher priority on reliability, and the DSPPR-A3C algorithm directly reduces the number of mapped links of SFC by jointly placing VNFs, resulting in an improvement in request reliability. The results of the two schemes are also comparable. Different from DSPPR-A3C, MO-SACO proposed in this article focuses on reducing the number of VNFs. After aggregation, the layout of SFC may be changed, and with the help of the advantages of cloud-fog-edge collaboration and offloading decision-making, the number of intermediate links that requests actually pass through is reduced, as expected, MO-SACO produces better performance in terms of reliability. Moreover, the reliability advantage

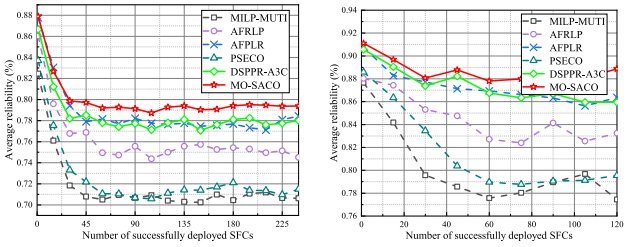


(a) Request delay in USANET. (b) Request delay in NSFNET. (c) VNF delay in USANET. (d) Link delay in USANET.

Fig. 7. SFC delay.

Table 4
Summary of experimental results.

Algorithm	USANET						NSFNET					
	Obj	Acceptance	Latency	Reliability	Cost	Variance	Obj	Acceptance	Latency	Reliability	Cost	Variance
MILP-MUTI	\	73.5%	72.98	70.63%	155.57	0.115	\	73.1%	38.74	77.45%	96.36	0.169
AFRLP	0.235	76.3%	64.08	74.51%	132.59	\	0.225	76.8%	33.86	83.24%	86.01	\
AFPLR	0.221	79.0%	63.98	78.46%	141.02	\	0.215	77.4%	34.54	86.37%	85.48	\
PSECO	0.276	63.8%	68.15	71.49%	134.15	0.101	0.262	72.5%	35.66	79.55%	87.75	0.146
DSPPR-A3C	0.202	76.8%	58.48	78.02%	133.25	0.108	0.193	78.0%	33.18	85.96%	81.98	0.156
MO-SACO	0.140	81.5%	55.89	79.39%	124.38	0.085	0.146	82.5%	31.58	88.87%	75.79	0.124



(a) Average reliability in USANET. (b) Average reliability in NSFNET.

Fig. 8. Reliability in different topologies.

shown in Fig. 8(a) is greater than that in 8(b), which proves that MO-SACO performs better in large-scale networks. This is because during the virtual link mapping process, adjacent VNFs pass through more than one physical link after being deployed, and the offloading decision based on the access network highlights this feature.

Fig. 9(a) and (b) illustrates the average SFC deployment cost of the MILP-MUTI, AFRLP, AFPLR, PSECO, DSPPR-A3C and MO-SACO methods. In networks of different sizes, the advantages of MO-SACO's average deployment cost compared to the other five algorithms are still obvious, but there is a difference in the cost of MO-SACO over DSPPR-A3C. As the network space expands, the larger the exploration space of MO-SACO, the more significant the cost optimization of nodes and links, while DSPPR-A3C has to back up more VNFs in a complex data center, resulting in a slight increase in instantiation costs and node resources.

In order to get into the details of deployment cost reduction, let us analyze Fig. 9(c) and (d), which show node costs (including VNF instantiation) and link costs respectively. In Fig. 9(c), we can clearly observe the difference in node resource costs. DSPPR-A3C has a higher VNF cost than other methods due to the backup mechanism. The other five strategies are distributed in two areas in the curve. The cost reduction of MO-SACO, AFPLR and AFRLP compared to MILP and PSECO depends on the aggregation decision of VNF. In Fig. 9(d), these methods show different results in terms of bandwidth overhead. AFRLP and DSPPR-A3C are committed to reducing the number of links to affect bandwidth costs. AFRLP prioritizes link reliability and favors paths with fewer hops, while DSPPR-A3C directly optimizes specific VNF links, so DSPPR-A3C is better than AFRLP. PSECO deploys as many VNFs as possible in the fog network. Although it takes advantage of the low bandwidth cost, the ping-pong path will extend the SFC path. When the MO-SACO deploys each VNF in the SFC to the physical node, it always find the optimal location based on the decisions in slowromancapiv@-A and slowromancapiv@-B, and tends to deploy multiple VNFs under the same access network.

