Vietnam Journal of Science and Technology ***64***(1) (2026) 139–161
DOI: https://doi.org/10.15625/2525-2518/22956

Service function chain embedding in centralized and distributed data centers - A comparison

Ma Viet Duc[1], Nguyen Trung Kien[2], Dao Dai Hiep[1], Nguyen Tai Hung[1], Nguyen Huu Thanh[1,*]

[1]*School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, No. 1 Dai Co Viet street, Ha Noi, Viet Nam*
[2]*Chair of Communication Networks, University of Würzburg, Sanderring, No. 2, Würzburg, 97070, Bavaria, Germany*

[*]Email: *thanh.nguyenhuu@hust.edu.vn*

Received: 28 May 2025; Accepted for publication: 04 September 2025

Abstract. Cloud computing has played an important role in providing IoT-based services recently, such as healthcare, smart grid, driving-assistant systems and so forth. In such a paradigm, there is a tendency to deploy services in the edge-cloud environment, where data centers or computing clusters are partly moved to the edge of the network to avoid service degradation or disruption due to the scarcity of physical resources. This paper analyzes and discusses the advantages and disadvantages of providing virtualized services based on network function virtualization in two edge-cloud scenarios, in which data centers are in the center or placed at the edge of the network. Furthermore, a novel service function chain embedding strategy has been proposed, which considers centralized or multiple distributed DCs scenarios, and focuses on DC-internal embedding in fat-tree fabrics under online arrivals and resource fragmentation. Performance evaluation results show that the proposed strategy can improve the efficiency of the cloud system in terms of resource utilization and power consumption.

Keywords: distributed cloud, edge–cloud computing, network function virtualization.

Classification numbers: 4.4.1, 4.7.1, 4.7.2.

1. INTRODUCTION

Cloud computing deployment has been ever-increasing recently as it plays a crucial role in the digital economy. On the one hand, cloud computing enables change in the way digital businesses are handled by separating the *service providers* from *infrastructure providers* by allowing third parties to virtualize and offer their services dynamically and flexibly on top of the physical substrate, thus reducing the operation and investment expenditures. Moreover, cloud computing is one important part of the *IoT–Cloud–Big Data–AI* ecosystem as it is the infrastructure to accommodate data and services.

However, the conventional remote cloud paradigm has some drawbacks as it relies on the data center infrastructure residing at the core of the network. In the context of IoT applications, many services or data that the cloud system must facilitate come from users or IoT devices located at the edge. Thus, a series of difficulties arise as follows: (i) *latency* occurring due to

limited transmission bandwidth and processing bottleneck at the intermediate network nodes from the edge to the cloud; (ii) *scarcity of resources* as network and server resources should be allocated along the service paths, whose resource availability might be limited; (iii) *energy consumption*, because services are usually hosted on high-performance servers in the data centers operating around the clock, consumed energy takes up a large amount of the total system energy consumption [1, 2].

In that context, fog and edge computing, a concept first introduced by Cisco some years ago [3, 4] has emerged recently as a solution for the aforementioned problems. By putting processing and computing power on computing clusters or mini data centers near the edge, edge computing is about the concept of *cloud close to the ground* that can reduce the high-volume data traffic between the edge and the core of the network, thereby minimizing the disadvantages of cloud computing. Thus, the combination of edge and cloud is often implemented to utilize both edge and cloud advantages.

In the edge–cloud paradigm, *Network Function Virtualization (NFV)* enables network services to be deployed as *Service Function Chains (SFCs)*, where each chain consists of multiple *Virtual Network Functions (VNFs)* running on commodity servers. The NFV thus provides the flexibility to instantiate and manage services on demand across distributed infrastructures. The construction of an SFC, often called *service function chaining*, involves three main subproblems [5]:

1. *VNF Chain Composition (VNF-CC or VNF mapping)*: assigning VNFs to physical servers based on incoming SFC requests.

2. *VNF Forwarding Graph Embedding (VNF-FGE)*: mapping the logical interconnections between VNFs onto virtual switches and links.

