Computer Science and Information Systems 19(2):803–827 https://doi.org/10.2298/CSIS210920006D

Optimized Placement of Symmetrical Service Function Chain in Network Function Virtualization

Nhat-Minh Dang-Quang[1] and Myungsik Yoo[2]

[1] Department of Information Communication Convergence Technology, Soongsil University
Seoul 06978, Korea
minhdqn@soongsil.ac.kr
[2] School of Electronic Engineering, Soongsil University
Seoul 06978, Korea
myoo@ssu.ac.kr

Abstract. Network function virtualization (NFV) is one of the key technology enablers for actualizing 5G networks. With NFV, virtual network functions (VNFs) are linked together as a service function chain (SFC), which provides network functionality for the customer on demand. However, how to efficiently find a suitable placement for VNFs regarding the given objectives is an extremely difficult issue. The existing approaches assume that the SFC has a simple and asymmetrical pattern that is unsuitable to modeling a real system. We address this limitation by studying a VNF placement optimization problem with symmetrical SFCs that can support both symmetric and asymmetric traffic flows. This NP-hard problem is formulated as a mixed-integer linear programming (MILP) model. An iterative greedy-based heuristic is proposed to overcome the complexity of the MILP model. Extensive simulation results show that the proposed heuristic can obtain a near-optimal solution compared to MILP for a small-scale network, and at the same time, is superior to a traditional heuristic for a large-scale network.

Keywords: Network function virtualization, multi-objective, VNF placement optimization, symmetric, heuristic.

1. Introduction

Network function virtualization (NFV) [1] has emerged as a new network paradigm that can overcome the limitations of traditional networks. NFV is a key emerging technology in multi-access edge computing (MEC) realizing the Internet-of-Things (IoT) and 5G networks [2]. Virtual customer premises equipment (vCPE) [3], which is the most popular use case of NFV, can provide a flexible platform where multiple network services (e.g., a firewall, router, dynamic host configuration protocol (DHCP), network address translation (NAT), and load balancing) are virtualized as virtual network functions (VNFs) and run on a common hardware platform. Through vCPE, service providers can rapidly develop new services and avoid all manual processes.

A network service usually consists of various VNFs based on the customer requirements. An ordered sequence of VNFs is formed through a service function chain (SFC) [4,5]. Some challenges having a significant impact on NFV include the efficient deployment of an SFC while ensuring that the service level agreements are satisfied and wisely allocating network resources. This is known as the VNF placement (VNF-P) problem, which is the focus of this study. As mentioned earlier, an ordered sequence of VNFs is formed as a service function chain (SFC). Given a set of requested SFCs, the goal of VNF-P is to place the VNFs on suitable physical nodes in the network with regard to the given objectives while satisfying constraints related to the nodes, edge capacities, and latency bound [8,9,13,14,15,19,20]. A good placement solution may considerably enhance network resource usage efficiency and lower CAPital EXpenditures and OPerating EXpenses (CAPEX/OPEX), resulting in increasing profitability for cloud service providers. Given different service requests by different users, the VNF placement challenge concerns how to deploy a sequence of SFCs, each of which contains numerous VNFs, into cloud available resources. Several open-source NFV platforms, including OpenStack Tacker[3], Open Source MANO (OSM)[4], Open Platform for NFV (OPNFV) [5], and Open Network Automation Platform (ONAP)[6], have also concentrated on this problem.

In practice, SFC may be asymmetric or symmetric, depending on the requirement of the service providers. A symmetrical SFC can process a given two-way flow, i.e., a request flow (e.g., from a client to a server of the network service) and a response flow (e.g., from a server of the network service to a client)[4]. The existing studies on traffic symmetry for SFC are limited [6]. A hybrid SFC has attributes of both symmetric and asymmetric SFCs; that is, some VNFs require symmetric traffic, whereas other VNFs do not require response traffic or are independent of the corresponding request traffic [5]. However, conventional approaches assume that network services have a relatively simple and asymmetric paradigm, which is unsuitable to the modeling of real systems. Common VNFs, such as a deep packet inspection (DPI) or firewall are required to process symmetric traffic flows because of the consistent state of the flow [4,7]. In addition, to reduce wasted resources and minimize the number of deployed VNFs, the VNF instances can be reused across several SFCs in the network [19,20].

Based on the above observations, we propose a VNF-P model for SFCs that can support both symmetric and asymmetric traffic flows. This model is formulated as a mixed-integer linear programming (MILP) model with an objective function that simultaneously minimizes the number of deployed VNF instances, the total required data rates, and the total latency. In the model, the VNF instances can also be shared across several SFCs to reduce the number of deployed VNF instances. A heuristic based on the iterative greedy algorithm is presented to solve the problem in large-scale networks owing to the complexity of MILP. The simulation results show that the proposed heuristic can obtain a near-optimal solution compared to the MILP for a small-scale network and is also superior to its counterpart for a large-scale network.

The remainder of this paper is organized as follows. Section 2 overviews previous related studies. Section 3 describes the problem formulation and its model. Section 4 presents an iterative greedy-based heuristic method for solving the problem. Section 5 shows the simulation settings and numerical results. Finally, section 6 provides some concluding remarks.

[3] https://docs.openstack.org/tacker
[4] https://osm.etsi.org/
[5] https://www.opnfv.org/
[6] https://www.onap.org/

2. Related work

Since the concept of NFV was introduced in 2012 [11], the VNF-P began to draw attention as the building block of SFCs. SFC placement usually consists of two-step process: first, the resource allocation problem; second, the traffic steering problem. The VNF-P problems have been widely studied in recent years. In the majority of survey works, the VNF-P problems have been formulated as an integer programming (ILP, MIP or MILP) model. Then, heuristic placement algorithms have been proposed [12]. Furthermore, we see that the goal of SFC placement is the objective function of the optimization problem. Here are frequently used goals:

– Quality of Service (QoS) parameters: QoS parameters include energy consumption, service latency, availability, etc. These parameters can help the service provider to know the quality of a network service which provided to users.
– Cost and Revenue: The network cost represents the deploying cost or operating cost of VNF on the nodes. Meanwhile, the revenue is the net value earned by serving traffic needs, optimized under capacity constraints.

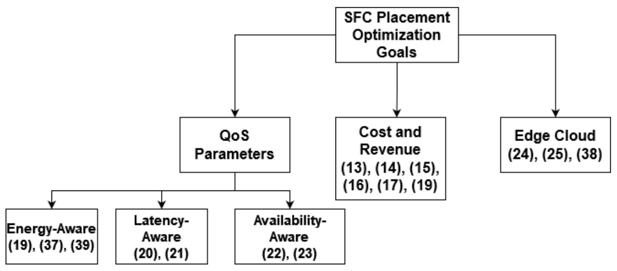

SFC Placement Optimization Goals
QoS Parameters
Cost and Revenue (13), (14), (15), (16), (17), (19)
Edge Cloud (24), (25), (38)
Energy-Aware (19), (37), (39)
Latency-Aware (20), (21)
Availability-Aware (22), (23)

Fig. 1. Related works to VNF-P

Figure 1 shows three common optimization goals used for VNF placement optimization problem such as QoS parameters, cost and revenue and edge cloud. In this section, we categorize the related studies based on these goals.

– For the QoS parameters: if the deployment VNFs are not done properly in a compute resource-sharing environment like cloud computing, QoS will have a substantial impact on overall cloud service performance. A QoS-aware VNF placement strategy would significantly minimize traffic transmission across the whole data center, and therefore congestion and data transfer time. There are three common parameters in QoS-aware:
  - Energy-aware: It aims to minimize the power consumptions, which is achieved through policies from Service level agreements (SLAs).
  - Latency-aware: It aims to minimize network latency, VNF migration delay, etc.
  - Availability-aware: It aims to maximize the availability of VNFs.
– For the Cost and Revenue: This is the basic and fundamental resource allocation problem in NFV [10]. The main purpose of this problem is to minimize the total network cost, traffic volume in network.

– For the edge cloud: The concept of Mobile Edge Computing (MEC) brings the computing resource closer to end-users. Applying NFV to MEC will help reducing the service time latency for end-users as well as helping service providers to get more benefit from lower expenditures and higher efficiency [10].

Firstly, QoS can significantly impact the overall performance of cloud services if the VNF-P is not studied well. We categorized it into three common sub-category optimization goals, including Energy-Aware, Latency-Aware, and Availability-Aware.