After completing the above simulation, we also found that MO-SACO is also effective in load balancing. We utilize

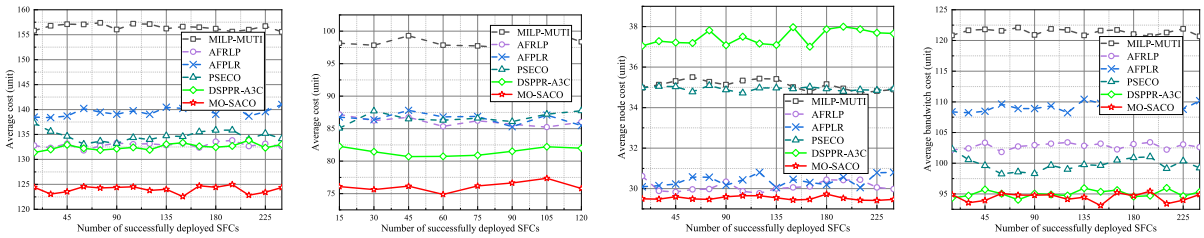
$$Var = \frac{1}{N_p}\sum_{i=1}^{N_p}(c_{n_i}^{use} - \frac{\sum_{j=1}^{N_p} c_{n_j}^{use}}{N_p})^2 \tag{26}$$

to calculate the load variance of cloud nodes in the network, where N_p means the set of cloud node. The statistics of the results are shown in Fig. 10(a) and (b). Whether in large-scale data centers or small networks with limited resources, the node load variance MO-SACO is numerically lower than other comparison algorithms. In the definition of queueing delay Eq. (17), the processing rate of a node is negatively correlated with its load, so the SFC tends to be deployed on servers with relatively sufficient resources in the available network set. In addition, our computation offloading decision jointly invokes the resources of cloud, fog, and edge devices, further relieving the load of the cloud core network and reserves more space for the cloud network to accommodate SFCs.

In order to demonstrate the superiority of MO-SACO more clearly, we synchronize the above experimental results in Table 4. The results presented show that the MO-SACO strategy is optimal among the six algorithms, compared with PSECO, it alleviates the ping-pong path problem of the access network and further emphasizes the availability of fog/edge resources, thus significantly increasing the service acceptance rate; aggregation decision-making overcomes the shortcomings of DSPPR-A3C's backup mechanism in resource doubling, making it to save cost significantly while maintaining near-lossless reliability. In addition, the synergy of the two optimization methods proposed in this paper also keeps the delay at a low level. Hence, MO-SACO can be used as an efficient heuristic algorithm for SFC deployment.

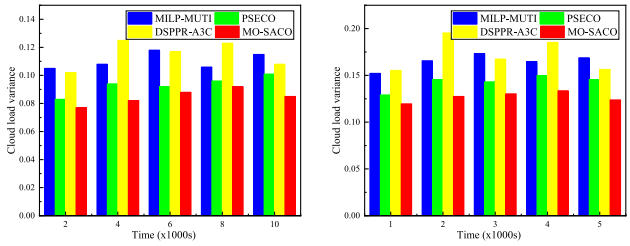
7. Conclusion

In this paper, we study the multi-objective optimization problem of SFC deployment. We formulated the problem as a MILP model that further expands the placement constraints of physical networks and developed a MO-SACO algorithm for the deployment problem of the



(a) Average cost in USANET. (b) Average cost in NSFNET. (c) Node and VNF instantiation cost in USANET. (d) Link resource cost in USANET.

Fig. 9. Costs in different topologies, including node and bandwidth cost.



(a) Server load variance in USANET. (b) Server load variance in NSFNET.

Fig. 10. Physical node load variance.

SFC, which focuses on the various stages of an SFC request. Specifically, first, before deploying, we design a VNF aggregation strategy to reduce its resource consumption without affecting its overall function; then, when deploying, we jointly invoke the resources of cloud, fog and edge devices by computing the offloading decision to minimize latency of SFC paths. Finally, after each request is deployed, we monitor their lifetime and performance index, and deal with non-compliant SFCs in a timely manner. In the simulation results, MO-SACO not only excels in latency and reliability, but also achieves comparable performance in the request acceptance rate, deployment cost, and load balancing.

According to the survey, there are various scenarios that cause SFC interruption during its life cycle, such as traffic overload or node downtime caused by an increase in the number of requests, VNF failure, user migration, etc. In future, we will focus on the SFC migration problem caused by the above scenarios. Specifically, we will adopt machine learning methods, e.g., using improved graph convolution networks to extract features of physical networks, and selecting other neural network models to generate SFC migration decisions and exploring the most advanced deep reinforcement learning strategies to ensure the optimality of the strategy.