3. *VNF Scheduling (VNFs-SCH)*: allocating execution slots for VNFs on the available machines.

In this work, we focus on the first two aspects, VNF-CC and VNF-FGE, which together form the SFC embedding problem. As illustrated in Figure 1, the SFC embedding represents a core challenge in NFV-enabled data centers: it determines how to efficiently place service chains onto the physical network substrate while meeting constraints such as resource availability, privacy, and energy consumption.

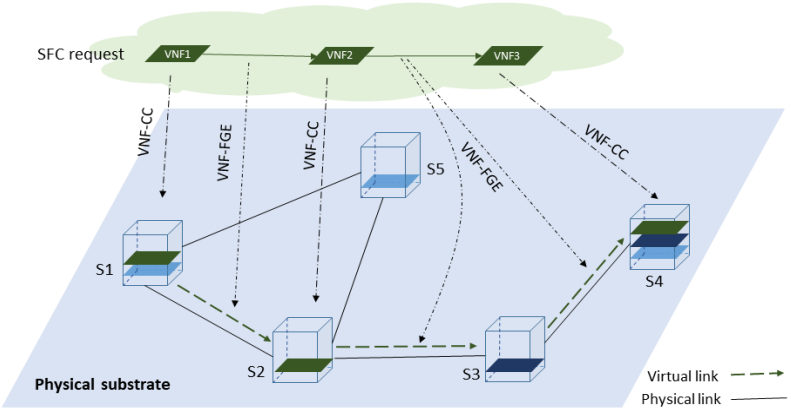


Figure 1. Service function chain embedding.

Solving the SFC embedding problem is NP-hard [6]. For that reason, current research mostly follows heuristic and meta-heuristic approaches.

In this research, we focus on energy and resource efficiency of service function chain embedding approaches in the edge-cloud environments with the following contributions:

- HRE-SFC: a unified, deployable heuristic algorithm that combines Pyramid Pointer Arrangement (PPA) for VNF ordering with energy-aware server consolidation and shortest-feasible link embedding, improving SFC acceptance rate.

- Under identical computing budgets and the same traffic topology, we isolate and compare (i) a *centralized cloud*, in which a big data center is placed at the core of the network; and (ii) a *distributed cloud*, in which several smaller data centers are moved to the edge of the network, yielding evidence-based recommendations for when to favor edge distribution versus core consolidation.

The rest of the paper is organized as follows. Section 2 discusses some related work on service function chaining. Section 3 formulates the SFC embedding problems and power profiling and modeling. In Section 4, we propose an energy and resource-aware SFC embedding mechanism. Section 5 shows some evaluation results. The last section concludes the work.

2. RELATED WORK

In this section, we first address some existing work on SFC embedding in general, followed by work focusing on SFC embedding in edge-cloud environments.

2.1. Service function chain embedding

SFC embedding - jointly mapping VNFs to compute nodes (VNF-CC) and routing virtual links (VNF-FGE) - is well-known NP-hard [6]. Early heuristics [7] clarified the problem space but assumed static or purely linear chains and therefore left three open issues that have dominated recent research:

- *Online embedding with fragmentation awareness.* Split-placement schemes such as the online heuristics in [8, 9] break a VNF across multiple hosts when contiguous resources are scarce, raising acceptance at the cost of higher orchestration complexity.

- *Cost/QoS–optimized exact models.* Approximation and ILP frameworks now blend placement, migration, and routing. Pham [10] maximizes long-term cost efficiency, while Erbati *et al.* [11] satisfy sub-millisecond latency for vehicular workloads.

- *Market- and energy-aware provisioning.* Wang *et al.* [12] introduce an auction-based CSAM mechanism for multi-cloud SFCs; Lin *et al.* [13] and Chintapalli *et al.* [14] curb power by shutting down idle devices or parallelizing SFC segments.