For the energy-ware, in [19], the authors proposed a Monte Carlo Tree Search (MCTS) based method that shares VNFs among the tenants to minimize the energy consumption of the servers in the VNF-P model. Abdelaal et al. [37] proposed a novel approach for VNF placement called VNFRP (Virtual network functions and their replica placement). They formulated the VNF placement problem as an integer linear programming problem to optimize energy consumption and SFC placement cost. Furthermore, they proposed a heuristic-based algorithm to solve the proposed problem. The simulation showed when the number of replicas is increased, VNFRP may dramatically enhance load balancing by up to 80%. Zeng et al. [39] proposed a VNF placement and routing technique to optimize network delay and energy consumption. The technique combines a classic genetic algorithm with a simplex approach with strong local search capabilities, avoiding the problem that traditional genetic algorithms are prone to falling into the local optimum solution. The results showed that the suggested technique is capable of reducing network latency efficiency while also reducing network energy usage.

For the latency-aware, the authors [21] formulated the VNF placement problem as an ILP model and proposed a hidden Markov Chain-based heuristic for placing the VNFs to optimize the cost and delay. Agarwal et al. [22] formulated the VNF placement and CPU allocation as a convex optimization problem. Then, they proposed a method, which was based on the MaxZ placement heuristic, to optimize VNF placement and CPU allocation decisions.

For the availability-aware, Zhao and Dán [23] formulated the VNF placement as an ILP. Then they split it into a master problem and a sub-problem. They assumed that there are U failure scenarios, and each failure scenario i has a probability p_i. The master problem and sub-problem have been solved iteratively by using Generalized Benders Decomposition (GBD). In each iteration, they produced an upper bound and lower bound for the objective value of the original problem. The iteration process stops when these bound values meet the termination condition. The study [24] solved the end-to-end delay and service chaining availability for VNF placement using an ILP and heuristic solution.

Secondly, we surveyed the literature about cost-aware resource allocation in VNF-P. This is the basic and fundamental resource allocation problem in NFV [10]. The authors [13] found the number of essential VNFs and allocate them to minimize the total network cost and the resource fragmentation. The authors [13] presented an integer linear programming (ILP) model and a dynamic programming-based heuristic for the VNF-P problem. In [14], the VNF-P model was proposed using mixed-integer linear programming (MILP), which considers standard and fast path VNF forwarding methods with two different optimization goals, including traffic engineering and NFVI cost minimization. In [15], the target of the VNF-P model was to minimize the overall traffic volume in the network, whereas some VNFs can change the traversing traffic volume, e.g., a WAN optimizer can compress the traffic before sending it to the next hop, resulting in a change in traffic volume. Pham et al. [16] proposed an algorithm based on the sampling Markov approximation for optimizing the operation and network traffic cost. Tomasasilli et al. [17] proposed two logarithmic factor approximation algorithms to optimize the deployment cost. The first algorithm was based on LP rounding, while the second algorithm was based on greedy algorithm. The authors in [20] analyzed the resource consumption on the servers and links. Their model allows different SFCs to share a single VNF if these SFCs demand the same VNF.

Thirdly, we also reviewed some works related to VNF-P in Edge cloud environment. Cziva et al. [25] formulated the VNF placement problem to minimize the total latency of all users to their VNFs. The authors [7] also applied the Optimal Stopping Theory to detect when to re-evaluate the optimal problem using two parameters: migration cost and path delay. Song et al. [26] put the study on the VNF placement for 5G edge computing using users' mobility. First, [26] proposed a user grouping model based on geographic information of user context and then defined (and calculated the optimal number of) clusters to minimize the delay of network services from one end to the other. Next, a graph partitioning algorithm that assigns VNFs to clusters was presented to minimize the movement between the user and clusters while optimizing the loss of users' data rate due to VNF migration. Tao et al. [38] formulated VNF placement problem to optimize both latency and energy consumption at the edge. A cost-minimizing optimization strategy is used to ensure the latency and energy consumption parameters. To overcome this challenge, they created a graph of edge systems and users. The placement of the VNFs is determined by the edge systems based on cost and user requirements. The findings demonstrate that the proposed VPE technique minimizes the cost of edge systems while maintaining the quality of the mobile user experience.

For a broader scope of resource allocation in NFV and its details, refer to previous comprehensive surveys [2,8,9,10].

The existing studies assumed that the SFC is asymmetric, which only considers a unidirectional traffic flow. Some Service Functions (SFs) need bidirectional flow, which means they required both forward and backward directions. For example, common VNFs such as deep packet inspection (DPI) or firewall are required to process symmetric traffic flows because of the consistent state of the flow. Thus, this is the limitation of the existing studies on VNF placement optimization problem due to it is not suitable to model real system. Thus, the existing studies on traffic symmetry are still limited.

There are few existing works that study symmetry SFC. Bifulco et al. [18] works on scalability and traffic steering for legacy mobile networks. Their proposal mentioned symmetry SFC, where the upstream and downstream traffics are the forward traffic and the backward traffic, respectively. The symmetry was achieved by using Network Address Translation (NAT). Their study took into account the overall symmetry of the chain, regardless of whether it is symmetric or asymmetric. Hantouti and Benamar [6] discuses benefits of using partially SFCs and introduce a new method for calculating the reverse route of symmetric SFCs. The result of the proposed method showed that it could help lower the state of forwarding and, as a result, the traffic delay. Therefore, the studies [6] and [18] do not focus on VNF placement for symmetric SFC. Thus, they are not presented in Figure 1.

By addressing mentioned limitations, this work studies the VNF placement optimization problem that can support both symmetric (unidirectional) and asymmetric (bidirectional) traffic flows, which is suitable for modeling real systems. Our contribution is to formulate symmetric-enabled VNF placement optimization problem in order to minimize the number of deployed VNF instances, data required rate of SFCs, and total latency. Considering symmetry traffic for SFCs also helps save the number of deployed VNF instances because the VNF instances can be shared across several service functions. Thus, it is necessary to study on the VNF placement optimization for symmetrical SFC.

3. Symmetric-enabled VNF Placement Problem

In this section, the system model and problem formulation are presented. Table 1 shows the notations used in this paper.

3.1. System Model

NFV-enabled Network A directed graph is used to describe an NFV-enabled network, namely, $G = (N, E)$, where N denotes the sets of nodes and E denotes the sets of edges. We represent the CPU and memory capacities of each network node $n \in N$ as n_{cpu} and n_{mem}, respectively. A data rate capacity associated with each network edge $e \in E$ is represented as e_{dr}. Each network edge e has a latency e_{lat}.

Service function chain (SFC) A service function chain is specified by $S = (I_S, P_S)$. Here, $i \in I_S$ denotes a VNF instance of an SFC. A VNF instance can be implemented in a virtual machine (VM) or container running over the network infrastructure. In such a case, a specific amount of resources, such as the CPU and memory, are required. Hence, i_{cpu}, i_{mem} represent the CPU and memory resource consumption for a VNF instance i, respectively. The CPU and memory consumption for the VNFs can have a uniform distribution or other specific distribution depending on the service providers. The directed path between two instances defined in an SFC is denoted by $p \in P_S$, which connects exactly a head instance p_{head} of p to a tail instance p_{tail} of p. In addition, p_{Mlat} denotes the maximum latency that can be tolerated for path p.

In this study, an SFC can process both asymmetric and symmetric traffic flows, which require different types of VNF instances.

Type of VNF instances The four types of VNF instances considered are as follows.

(i) A source instance i_{src} represents a client. Therefore, it is fixed at a specific location and does not consume any CPU or memory resources. The source instance is given an outgoing data rate f_S^{dr} of a flow $f_S \in F_n$, where F_n is a union of all flows of SFCs at node n and $F_n \subset F$, where F is a union of all flows.
(ii) A symmetrical instance i_{sym} can process the symmetric traffic flows, i.e., the request (rq.) flow from a client to a server and response (rsp.) flow from a server to a client. The existing studies on the VNF-P problem assume that the SFCs only have one type of VNF instance, which processes only the asymmetric traffic flows. However, a VNF instance may be symmetrical or asymmetrical depending on the network service requirements. Many common VNFs such as a deep packet inspection (DPI), firewall, and L4-L7 load balancer often require a symmetric traffic flow processing feature to ensure that the flow state is consistent[4,7].