CRediT authorship contribution statement

Junbi Xiao: Writing – review & editing, Methodology, Conceptualization. **Jiaqi Zheng:** Writing – review & editing, Conceptualization. **Wu Wen:** Conceptualization. **Mohsen Guizani:** Conceptualization. **Peiying Zhang:** Methodology, Funding acquisition. **Lizhuang Tan:** Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is partially supported by the Natural Science Foundation of Shandong Province under Grant ZR2023LZH017, ZR2022LZH015 and ZR2023QF025, partially supported by the Project for Integrated Innovation of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023PX057, partially supported by the Open Project of Key Laboratory of Computing Power Network and Information Security, Ministry of Education under Grant 2023ZD010, partially supported by the Talent Project of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023RCKY141, partially supported by the Industry- university Research Innovation Foundation of Ministry of Education of China under Grant 2021FNA01001, partially supported by the Major Scientific and Technological Projects of CNPC under Grant ZD2019-183-006.

References

[1] J.F. Cevallos Moreno, R. Sattler, R.P. Caulier Cisterna, L. Ricciardi Celsi, A. Sánchez Rodríguez, M. Mecella, Online service function chain deployment for live-streaming in virtualized content delivery networks: A deep reinforcement learning approach, Future Internet 13 (11) (2021) [Online]. Available: https://www.mdpi.com/1999-5903/13/11/278.

[2] D. Qi, S. Shen, G. Wang, Towards an efficient VNF placement in network function virtualization, Comput. Commun. 138 (2019) 81–89, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366418308247.

[3] W. Miao, G. Min, Y. Wu, H. Huang, Z. Zhao, H. Wang, C. Luo, Stochastic performance analysis of network function virtualization in future internet, IEEE J. Sel. Areas Commun. 37 (3) (2019) 613–626.

[4] G. Sun, Z. Chen, H. Yu, X. Du, M. Guizani, Online parallelized service function chain orchestration in data center networks, IEEE Access 7 (2019) 100147–100161.

[5] J.M. Halpern, C. Pignataro, Service Function Chaining (SFC) Architecture, RFC 7665, 2015, [Online]. Available: https://www.rfc-editor.org/info/rfc7665.

[6] P. Zhang, X. Pang, Y. Bi, H. Yao, H. Pan, N. Kumar, DSCD: Delay sensitive cross-domain virtual network embedding algorithm, IEEE Trans. Netw. Sci. Eng. 7 (4) (2020) 2913–2925.

[7] P. Cong, G. Xu, T. Wei, K. Li, A survey of profit optimization techniques for cloud providers, ACM Comput. Surv. 53 (2) (2020) [Online]. Available: https://doi.org/10.1145/3376917.

[8] H. Hantouti, N. Benamar, T. Taleb, Service function chaining in 5G & beyond networks: Challenges and open research issues, IEEE Netw. 34 (4) (2020) 320–327.

[9] C. Mouradian, D. Naboulsi, S. Yangui, R.H. Glitho, M.J. Morrow, P.A. Polakos, A comprehensive survey on fog computing: State-of-the-Art and research challenges, IEEE Commun. Surv. Tutor. 20 (1) (2018) 416–464.

[10] S. Shen, L. Huang, H. Zhou, S. Yu, E. Fan, Q. Cao, Multistage signaling game-based optimal detection strategies for suppressing malware diffusion in fog-cloud-based IoT networks, IEEE Internet Things J. 5 (2) (2018) 1043–1054.

[11] Y. Shen, S. Shen, Z. Wu, H. Zhou, S. Yu, Signaling game-based availability assessment for edge computing-assisted IoT systems with malware dissemination, J. Inf. Secur. Appl. 66 (2022) 103140, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212622000308.

[12] Z. Dongcheng, X. Jingzhao, S. Gang, Research on deployment of service function chains for delay-sensitive services, J. Univ. Electron. Sci. Technol. China 50 (6) (2021) 852–860.

[13] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (5) (2016) 2795–2808.

[14] H. Jin, X. Zhu, C. Zhao, Computation offloading optimization based on probabilistic SFC for mobile online gaming in heterogeneous network, IEEE Access 7 (2019) 52168–52180.