The studies of [8, 12, 15, 16] introduce heuristic algorithms for service function chain embedding to solve a joint VNF-CC and VNF-FGE problem. When performing VNF and graph embedding, if the corresponding resources on the substrate nodes or links are not sufficient due to the resource fragmentation, the VNF embedding algorithm splits the virtual network function n into two virtual network functions that can be embedded in different physical locations, thereby improving the resource availability and enhancing the embedding acceptance rate. However, splitting the original service chain into two or more paths may require complex

management operations in the virtualization platform to maintain the state of the service chain. Moreover, the VNF mapping algorithm in this work selects a server cluster that has more capacity than the others. In some cases, this requires that more immediate servers in the satisfied cluster be turned on instead of using existing active servers, which leads to more power consumption.

Deep-reinforcement-learning (DRL) remains the preferred data-driven alternative, with NFVDeep and its successors delivering near-instant, high-throughput decisions under non-stationary loads [17, 18].

2.2. Service function chain embedding in the edge-cloud environment

The comprehensive review work in [19, 20] has shown that various other research works emphasize the need for SFC and its deployment in environments such as 5G networks, edge-cloud environments. The physical infrastructure hosting SFCs in the edge-cloud environments are generally categorized into *centralized* and *distributed* data centers, in which the data centers are located in the core of the network or moved to the edge, respectively.

The research in [21] focuses on the problem of energy-efficient orchestration of online SFC requests in a multi-domain network. The authors proposed the *energy-efficient online SFC request orchestration across multiple domains* (EE-SFCO-MD) algorithm for minimizing energy consumption on SFC embedding. Pei *et al.* [22] study the service function chain embedding with dynamic VNF placement in a geo-distributed cloud system. They focus on minimizing the embedding cost for SFC requests and optimizing the number of VNF instances for the reduction of the total VNF running time. Kaur *et al.* [23] have discussed the edge-cloud interplay and proposed both energy-aware and QoS-guaranteed mechanisms using SDN in the edge-cloud architecture. The authors proposed handling mechanisms of the network flow and the trade-off between energy efficiency and QoS. However, this work considers the network resources and power consumption of network devices but not the power consumption of servers in data centers, which represents a large amount of total power consumption [24].

Lin *et al.* [25] deployed AI inference service with many deep neural networks, like an SFC for end devices with edge cloud computing, called hybrid computing environments. The optimization algorithm is built with a cost-driven off-loading strategy focusing on reducing the system cost caused by data transmission by each layer. Zhang *et al.* [26] addressed the SFC placement problem by reusing VNFs through deep reinforcement learning based approaches. As a dynamic planning model, this algorithm can reconcile service costs and Quality of Service (QoS) by considering resource constraints and dynamic distribution analysis of VNFs to improve the system performance. For SFC migration as same as [10], Liang *et al.* [27] proposed two SFC migration algorithms to efficiently optimize the average latency of all SFCs in the Jackson network. Ros *et al.* [28] proposed a DRL-based framework that performs intelligent task offloading and VNF placement in MEC for IoT networks, achieving significant reductions in service latency and energy consumption compared to baseline heuristics. Poltronieri *et al.* [29] introduced MECForge, a Deep Q-Network (DQN) agent that maximizes the value-of-information delivered to end users as a holistic resource management criterion. By learning to prioritize important traffic, their approach improves overall user-perceived utility in 5G-edge scenarios [29]. The schedule of SFCs aims to optimize the average latency of all deployed SFCs as well as reasonably fulfill all requirements that are predefined in policies. Such centralized DRL-based orchestrators can adaptively place and route SFCs, outperforming static policies on throughput and delay [28].

2.3. Discussion

The previously mentioned works primarily focus on high-level evaluations and fall short in addressing online and dynamic SFC scenarios, as well as the energy costs associated with both servers and inter-data center networks. To better compare centralized vs. distributed cloud paradigms, this study conducts a quantitative literature-based analysis of their benefits, drawbacks, and trade-offs using three criteria: (i) resource efficiency (maximizing admitted SFCs under limited compute/memory/bandwidth), (ii) energy efficiency (reducing data-center and network power to cut operating costs and environmental impact), and (iii) the trade-off between utilization/energy gains and system complexity due to periodic re-optimization and migrations.