Table 1. Notations Used in the Paper

Notation	Description
Parameters	
$G = (N, E)$	A directed graph including a set of nodes N and a set of edges E.
$n_{cpu}, n_{mem} > 0$	CPU and memory capacities of node $n \in N$, respectively.
$e_{lat}, e_{dr} > 0$	Latency and data rate capacity of edge $e \in E$, respectively.
$S = (I_S, P_S)$	SFC including a set of instances $i \in I_S$ and a set of paths $p \in P_S$.
p_{head}, p_{tail}	Head and tail instances of path p.
$p_{Mlat} > 0$	Maximum latency of p that can be tolerated.
$i_{cpu}, i_{mem} > 0$	CPU and memory requirements of an instance.
i_{src}, i_{sym}, i_{asym}, i_{dst}	Role of instances, including source, symmetrical, asymmetrical, and destination instances.
r_i^{rq}, r_i^{rsp}	Request and response data rate scaling of an instance i.
F_n	A collection of all flows corresponding to all SFCs at node n.
F	A union of all flows, i.e., $F_n \subset F$.
$f_S \in F_n$	A flow from a source instance of an SFC S at node n. Each f_S has a data rate value of f_S^{dr}.
$0 \leq w_1, w_2, w_3 \leq 1, w_1+w_2+w_3 = 1$	Weighting factors of objectives.
Auxiliary variables	
$ingress_{i,n}^{f_S,rq}$, $ingress_{i,n}^{f_S,rsp} \geq 0$	Incoming data rate of request and response flows f_S of instance i placed at node n.
$egress_{i,n}^{f_S,rq}$, $egress_{i,n}^{f_S,rsp} \geq 0$	Outgoing data rate of request and response flows f_S of instance i placed at node n.
r_i^{rq}	data rate scaling of instance i for request flow
r_i^{rsp}	data rate scaling of instance i for response flow
Decision variables	
$y_{i,n} \in \{0, 1\}$	1 if an instance i is placed at node n.
$z_{n_1,n_2}^{p,e} \in \{0, 1\}$	1 if an edge e is used for sending traffic of path p that has p_{head} placed at n_1 and p_{tail} placed at n_2.
$dr_{n_1,n_2}^{p,e} \geq 0$	Required data rate at an edge e using the path p for sending traffic when the path p has p_{head} placed at n_1 and p_{tail} placed at n_2.

(iii) An asymmetrical instance i_{asym} only processes either a request or a response flow. Therefore, it has only incoming/outgoing data rates of either a request or response flow.

(iv) A destination instance i_{dst} redirects the traffic flow from an incoming request to an outgoing response. Therefore, it has the incoming data rates of the request flow and outgoing data rates of the response flow. In this paper, the destination can be the server of a network service or content caching server, which can be flexibly deployed in the network.

Each VNF instance has the incoming request/response flows and the outgoing request/response flows based on the type. We define four data rate values of a VNF instance i placed at node n of a flow f_S, i.e., incoming request data rate $ingress_{i,n}^{f_S,rq}$, outgoing request data rate $egress_{i,n}^{f_S,rq}$, incoming response data rate $ingress_{i,n}^{f_S,rsp}$, and outgoing response data rate $egress_{i,n}^{f_S,rsp}$. The required data rate at an edge of a flow may be changed when the flow passes a VNF instance [15]. For example, the WAN optimizer VNF compresses the traffic before sending it to the next hop, resulting in a traffic savings of up to 80% [27], the video optimizer VNF can decrease the data rate by up to 50% owing to a video trans-rating procedure [28], or the content filtering VNF can reduce the required data rate by blocking the video streaming during working hours [29]. Therefore, we define the data rate scaling of instance i for the request and response flows, i.e., $r_i^{rq} = \frac{egress_{i,n}^{f_S,rq}}{ingress_{i,n}^{f_S,rq}}$ and $r_i^{rsp} = \frac{egress_{i,n}^{f_S,rsp}}{ingress_{i,n}^{f_S,rsp}}$, respectively. Figure 2 shows four types of VNF instances with their flows and data rates. Figure 3 illustrates a sample of a symmetric-enabled SFC.

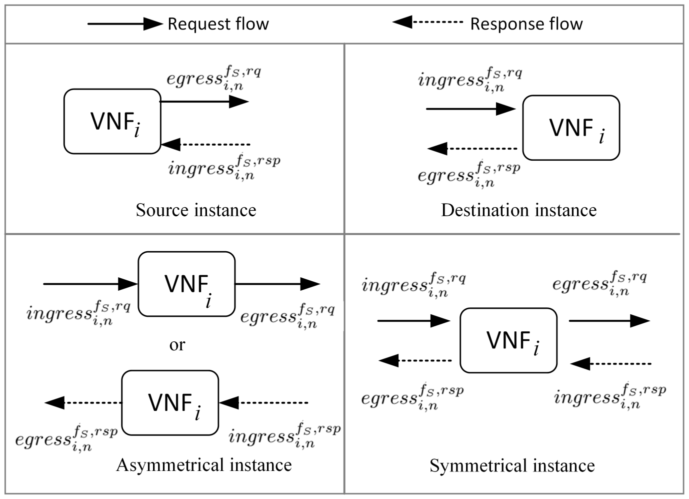
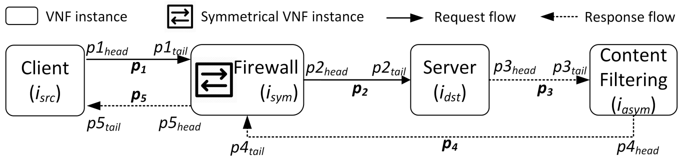


Fig. 2. Four types of VNF instances with their flows and data rates



VNF instance
Symmetrical VNF instance
Request flow
Response flow
Client (i_{src})
Firewall (i_{sym})
Server (i_{dst})
Content Filtering (i_{asym})
$p1_{head}$ $p1_{tail}$ p_1
$p2_{head}$ $p2_{tail}$ p_2
$p3_{head}$ $p3_{tail}$ p_3
$p4_{head}$ $p4_{tail}$ p_4
$p5_{head}$ $p5_{tail}$ p_5

Fig. 3. A sample of symmetric-enabled SFC

Shared VNF The model also adapts the concept of the sharing and reuse of VNFs [19,20], which allows the different SFCs to use the same VNF instances with the same identifier in their flows. The VNF reuse strategy can improve the resource utilization of the servers and reduce the number of VNF instances deployed. A VNF instance can have the request and/or response flows for every SFC using that instance. This means there are sets of incoming and outgoing data rates corresponding to each SFC at a shared instance. All instances except for the source instance can be shared.

Figure 4 shows an example of symmetrical SFCs placed in the Abilene network and sharing an instance in which the source instance of SFC 1 is fixed at node 7, and the source instance of SFC 2 is fixed at node 2. The sample SFC consists of a source instance (Client - CLT), a firewall instance (FW), a server instance (SVR), and a content filtering instance (CF).

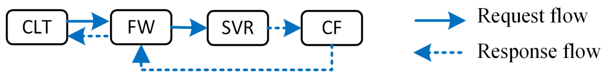
3.2. Problem Formulation

The objective of the VNF-P problem is to find the optimal location of VNF instances of symmetric-enabled SFCs in the network such that the number of VNF instances required, the data rate of the SFCs required, and the total latency caused by the SFCs are simultaneously minimized while satisfying the constraints related to the node and edge capacities, as well as the latency bounds for each SFC.

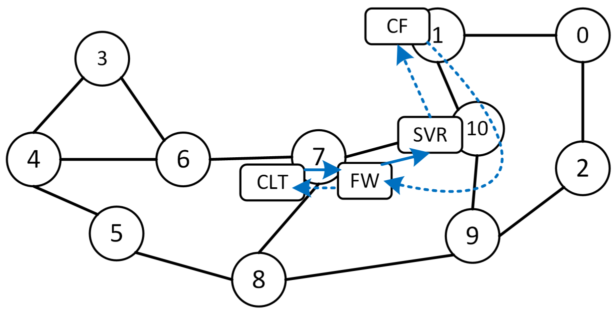
Variable Declaration The VNF-P problem is formulated as a mixed-integer linear programming (MILP) [30] model. The decision variables of the model are detailed as follows.

(i) The binary variable $y_{i,n}$ represents a placement of the instance i at node n.
(ii) The binary variable $z^{p,e}_{n_1,n_2}$ represents whether an edge e is used for sending traffic of path p, which has a head instance p_{head} placed at node n_1 and a tail instance p_{tail} placed at node n_2.
(iii) The continuous variable $dr^{p,e}_{n_1,n_2}$ represents a required data rate of path p at an edge e if that edge is used for sending traffic of path p, which has a head instance p_{head} placed at node n_1 and a tail instance p_{tail} placed at node n_2.