[15] J. Wu, H. Dai, Y. Wang, S. Shen, C.-Z. Xu, PECCO: A profit and cost-oriented computation offloading scheme in edge-cloud environment with improved moth-flame optimisation, Concurr. Comput.: Pract. Exper. 34 (2022).
[16] A. Zamani, S. Sharifian, A novel approach for service function chain (SFC) mapping with multiple SFC instances in a fog-to-cloud computing system, in: 2018 4th Iranian Conference on Signal Processing and Intelligent Systems, ICSPIS, 2018, pp. 48–52.
[17] D. Zhao, L. Luo, H. Yu, V. Chang, R. Buyya, G. Sun, Security-SLA-guaranteed service function chain deployment in cloud-fog computing networks, Cluster Comput. 24 (3) (2021) 2479–2494, [Online]. Available: https://doi.org/10.1007/s10586-021-03278-4.
[18] H. Zhang, Y. Yang, X. Huang, C. Fang, P. Zhang, Ultra-low latency multi-task offloading in mobile edge computing, IEEE Access 9 (2021) 32569–32581.
[19] P. Zhang, H. Yao, Y. Liu, Virtual network embedding based on computing, network, and storage resource constraints, IEEE Internet Things J. 5 (5) (2018) 3298–3304.
[20] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, R. Boutaba, Delay-aware VNF placement and chaining based on a flexible resource allocation approach, in: 2017 13th International Conference on Network and Service Management, CNSM, 2017, pp. 1–7.
[21] T. Subramanya, D. Harutyunyan, R. Riggio, Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks, Comput. Netw. 166 (2020) 106980, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619310254.
[22] Q. Zhang, F. Liu, C. Zeng, Adaptive interference-aware VNF placement for service-customized 5G network slices, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 2449–2457.
[23] Z. Huang, W. Zhong, D. Li, H. Lu, Delay constrained SFC orchestration for edge intelligence-enabled IIoT: A DRL approach, J. Netw. Syst. Manage. 31 (3) (2023) [Online]. Available: https://doi.org/10.1007/s10922-023-09743-2.
[24] R. Kang, F. He, T. Sato, E. Oki, Virtual network function allocation to maximize continuous available time of service function chains with availability schedule, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 1556–1570.
[25] D. Zhai, X. Meng, Z. Yu, X. Han, Reliability-aware service function chain backup protection method, IEEE Access 9 (2021) 14660–14676.
[26] S. Herker, X. An, W. Kiess, S. Beker, A. Kirstaedter, Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements, in: 2015 IEEE Globecom Workshops (GC Wkshps), 2015, pp. 1–7.
[27] Y. Zeng, Z. Qu, S. Guo, B. Tang, B. Ye, J. Li, J. Zhang, RuleDRL: Reliability-aware SFC provisioning with bounded approximations in dynamic environments, IEEE Trans. Serv. Comput. 16 (5) (2023) 3651–3664.
[28] T. Lun, Z. Pei-Pei, Z. Guo-Fan, C. Qian-Bin, Dynamic deployment algorithm for service function chaining with QoS guarantee, J. Beijing Univ. Posts Telecom 16 (4) (2018) 90–96.
[29] C. Yajun, L. Hua, R. Hongwei, X. Tong, R. Wang, Service function chain deployment method for user QoS and network resource awareness, Small Microcomput. Syst. 42 (9) (2021) 1931–1937.
[30] M.T. Beck, J.F. Botero, Scalable and coordinated allocation of service function chains, Comput. Commun. 102 (2017) 78–88, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366416303577.
[31] H. Chen, X. Wang, Y. Zhao, T. Song, Y. Wang, S. Xu, L. Li, MOSC: a method to assign the outsourcing of service function chain across multiple clouds, Comput. Netw. 133 (2018) 166–182, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912861830029X.
[32] Z. Luo, C. Wu, Z. Li, W. Zhou, Scaling geo-distributed network function chains: A prediction and learning framework, IEEE J. Sel. Areas Commun. 37 (8) (2019) 1838–1850.
[33] A. Zamani, B. Bakhshi, S. Sharifian, An efficient load balancing approach for service function chain mapping, Comput. Electr. Eng. 90 (2021) 106890, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790620307436.
[34] H. Xu, G. Fan, L. Sun, W. Li, G. Kuang, B. Fan, G. Ahmadi, Dynamic SFC placement scheme with parallelized SFCs and reuse of initialized VNFs: An A3C-based DRL approach, J. King Saud Univ. - Comput. Inf. Sci. 35 (6) (2023) 101577, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157823001313.
[35] C. Han, S. Xu, S. Guo, X. Qiu, A. Xiong, P. Yu, K. Guo, D. Guo, A multi-objective service function chain mapping mechanism for IoT networks, in: 2019 15th International Wireless Communications & Mobile Computing Conference, IWCMC, 2019, pp. 72–77.
[36] X. Shang, Z. Liu, Y. Yang, Online service function chain placement for cost-effectiveness and network congestion control, IEEE Trans. Comput. 71 (1) (2022) 27–39.
[37] P. Zhang, C. Wang, G.S. Aujla, N. Kumar, M. Guizani, Iov scenario: Implementation of a bandwidth aware algorithm in wireless network communication mode, IEEE Trans. Veh. Technol. 69 (12) (2020) 15774–15785.