Unlike previous approaches that increase acceptance by splitting VNFs or virtual links, we propose HRE-SFC, a heuristic embedding pipeline that preserves service integrity while effectively handling multi-resource fragmentation. HRE-SFC supports both centralized and distributed data center architectures, operates in an online manner, and explicitly models power consumption of both servers and switches, as well as bandwidth constraints in heterogeneous network environments. At its core, HRE-SFC introduces a novel Pyramid Pointer Arrangement (PPA) - a pointer-swept VNF ordering strategy that generalizes and interpolates between traditional FFD and FFI heuristics, without incurring the orchestration complexity of splitting. Combined with energy-aware VNF consolidation and shortest-feasible-path link embedding, this design achieves energy-proportional behavior, where power consumption per accepted SFC remains nearly constant across varying traffic loads - a property not observed in the existing splitting-based baselines.

3. PROBLEM FORMULATION AND SYSTEM MODELING

Following the previous section that outlines general challenges of SFC embedding in various edge-cloud environments, this section focuses on the specific problem addressed in our work. The mathematical modeling of this problem is presented afterward.

3.1. Use case scenarios in the edge-cloud environments

In this study, we adopt an edge–cloud paradigm, where the cloud system is designed to process services and data generated at the IoT level. At this level, devices such as sensors, actuators, and embedded controllers/computers continuously produce traffic in applications and cyber-physical systems (CPS), including smart grids, healthcare, autonomous vehicles, and smart factories. These traffic flows are not modeled in detail; rather, they are treated as stochastic inputs that naturally enter the network. In this paradigm, an SFC request originates from a *service origination point* in an *ingress node* located at the edge of the network, to the set of VNFs, which demand memory (M_d) and compute (C_d) resources hosted by servers in the data center. Our analysis therefore focuses on SFC embedding within the data-center infrastructure, while IoT devices are considered only as traffic sources that trigger such requests. In such a scenario, the data path of the SFC spans from the edge of the network to the data center, over the core network as shown in Figure 2. Furthermore, two scenarios are considered, namely:

- *Centralized data center* scenario, in which a big data center resides at the core of the network and serves SFC requests coming from all ingress nodes.

- *Distributed data centers* scenario, in which the big data center is divided into several smaller data centers that are located near the edge. Each data center serves SFC requests from local edges.

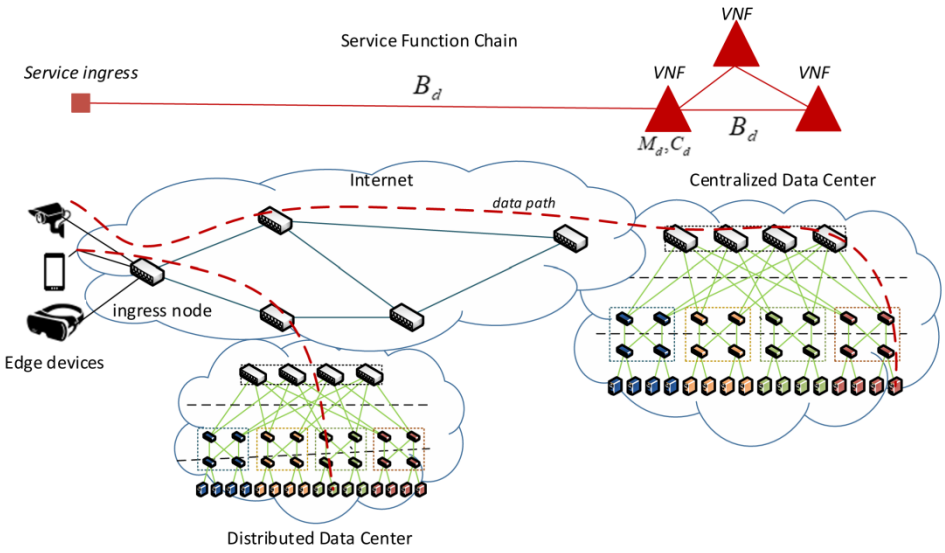


Figure 2. Service function chain originated from the edge.