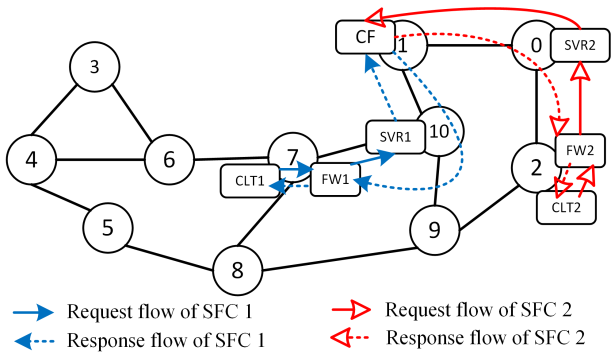
In addition, the continuous variables $ingress^{f_S,rq}_{i,n}$, $ingress^{f_S,rsp}_{i,n}$, $egress^{f_S,rq}_{i,n}$, and $egress^{f_S,rsp}_{i,n}$, which are auxiliary variables, are data rate values of an incoming/outgoing request/response corresponding to flows f_S of an instance i placed at node n, respectively.



A sample of symmetric-enabled SFC.



A sample SFC placed in the network.



Two sample SFCs sharing an instance.

Fig. 4. An example of symmetric-enabled SFCs placed in the network

Objective Function and Constraints The optimization objective is to simultaneously minimize the number of VNF instances placed across the network (obj_1), the total required data rate (obj_2), and the total latency (obj_3). The problem can be formulated following the MILP model.

$$\begin{aligned}&\text{minimize}\\ &w_1 \cdot obj_1(\mathbf{x}) + w_2 \cdot obj_2(\mathbf{x}) + w_3 \cdot obj_3(\mathbf{x})\\ &\text{subject to } (1) - (15)\end{aligned}$$

where

(i) $\mathbf{x} = (y_{i,n}, dr^{p,e}_{n_1,n_2}, z^{p,e}_{n_1,n_2})$, which is a decision vector in a feasible set. The feasible set is defined by the following constraints.
(ii) $obj_1(\mathbf{x}) = \sum_{i \in I_S, n \in N} y_{i,n}$. This function defines the total number of VNF instances deployed in the network.
(iii) $obj_2(\mathbf{x}) = \sum_{p \in P_S, n_1, n_2 \in N, e \in E} dr^{p,e}_{n_1,n_2}$. This function defines the total data rate required by all SFCs in the network.
(iv) $obj_3(\mathbf{x}) = \sum_{p \in P_S, n_1, n_2 \in N, e \in E} (z^{p,e}_{n_1,n_2} \cdot e_{lat})$. This function defines the total latency of all SFCs in the network.
(v) w_1, w_2, and w_3 are the weighting factors in which $0 \leq w_1, w_2, w_3 \leq 1$, and $w_1 + w_2 + w_3 = 1$, and are used to assign the importance of the functions based on the specific requirement of the service providers.

The following constraints are considered in the model. Constraints (1) and (2) are the mapping consistency rules for the source instance. In (1), every source instance is placed in a predefined node. The outgoing data rate of every source instance equals the predetermined data rate of the flow exiting from that instance, as shown in (2).

$$\forall S, \forall i \in I_S, \text{if } i \text{ is } i_{src}, \exists! n \in N : y_{i,n} = 1 \tag{1}$$

$$\begin{aligned}&\forall S, \forall i \in I_S, \forall n \in N, \text{if } y_{i,n} = 1, \text{if } i \text{ is } i_{src},\\ &\exists! f_S \in F_n : egress^{f_S,rq}_{i,n} = f^{dr}_S\end{aligned} \tag{2}$$

Constraints (3), (4), and (5) express the data rate rules of an instance i when placed at node n. In (3), if an instance is i_{sym} or i_{asym}, for a request flow, the outgoing data rate of that instance equals the r^{rq}_i scaling rate of the incoming data of that instance. In (4), if an instance is i_{sym} or i_{asym}, for a response flow, the outgoing data rate of that instance equals the r^{rsp}_i scaling rate of the incoming data of that instance. In (5), if an instance is i_{dst}, the outgoing data rate of the response flow of that instance equals the r^{rq}_i scaling rate of the incoming data of the request flow of that instance.

$$\begin{aligned}&\forall S, \forall i \in I_S, \forall n \in N, \forall f_S \in F, \text{if } y_{i,n} = 1,\\ &\text{if } i \text{ is } i_{sym} \text{ or } i_{asym} : egress^{f_S,rq}_{i,n} = r^{rq}_i \cdot ingress^{f_S,rq}_{i,n}\end{aligned} \tag{3}$$

$$\begin{aligned}&\forall S, \forall i \in I_S, \forall n \in N, \forall f_S \in F, \text{if } y_{i,n} = 1,\\ &\text{if } i \text{ is } i_{sym} \text{ or } i_{asym} : egress^{f_S,rsp}_{i,n} = r^{rsp}_i \cdot ingress^{f_S,rsp}_{i,n}\end{aligned} \tag{4}$$

$$\begin{aligned} &\forall S, \forall i \in I_S, \forall n \in N, \forall f_S \in F, \text{if } y_{i,n} = 1, \\ &\text{if } i \text{ is } i_{dst} : egress_{i,n}^{f_S,rsp} = r_i^{rq} \cdot ingress_{i,n}^{f_S,rq} \end{aligned} \tag{5}$$

Constraints (6) and (7) show the data rate rules of the head and tail instances of path p. The incoming data rate of tail instance p_{tail} of path p equals the outgoing data rate of a head instance p_{head} of that path correlated with the request and response flow f_S.

$$\begin{aligned} &\forall S, \forall p \in P_S, \forall n_1, n_2 \in N, \forall f_S \in F, \\ &\text{if } y_{p_{head},n_1} = y_{p_{tail},n_2} = 1 : egress_{p_{head},n_1}^{f_S,rq} \\ &= ingress_{p_{tail},n_2}^{f_S,rq} \end{aligned} \tag{6}$$

$$\begin{aligned} &\forall S, \forall p \in P_S, \forall n_1, n_2 \in N, \forall f_S \in F, \\ &\text{if } y_{p_{head},n_1} = y_{p_{tail},n_2} = 1 : egress_{p_{head},n_1}^{f_S,rsp} \\ &= ingress_{p_{tail},n_2}^{f_S,rsp} \end{aligned} \tag{7}$$

Constraint (8) expresses the flow conservation in which the flow must leave an egress of an instance if the flow passes through it. The required data rate at every edge along a path p equals the outgoing data rate of a head instance of that path over all flows.

$$\begin{aligned} &\forall S, \forall p \in P_S, \forall n, n_1, n_2 \in N, \text{if } y_{p_{head},n_1} = y_{p_{tail},n_2} = 1 : \\ &\sum_{nn' \in E} dr_{n_1,n_2}^{p,nn'} - \sum_{n'n \in E} dr_{n_1,n_2}^{p,n'n} = \\ &\begin{cases} \sum_{f_S} egress_{p_{head},n_1}^{f_S,rq} & \text{if } n = n_1 \neq n_2, f_S \text{ is request.} \\ \sum_{f_S} egress_{p_{head},n_1}^{f_S,rsp} & \text{if } n = n_1 \neq n_2, f_S \text{ is response.} \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{8}$$

Constraint (9) ensures the consistency of the variables $z_{n_1,n_2}^{p,e}$ and $dr_{n_1,n_2}^{p,e}$. If a path p uses an edge e for sending traffic, a required data rate of that path exists at that edge. Otherwise, it does not.

$$\begin{aligned} &\forall S, \forall p \in P_S, \forall n_1, n_2 \in N, \forall e \in E : \\ &z_{n_1,n_2}^{p,e} = \begin{cases} 1 & \text{if } dr_{n_1,n_2}^{p,e} > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{9}$$

Constraint (10) prevents a loop in the path. It states that a path p should only use one direction of an edge if that edge is used for sending traffic.

$$\begin{aligned} &\forall S, \forall p \in P_S, \forall n_1, n_2 \in N, \forall nn' \in E, \\ &\text{if } n'n \in E : z_{n_1,n_2}^{p,n'n} + z_{n_1,n_2}^{p,nn'} \leq 1 \end{aligned} \tag{10}$$