Here, a comprehensive problem of routing for both data path and resource allocation for computing units (VNFs) over a substrate network is presented. Our research questions consist of two parts:

- Providing similar computer capacity, which DC model, *centralized* or *distributed*, is more resource and energy efficient?

- How to optimize the online SFC embedding problem in a distributed and centralized DC manner?

These questions are analyzed via a comprehensive simulation, in which a realistic network topology is considered, representing a distributed and centralized DC system. SFC requests from end devices originate from ingress points and are mapped into the system via a resource and energy-efficient SFC placement. The algorithm can handle online requests depending on either distributed or centralized DC scenario.

3.2. Edge-cloud SFC embedding problem

In this section, the service chain embedding is modeled as an optimization problem focusing on minimizing the total power consumption of both servers and network devices used to embed SFC requests, while resource efficiency is improved.

Firstly, the *physical substrate* can be modeled as a weighted graph $G^p = (S^p, N^p, L^p)$; where S^p denotes a set of physical servers, N^p denotes a set of network devices (switches), and L^p denotes a set of physical links that interconnect servers and network devices. Resources of physical servers are characterized by memory and CPU capacity. $M_{acap}(S_i^p)$ and $C_{acap}(S_i^p)$ denote available (or leftover) memory and CPU of S_i^p, respectively ($acap$ stands for available

capacity). The resource of physical links is bandwidth. Let us denote $B_{acap}(L_i^p)$ as the available bandwidth of a link L_i^p.

Next, dynamic SFC requests are considered, in which a series of SFC requests join and leave the system overtime. We model the i^{th} SFC as a weighted graph $R_i = (VNF_i, VL_i, t_i, d_i)$, in which t_i and d_i denote the arrival time and duration of SFC, respectively. VNF_i denotes a set of virtual network functions belonging to the i^{th} SFC with the corresponding CPU demand $C_d(VNF_i)$ and memory demand $M_d(VNF_i)$. VL_i denotes the set of bandwidth demands including $vl_i^{s,d} \in VL_i$, which is the bandwidth demand of virtual links from the source vnf_i^s to the destination vnf_i^d; $\{vnf_i^s, vnf_i^d\} \in VNF_i$.

Given a SFC request R_i and a physical data center G^p, embedding R_i onto G^p means to find a subset of S^p, N^p, L^p at time t_i that satisfies the requirement of VNF_i and VL_i. Solving this embedding problem as Integer Linear Programming is NP-hard (Schrijver, 1998). In this work we divide SFC embedding into two sub-problems: (1) *virtual network function mapping* (VNF-M) that maps the VNFs of an SFC request onto the physical servers; and (2) *virtual link mapping* (VLiM), or the *VNF forwarding graph mapping* that maps matrix of link demands onto the substrate links. Let $acap: \{S^p \cup L^p\} \rightarrow G^p$ be a function that returns an available capacity of physical DC, either servers or network devices. Besides, for each SFC request i^{th}, let $dem_i: VNF_i \cup VL_i \rightarrow G^p$ be a function that assigns demand to an element of this SFC. Then, SFC embedding consists of two functions VNF-M and VLiM as presented in Eq. (1).

$$f_i: VNF_i \rightarrow S^p, \text{ and } k_i: VL^i \rightarrow (N^p, L^P) \tag{1}$$

These two mapping functions form an embedding for SFC_i. Computational resources (CPU and memory) required by a $vnf_{ij} \in VNF_i$ must be lower than those available in the physical server hosting it. Likewise, the required bandwidth of a virtual link must be lower than the available bandwidth of all physical links on the path of the DC that the virtual link $vl_i^{s,d}$ is mapped, these conditions are expressed in Eqs. (2) and (3).

$$\forall vnf_{ij} \in VNF_i : dem_i(vnf_{ij}) \le acap[f_i(vnf_{ij})] \tag{2}$$

$$\forall vl_i^{s,d} \in VL_i : \forall L_i^P \in k_i(vl_i^{s,d}) : dem_i(vl_i^{s,d}) \le acap(L_i^p) \tag{3}$$