Constraint (11) guarantees that the total latency of the edges used by a path p cannot surpass the maximum latency of that path.

$$\forall S, \forall p \in P_S, \forall n_1, n_2 \in N : \sum_{e \in E} (z_{n_1,n_2}^{p,e} \cdot e_{lat}) \leq p_{Mlat} \tag{11}$$

Constraint (12) ensures that a request flow and the corresponding response flow must go through the same symmetrical instance i_{sym}.

$$\begin{aligned}
&\forall S, \forall i \in I_S, \forall n \in N, \forall f_S \in F, \text{if } i \text{ is } i_{sym}: \\
&\text{if } ingress_{i,n}^{f_S,rq} + egress_{i,n}^{f_S,rq} > 0: \\
&ingress_{i,n}^{f_S,rsp} + egress_{i,n}^{f_S,rsp} > 0 \\
&\text{if } ingress_{i,n}^{f_S,rq} + egress_{i,n}^{f_S,rq} = 0: \\
&ingress_{i,n}^{f_S,rsp} + egress_{i,n}^{f_S,rsp} = 0
\end{aligned} \tag{12}$$

Constraints (13), (14), and (15) are capacity constraints. In (13), the required data rates at an edge cannot exceed the data rate capacity of that edge. The node resource capacity constraints are shown in (14) and (15).

$$\forall S, \forall e \in E: \sum_{p \in P_S, n_1, n_2 \in N} dr_{n_1,n_2}^{p,e} \leq e_{dr} \tag{13}$$

$$\forall S, \forall n \in N: \sum_{i \in I_S} (i_{cpu} \cdot y_{i,n}) \leq n_{cpu} \tag{14}$$

$$\forall S, \forall n \in N: \sum_{i \in I_S} (i_{mem} \cdot y_{i,n}) \leq n_{mem} \tag{15}$$

Model Complexity With the help of optimization solvers (e.g., Gurobi [32], CPLEX [33]), the optimal solution can be achieved using a combinatorial search (e.g., Branch-and-Bound algorithm [34]). However, the VNF-P problem has been proven to be NP-hard [8,13]. This means that the time complexity increases exponentially with the network size. Therefore, it becomes a challenge to obtain an optimal solution when a network is large.

4. Proposed Heuristic-based Placement Method

Because the VNF placement problem is an NP-hard problem, which means that there is no algorithm that will always efficiently produce the exactly correct answer on all inputs. To address the NP-hard complexity of the problem, we propose a heuristic to solve it. The benefits of the proposed Heuristic-based Placement method is to find a feasible (not optimal) solution that is good enough to quickly solve and achieve optimization placement goals. The heuristic includes three steps. First, the edges in the network are weighted based on the latency and data rate capacity. Second, an initial solution is given by a greedy algorithm based on the calculated weight of the edges. Finally, an iterative greedy algorithm improves the initial solution using a random placement process.

4.1. Heuristic-based placement process

Every edge e is assigned a weight $e_w = \frac{1}{e_{dr}} + e_{lat}$ and all pairs with the shortest (least-weight) paths in the network are found using the Floyd-Warshall algorithm [31] concurrently in which an edge with a higher data rate capacity and lower latency has a lower weight compared to the other edges.

A greedy algorithm aims to select a node for placing a new VNF instance if that node has sufficient capacity available and has the least-weight path from the current node of the VNF instances. The greedy algorithm is illustrated in Algorithm 1. The instances are processed following the sequence of the flow in both directions in the SFC. The source instance is placed to a predefined node. Each instance i has a path p that connects the prior instance to it. The algorithm finds prospective nodes for deploying the instance i corresponding to p. A node is a prospective node if it satisfies the node, edge capacity, and latency constraints. To support shared instances, a node can be a prospective node if it has deployed instances that are the same type as the considered instance i, and satisfies the edge capacity and latency constraints. If there is no prospective node, the algorithm returns an infeasible solution. Otherwise, the algorithm creates or reuses (if exists) the instance i at a prospective node that has the least-weight path from the location of the prior instance. It should be noted that if the request flow passes an instance i_{sym}, the corresponding response flow must return exactly to that instance.

Algorithm 1 Greedy algorithm pseudocode

```
Input: Weighted $G = (N, E)$; $\forall S$; $i_{rand} = null$.
Output: Placement Solution $Sol$.
 1: for all $S$ do
 2:     Place $i_{src}$ at a predefined location.
 3:     for all other $i$ in $I_S$ in both directions of flow $f_S$ do
 4:         Get path $p$ coming to instance $i$ in flow $f_S$.
 5:         if $f_S$ is in response direction & $i$ is $i_{sym}$ then
 6:             Map $p$ to least-weight path connected node of $i_{sym}$.
 7:         else
 8:             Find prospective nodes that satisfy node, edge capacities, and latency constraints.
 9:             if perspectives nodes do not exist then
10:                 Return infeasible solution.
11:             end if
12:             Select a prospective node that has the least-weight path from the location of the prior
                instance.
13:             Create or reuse (if exists) instance $i$ on the selected node and map $p$ to the correspond-
                ing least-weight path.
14:             Update node and edges capacities.
15:         end if
16:     end for
17: end for
18: return Solution $Sol$
```

The solution of the greedy algorithm may be local optimal because the greedy algorithm simply chooses the minimum-weight path with the corresponding end node to place a new VNF instance. An iterative greedy algorithm is proposed to avoid the local optimality of the greedy algorithm, as shown in Algorithm 2. Given the initial solution by the greedy algorithm and predefined maximum number of iterations, the iterative greedy algorithm arbitrarily chooses an instance in the current solution and places it to a different location. For every iteration, the algorithm creates a new solution, which has a new objective value calculated using the objective function. The algorithm then compares two objective values of the two solutions, chooses the solution with the least objective value, and assigns it as the best solution. This process is repeated with the current best solution until the maximum number of iterations is reached. The algorithm will return the solution with the lowest objective value.

Algorithm 2 Iterative greedy algorithm pseudocode

```
Input: Weighted $G = (N, E)$; $\forall S$; $Sol$; $n\_iter$.
Output: Best solution $Sol_{best}$
1: $Sol_{best} \leftarrow Sol$
2: $iter \leftarrow 0$
3: while $iter < n\_iter$ do
4:     $iter \leftarrow iter + 1$
5:     Select a random instance $i_{rand}$ from $Sol_{best}$ except the source instances.
6:     $Sol_{new} \leftarrow$ run greedy algorithm but assign $i_{rand}$ at a different node.
7:     $obj_{best} \leftarrow$ calculate the objective value of $Sol_{best}$
8:     $obj_{new} \leftarrow$ calculate the objective value of $Sol_{new}$
9:     if $obj_{new} < obj_{best}$ then
10:        $Sol_{best} \leftarrow Sol_{new}$
11:    end if
12: end while
13: return $Sol_{best}$
```

4.2. Complexity Analysis

The time complexity of the placement process is a combination of the phases, including finding all pairs of shortest paths using the Floyd-Warshall algorithm and repeating the greedy algorithm in the number of iterations. The Floyd–Warshall algorithm takes $O(N^3)$ [31]. An efficient implementation of the greedy algorithm is used to compute the placement of an instance, taking $O(N \log N)$. For an SFC, it takes $O(IN \log N)$, where I is the maximum number of instances needed to be considered in both flow directions. Therefore, for K number of SFCs, the greedy algorithm takes $O(KIN \log N)$. By contrast, the iterative greedy algorithm repeats the greedy algorithm in the number of iterations M. Therefore, it takes $O(MKIN \log N)$. Hence, the overall running time of the heuristic-based placement process is $O(N^3 + MKIN \log N)$.

5. Performance Evaluation

In this section, extensive simulations conducted to verify the proposed model and algorithms are described.

5.1. Simulation Settings

The algorithms under evaluation are the proposed MILP (denoted as **MILP**), the proposed heuristic algorithm using 20 iterations (as described in section Conclusion, and denoted as **proposed heuristic**), and the first-fit heuristic algorithm (a baseline algorithm, denoted as **firstfit heuristic**), which takes each instance in turn and places it into the first node that can accommodate it. Gurobi optimizer 8.1 [32] using a Branch-and-Bound algorithm [34] is used to solve the MILP model. Python 3 programming language was used to develop the simulation and algorithms. All computations were conducted on a PC supplied with an Intel Xeon CPU E3-1230 V2 @ 3.30 GHz, 16 GB RAM, running Windows 10 x64 OS.

The SFC used for the evaluation is a content-filtering service, as illustrated in Fig. 5. It consists of a (Client - CLT), a firewall instance (FW), a server instance (SVR), and a content filtering instance (CF). First, CLT sends requests to FW. The FW is responsible for analyzing inbound and outbound network traffic and chooses whether to allow or prohibit certain types of traffic based on a set of predefined rules. It also supports symmetry traffic flows, which means it always receives and responds to requests from CLT. If the request is valid, the FW forwards the request to SVR to process. After processing the request of CLT, the SVR sends the response to CF. The CF blocks content that contains harmful information, such as pornographic content, etc. Finally, it sends back the response to FW to return to CLT. We assume that the CPU and memory consumption of each VNF instance obey a uniform distribution of (2,4). The data rate scaling equals 1 for all instances except for the CF instance, which has a data rate scaling r_i^{rsp} of 0.5. The maximum latency that can be tolerated for each path p_{Mlat} is 20 ms. Every SFC has a required outgoing data rate of $f_S^{dr} = 1$ (Gbps) from its source instance for all simulations.

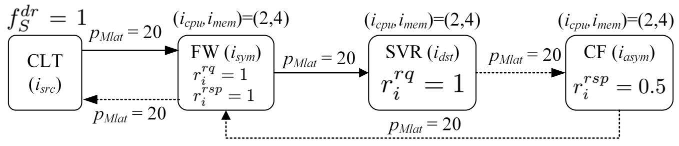


Fig. 5. Symmetric-enabled content filtering SFC used for the evaluation

The simulations were conducted on the Abilene network, which is an ISP backbone network, and the Geant network, which is a research backbone network. Both networks were taken from the Internet topology zoo dataset [35]. In these networks, the edge latency e_{lat} in milliseconds is estimated as the propagation latency calculated from the geographical distances between nodes [36]. Table 2 shows the details of the networks used and their setups.

The inputs of the experiment are the network, SFC and number of SFCs need to be deployed. We evaluate the ouput of proposed heuristic-based placement method with other methods by four objectives: Number of deployed VNFs instances, Total required data rate measured in giga bytes per second (Gbps), Total latency measured in second (s) and Computational time measured in second (s).

Table 3 shows the four settings of the weighting factors. Using the balance setting of the weighting factor, we first look at the effect of the symmetric feature of the SFCs in the proposed model as compared to a conventional model. Then, the optimal results achieved using the MILP of the proposed model with the four different settings of the weighting factors are examined. The number of SFCs increases from 1 to 5 owing to the

Table 2. Network topologies used for simulations

Network	$\|N\|$	$\|E\|$	e_{dr}	n_{cpu}	n_{mem}
			(Gbps)	(Unit)	(GB)
`Abilene`	11	28	10	4	8
`Geant`	40	122	20	4	8

Table 3. Weighting factor settings

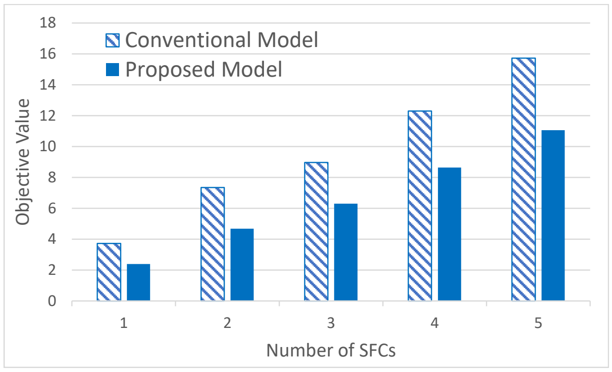
Scenarios	w_1	w_2	w_3
`MILP-Balance`	1/3	1/3	1/3
`MILP-VNF`	1	0	0
`MILP-Dr`	0	1	0

exponential computational time of the MILP. Using a balanced setting of the weighting factors, the MILP is then compared with the proposed heuristic and the first-fit heuristic. The proposed heuristic is then compared with the first-fit heuristic in the large-scale Geant network with the number of SFCs varying from 1 to 20. Every SFC has a different source instance location from the others.

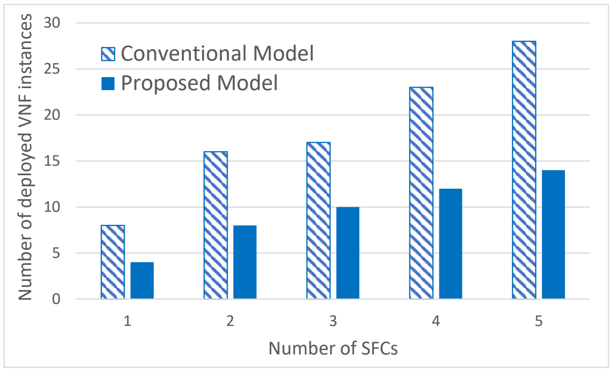
5.2. Simulation Results

Conventional model versus proposed model One disadvantage of the conventional model, which does not support the symmetric feature of the VNF instances, is that it cannot ensure a consistent flow state. By contrast, if the model does not support symmetrical VNF instances, the given SFC should be separated into sub-SFCs, i.e., (i) $CLT \rightarrow FW \rightarrow SVR$, (ii) $SVR \rightarrow CF \rightarrow FW$, and (iii) $FW \rightarrow CLT$. The additional instances need to be deployed to process these SFCs instead of the one-time process of the proposed model. This can lead to an increase in the number of deployed VNF instances required, which increases the node's resource consumption and server energy consumption. The simulation was conducted on the Abilene network. The results of the conventional and proposed models differ significantly only in the number of VNF instances deployed. The latency and data rate results are the same because the two models achieve the same optimal placement of the VNF instances. As shown in Fig. 6, the gap between the objective value and the number of VNF instances required between the two models increases with the number of SFCs.

Effect of the weighting factors Figure 7 illustrates the effect of different weighting factors on the MILP performance. MILP-VNF maintains the lowest number of deployed VNF instances by sharing all deployed VNF instances except for the source instances. However, it must sacrifice the data rate and latency objectives because the flow must go further to reach the shared instances. By contrast, MILP-Dr and MILP-Lat deploy numerous instances within the node capacities to reduce the data rate and latency, respectively. MILP-Balance tries to minimize all three objectives simultaneously because these three objectives have equal weighting factors. The results show that MILP-Balance can archive the optimal required data rate and total latency of MILP-Dr and MILP-Lat, where the number of deployed VNF instances is not overly high compared to MILP-VNF.

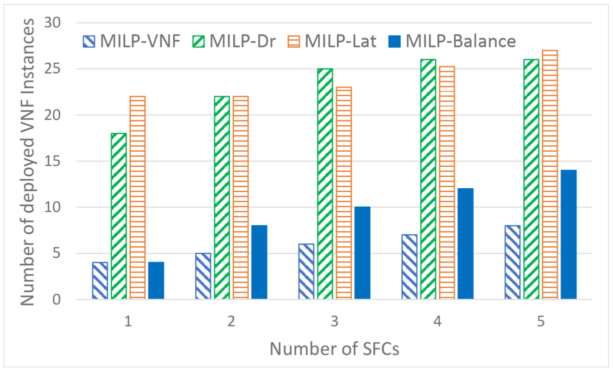


Objective value

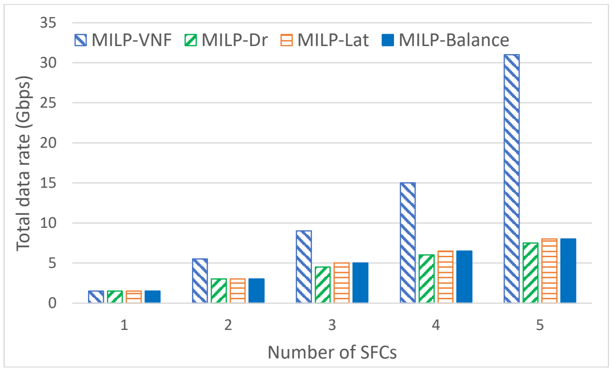


Number of deployed VNF instances

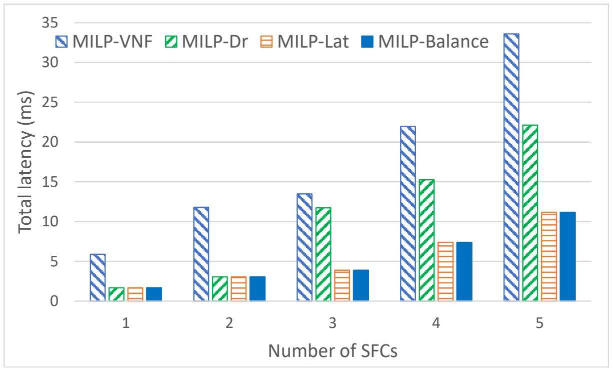
Fig. 6. Comparison of conventional and proposed models