Let $x_{i,j}^k$ be a binary function indicating whether the $vnf_{i,j} \in VNF_i$ is allocated in server *k*. Then we have the constraints of the functions f_i and k_i as below. One virtual machine is mapped on only one of the servers set S^p as in Eq. (4) if successful or none if unsuccessful.

$$\sum_{k \in \arg(S^p)} x_{i,j}^k \le 1, \ \forall vnf_{i,j} \in VNF_i \tag{4}$$

Let $\texttt{state}_t: \{S^p \cup N^p \cup L^p\} \rightarrow G^p$ denote the function returning a state at time *t* of an element of the substrate network by binary values, which return 1 when turning on (`ON_State`) and 0 otherwise (`OFF_State`). Thus,

- $\texttt{state}(s_i, t)$ - state of the physical server $s_i \in S^p$ at time *t*; $\texttt{state}(s_i, t) \in \{0,1\}$, corresponding to "ON" or "OFF".

- $\texttt{state}(n,t)$ -state of the physical network device $n \in N^p$ at time *t*; $\texttt{state}(n,t) \in \{0,1\}$.

- $\texttt{state}(L_i^p,t)$ - state of the physical links $L_i^p \in k_i(l_{s,d}^{v_i})$ at time *t*; $\texttt{state}(L_i^p,t) \in \{0,1\}$.

All physical elements that host the SFC_i must be turned on as expressed in Eqs. (5) and (6).

$$\forall S_i^p \in f_i(VNF_i) : \texttt{state}(S_i^p,t) = 1 \tag{5}$$

$$\forall vl_i^{s,d} \in VL_i : \forall L_i^p \in k_i(l_i^{s,d}) : \texttt{state}(L_i^p,t) = 1 \tag{6}$$

3.3. Energy modeling

The working state of physical machines, switches, and links in terms of energy consumption are formulated as follows.

3.3.1. Physical machines

As discussed in our previous work [1], the total power consumption model of a physical machine s_i is defined in Eq. (7). Basically, system power consumption consists of two components: (i) the baseline power consumption of the machine in idle mode; and (ii) the power consumption for processing computing jobs, which is proportional to the system utilization.

$$P_S(t) = \sum_{s_i \in S^p} \texttt{state}(s_i,t) \times (\gamma \times U + \delta) \tag{7}$$

where δ is the linear power coefficient when a machine operates with the computing utilization *U* (in percentage), γ is the baseline power of the device. These parameters are based on [30], where $\delta = 1.113$, $\gamma = 205.1$, and depend on the type of the device.

3.3.2. Data center networks

This research makes use of the *Fat-tree* as the network topology in the data center. Fat-tree is a common DC network architecture that can reduce the oversubscription ratio and remove the single point of failures of the hierarchical architecture in the data center network [31, 32]. In a *k*-ary Fat-tree topology, similar k-port switches are allocated in the core, aggregation, and edge layer, whereby the aggregation and edge switches are divided into *k* Performance Optimized Data Centers (POD), each containing two layers of $k/2$ switches. Each switch in the edge layer is connected to $k/2$ servers. Thus, a *k*-ary Fat-Tree data center supports $k^3/4$ servers and has a total number of $5k^2/4$ switches.

We assume that the switches and servers can change the clock frequently to adapt to their working states according to the actual load conditions. For instance, a server is in the `OFF_State` if it does not host any VNF, while a switch can adapt its interface capacity according to the network traffic [30, 33, 34]. Thus, as discussed in [1], the power consumption of the data center network, including switches and links, $P_N(t)$ at time *t* is denoted as the total power of all switches with static power (or baseline power), P_{static}, and the power consumption of their interfaces P_j with the corresponding operating speed, as expressed in Eq. (8).

$$P_N(t) = \sum_{\forall n \in N^p} \texttt{state}(n,t).[P_{static} + \sum_{j=1}^{k} P_j] \tag{8}$$

In this work, the power profile of an energy-aware commercial 24-port HP Enterprise switch [35] is used. The power profile of the switch under 4 working states is summarized in Table 1.