Number of deployed VNFs instances



Total required data rate (Gbps)



Total latency (ms)

Fig. 7. The effect of different weighting factors on MILP performance in the Abilene network

Results of MILP and heuristics in the Abilene network From Fig. 8, it can be seen that the proposed heuristic outperforms the first-fit heuristic and achieves a near-optimal objective value compared to MILP with an average optimality gap of 9.7%. The details of each objective are shown in Fig. 9. The first-fit heuristic has the lowest number of deployed VNF instances among the different algorithms because it chooses the first node that can accommodate the VNF instance and reuse that instance in other SFCs. However, the first-fit heuristic must sacrifice the latency and data rates required because the distance between VNF instances is greater than that in the solution to the other algorithms. It also shows that the proposed heuristic can simultaneously minimize all objectives and effectively avoid being trapped in the local optimality compared to the first-fit heuristic. In terms of the computational time, as a simple algorithm, the first-fit heuristic outperforms the other algorithms. However, the proposed heuristic with 20 repetitions only takes approximately 1 s to solve the problem with five SFCs, whereas the MILP takes 10.5 h.

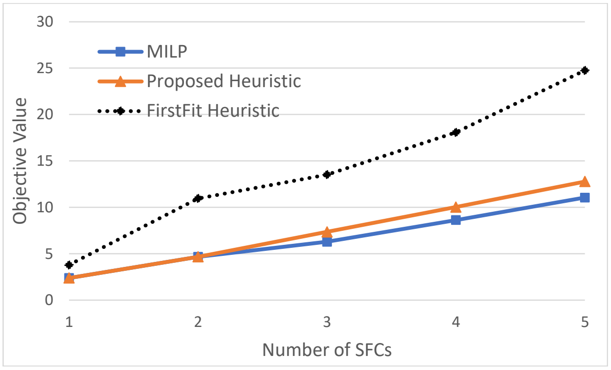
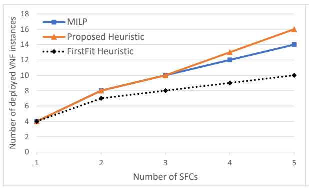
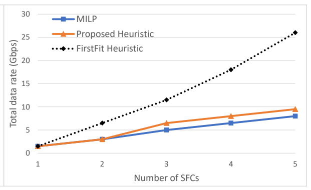


Fig. 8. Objective value of the algorithms in Abilene network

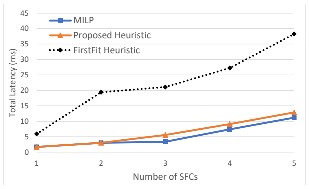
Results of heuristics in the Geant network With the increase in the size of the network and number of SFCs, the MILP cannot obtain the solution within an acceptable time. Therefore, we only compare the performance of the proposed heuristic and first-fit heuristic in the Geant network. The objective value given by the proposed heuristic is again always lower than the objective value given by the first-fit heuristic, as shown in Fig. 10. The gap between two objective values of the algorithms under evaluation increases when increasing the number of SFCs. Figure 11 shows the details of all objectives. The first-fit heuristic uses the same shared VNF instances and therefore keeps the number of VNF instances required as low as possible. However, the latency and data rate required by the first-fit heuristic are too high compared to that of the proposed heuristic. It should be noted that the first-fit heuristic cannot produce a solution because it violates the data rate capacity constraint when the number of SFCs is 20. The proposed heuristic undoubtedly has a higher computational time compared to the first-fit heuristic. However, it only takes under a minute (i.e., 42 s) when placing 20 SFCs at one time. However, with the quality of the solution determined by the proposed heuristic, this can be acceptable when



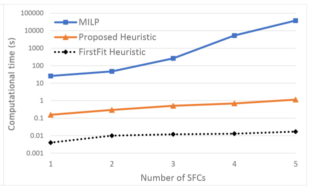
Number of deployed VNFs instances



Total required data rate (Gbps)



Total latency (ms)



Computational time (s)

Fig. 9. Performance comparison in Abilene network

running on a large network. The computational time of this heuristic can be reduced if we choose a suitable number of iterations and implement it using high-performance servers.

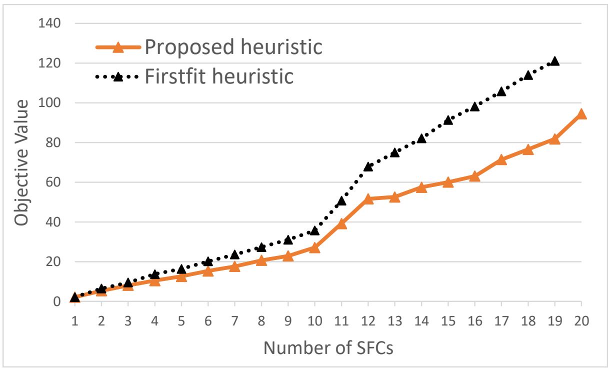
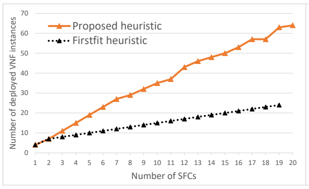


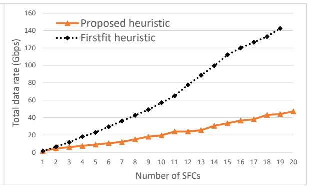
Fig. 10. Objective value of the iterative greedy algorithm and the greedy algorithm in Geant network

6. Conclusion

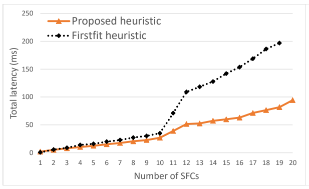
Conventional approaches assume that network services have a relatively simple and asymmetric paradigm, which is unsuitable to the modeling of real systems. Common VNFs, such as a deep packet inspection (DPI) or firewall are required to process symmetric traffic flows because of the consistent state of the flow [4,7]. The impact of this study is that



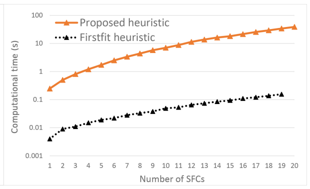
Number of deployed VNFs instances



Total required data rate (Gbps)



Total latency (ms)



Computational time (s)

Fig. 11. Performance comparison in Geant network

it focused on the VNF placement problem with reusable VNF instances for symmetric-enabled SFC, which can process both asymmetric and symmetric traffic flows. The symmetric features of SFC can not only ensure the consistency of the flow state but also reduce the number of deployed instances. This NP-hard problem was formulated using a multi-objective MILP model, which minimizes number of VNF instances, data rate of SFCs, and total latency. Owing to the complexity of the MILP model, an iterative greedy-based heuristic was proposed to solve the problem in large-scale networks. The benefits of the proposed Heuristic-based Placement method is to find a feasible (not optimal) solution that is good enough to quickly solve and achieve optimization placement goals. The extensive simulation results showed that the proposed heuristic can gain the near-optimal solution (under a 10% optimality gap) within a shorter time period compared to the MILP approach for a small-sized network. The performance of the proposed heuristic was also superior to the baseline first-fit heuristic for a large-sized network.

For future works, we plan to formulate the optimization problem for more real-world scenarios. We will conduct more experiments on other common networks used in VNF placement optimization problem such as AAR, JGN2plus, etc., which can be found in [35]. Also, due to the development of artificial intelligence, especially with Deep Reinforcement Learning (DRL) in solving VNF-P problem, the authors will apply this technique to solve the optimization problems.

Acknowledgments. This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support programs (IITP-2021-0-02046) and (IITP-2021-2017-0-01633) supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation)

References

1. ETSI NFV, “Network function virtualisation: An introduction, benefits, enablers, challenges & call for action,” Introductory White Paper, Issue 1, *SDN and OpenFlow World Congress*, Darmstadt, Germany, Oct. 2012.
2. B. Yi, W. Xingwei, L. Keqin, and H. Min, “A comprehensive survey of network function virtualization,” *Comput. Netw.*, Vol. 133, pp. 212-262, 2018.
3. T.H. Nguyen, T. Nguyen and M. Yoo, “Analysis of deployment approaches for virtual customer premises equipment,” in *Proc. 32th IEEE Int. Conf. Inform. Netw. (ICOIN 2018)*, pp. 289-291, 2018.
4. P. Quinn and T. Nadeau, “Problem statement for service function chaining,” No. RFC 7498. 2015.
5. J. Halpern and C. Pignataro, “Service function chaining (sfc) architecture,” No. RFC 7665. 2015.
6. H. Hantouti and N. Benamar, ”Partially Symmetric Service Function Chains,” 2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), 2019, pp. 1-6, doi: 10.1109/MENACOMM46666.2019.8988534.
7. C. Zhang, S. Addepalli, N. Murthy, L. Fourie, M. Zarny, and L. Dunbar, “L4-L7 Service Function Chaining Solution Architecture,” Open Networking Foundation, ONF TS-027, 2015.
8. J.G. Herrera and J.F. Botero, “Resource allocation in NFV: A comprehensive survey.” *IEEE Trans. Netw. Service Manag.*, Vol. 13, No. 3, pp. 518-532, 2016.
9. A. Laghrissi, and T. Tarik, “A Survey on the Placement of Virtual Resources and Virtual Network Functions,” *IEEE Commun. Surveys Tuts.*, 2018.
10. S. Yang, F. Li, S. Trajanovski, R. Yahyapour and X. Fu, ”Recent Advances of Resource Allocation in Network Function Virtualization,” in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 2, pp. 295-314, 1 Feb. 2021, doi: 10.1109/TPDS.2020.3017001.
11. Network functions virtualisation: An introduction benefits enablers challenges & call for action, October 2012.
12. A. Mohamad and H. S. Hassanein, ”On Demonstrating the Gain of SFC Placement with VNF Sharing at the Edge,” 2019 IEEE Global Communications Conference (GLOBECOM), 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9014106.
13. F. Bari, R.C. Shihabur, A. Reaz, B. Raouf, and C.M.B.D Otto, “Orchestrating virtualized network functions,” *IEEE Trans. Netw. Service Manag.*, Vol. 13, No. 4, pp. 725-739, 2016.
14. M. Gao, B. Addis, M. Bouet and S. Secci, “Optimal orchestration of virtual network functions,” *Comput. Netw.*, Vol. 142, pp. 108-127, 2018.
15. W. Ma, S. Oscar, B. Jonathan, P. Deng, and P. Niki, “Traffic aware placement of interdependent nfv middleboxes.” in *Proc. IEEE Conf. Comput. Commun. (INFOCOM 2017)*, pp. 1-9, 2017.
16. C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, “Trafficaware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach,” IEEE Trans. Services Comput., vol. 13, no. 1, pp. 172–185, Jan./Feb. 2020.
17. A. Tomassillik, F. Giroire, N. Huin, and S. Perennes, “Provably efficient algorithms for placement of service function chains with ordering constraints,” in Proc. IEEE INFOCOM, 2018, pp. 774–782.
18. R. Bifulco, A. Matsiuk and A. Silvestro, ”CATENAE: A scalable service function chaining system for legacy mobile networks”, Int. J. Netw. Manag., vol. 27, no. 2, pp. 1-14, 2017
19. Soualah, Oussama, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. “Energy efficient algorithm for VNF placement and chaining.” in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing (CCGRID 2017)*, pp. 579-588, 2017.
20. T.W. Kuo, B.H. Liou, K.C.J Lin, and M.J. Tsai. “Deploying chains of virtual network functions: On the relation between link and server usage.” *IEEE/ACM Trans. Netw.*, Vol. 26, No. 4, pp. 1562-1576, 2018.

21. H. Chen et al., “MOSC: A method to assign the outsourcing of service function chain across multiple clouds,” Comput. Netw., vol. 133, pp. 166–182, 2018.
22. S. Agarwal, F. Malandrino, C. Chiasserini and S. De, ”Joint VNF Placement and CPU Allocation in 5G,” IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 1943-1951, doi: 10.1109/INFOCOM.2018.8485943.
23. P. Zhao and G. Dán, ”Resilient placement of virtual process control functions in mobile edge clouds,” 2017 IFIP Networking Conference (IFIP Networking) and Workshops, 2017, pp. 1-9, doi: 10.23919/IFIPNetworking.2017.8264849.
24. P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi and W. Kellerer, ”QoS-driven function placement reducing expenditures in NFV deployments,” 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1-7, doi: 10.1109/ICC.2017.7996513.
25. R. Cziva, C. Anagnostopoulos and D. P. Pezaros, ”Dynamic, Latency-Optimal vNF Placement at the Network Edge,” IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 693-701, doi: 10.1109/INFOCOM.2018.8486021.
26. S. Song, C. Lee, H. Cho, G. Lim and J. Chung, ”Clustered Virtualized Network Functions Resource Allocation based on Context-Aware Grouping in 5G Edge Networks,” in IEEE Transactions on Mobile Computing, vol. 19, no. 5, pp. 1072-1083, 1 May 2020, doi: 10.1109/TMC.2019.2907593.
27. Citrix, “Improve the XenApp and XenDesktop experience for branch and mobile workers with NetScaler SD-WAN.” [Online]. Available: https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/improve-the-xendesktop-experience-for-branch-and-mobile-workers-with-netscaler-sdwan.pdf
28. Tellabs, “Mobile Video Optimization Concept and Benefits,” White Paper, 2011. [Online]. Available: https://s3.amazonaws.com/zanran_storage/www.tellabs.com /ContentPages/2438991029.pdf
29. WebtTitan, “Internet Content Filtering Service.” [Online]. Available: https://www.webtitan.com/internet-content-filtering-service/
30. C.A. Floudas, “Nonlinear and mixed-integer optimization: fundamentals and applications,” Oxford University Press, 1995.
31. T.H. Cormen, C.E. Leiserson and R.L. Rivest, “The floyd-warshall algorithm,” Introduction to Algorithms, 558, p.565, 1990.
32. Gurobi Optimization. [Online]. Available: http://www.gurobi.com/
33. IBM ILOG CPLEX Optimization Studio. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer
34. J. Clausen, “Branch and bound algorithms-principles and examples,” Department of Computer Science, University of Copenhagen, p.1-30, 1999.
35. S. Knight, H.X. Nguyen, N. Falkner, R. Bowden and M. Roughan, “The internet topology zoo.” *IEEE J. Sel. Areas Commun.*, Vol. 29, No. 9, pp.1765-1775, 2011.
36. J. Kurose, and R. Keith, “Computer networks and the internet.” Computer networking: A Top-down approach. 7th ed. London: Pearson, 2016.
37. Abdelaal, Marwa A. and Ebrahim, Gamal A. and Anis, Wagdy R., ”Efficient Placement of Service Function Chains in Cloud Computing Environments” Electronics, 2021, doi: 10.3390/electronics10030323.
38. Tao, Xiaoyi and Ota, Kaoru and Dong, Mianxiong and Qi, Heng and Li, Keqiu, ”Cost as Performance: VNF Placement at the Edge” IEEE Networking Letters, 2021, doi: 10.1109/LNET.2021.3065651.
39. Zeng, Ying and Shi, Zhan and Wu, Zanhong, ”VNF Placement and Routing Algorithm for Energy Saving and QoS Guarantee” Proceedings of the 9th International Conference on Computer Engineering and Networks, 2021, doi: 10.1007/978-981-15-3753-089.

Nhat-Minh Dang-Quang received a Master's degree in Information and Communication Technology (ICT) from Soongsil University, Seoul, South Korea in February, 2022. He also received a B.Eng. degree in Software Engineering from the University of Information Technology, Vietnam National University—Ho Chi Minh City, Ho Chi Minh City, Vietnam, in 2019. His research interests include cloud computing, auto-scaling, and self-healing.

Myungsik Yoo received B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, South Korea, in 1989 and 1991, respectively, and a Ph.D. degree in electrical engineering from The State University of New York at Buffalo, New York, in 2000. He was a Senior Research Engineer with the Nokia Research Center, Burlington, MA. He is currently a Full Professor with the School of Electronic Engineering, Soongsil University, Seoul. His research interests include visible light communications, cloud computing, Internet protocols.

Received: September 20, 2021; Accepted: February 24, 2022.