Table 1. Power profile of HP enterprise switch [35].

Operation mode	Power (W)
P_{static}	39
P_{10}-10 Mbps per port	0.42
P_{100}-100 Mbps per port	0.48
P_{1000}-1 Gbps per port	0.9

3.4. Resource and energy optimization formulation

The main objective of this work is to improve resource utilization and reduce the total energy consumption of the physical substrate. The first objective is to maximize the system resource utilization by maximizing the number of accepted SFCs. Let us denote *Z* as the total number of VNFs belonging to all accepted SFCs. As indicated in Eq. (9), the system utilization can be maximized by maximizing *Z*.

$$\textbf{maximize } Z = \sum_{\forall VNF_i \in \{VNF\}} \sum_{\forall vnf_{i,j} \in VNF_i} \sum_{\forall s^k \in S^p} x_{i,j}^k \tag{9}$$

where $x_{i,j}^k$ is the binary function; $x_{i,j}^k = 1$ indicates that resources of server *k* are successfully allocated to the $vnf_{i,j} \in VNF_i$. Let $P_G(t)$ be the total system power consumption at time *t*, $P_S(t)$ be the power consumption of all servers, and $P_N(t)$ be the power consumption of all network nodes that host the above accepted SFCs. $P_G(t)$ can be expressed with $P_G(t) = P_S(t) + P_N(t)$.

Once a set of devices in the physical substrate is found to satisfy the first objective (Eq. (9)), the energy-aware SFC embedding algorithm focuses on finding the best solution with the minimum number of servers and network devices in `ON_State`. Thus, the second objective of the optimization problem is defined as Eq. (10):

$$\textbf{minimize } P_G(t) = P_S(t) + P_N(t) \tag{10}$$

The constraints of the SFC embedding objectives are expressed in Eqs. (2), (3) and (4).

4. RESOURCE AND ENERGY-AWARE SERVICE FUNCTION CHAINING

4.1. SFC embedding strategies

As discussed above, solving the optimization problem described in Eqs. (9) and (10) is *NP*-hard and cannot be done in real-time. In this section, we address two heuristic SFC embedding strategies towards *energy efficiency* and *resource efficiency*. In each strategy, energy-aware node (VNF-CC) and link mapping (VNF-FGE) algorithms are conducted, which will be addressed later in Section 4.2. We adopted the traditional mapping technique, in which the VNF embedding is done through the mapping of the VNF on a virtual machine (VM) on a physical server. A separate VM hosts each VNF. It is also assumed that the SFC requests arrive

and leave the system dynamically. As our goal is to maximize resource utilization while minimizing energy consumption in distributed and centralized DC contexts, two problems need to be addressed as follows.

Selecting DC for the data path from end devices to the SFC that resides in DC. As long routing path often suffers from congested links, in the distributed DC scenarios, our algorithm prefers the DC that is in proximity of the ingress point.

Selecting group of servers in the chosen DC for VNF mappings. After a DC has been decided, the whole problem returns to the basic SFC embedding, which influences most of the energy consumption and resource consumption of the entire system. The following strategies are applied to achieve our goal of jointly optimizing energy and resources.

4.1.1. Energy efficiency - strategically placing VNFs into physical servers so that overall power consumption is minimal

As modeled in Section 3.3, general power consumption is mainly attributed to physical servers and network devices in DCs. To minimize this metric, the overall idea relies on efficiently placing VNFs into a minimum number of physical servers, and rerouting virtual links through the least network devices, so that the rest can be put in standby mode. The former problem is modeled as the *bin-packing* problem [36], of which our algorithm is inspired by *First-Fit Decreasing* (FFD), which has been proven a fast and efficient solution for bin-packing [37]. The details of the proposed SFC embedding algorithm will be presented in Section 4.2.

4.1.2. Resource efficiency - Strategically arranging VNFs in a set of servers so that maximum number of SFC can be accepted

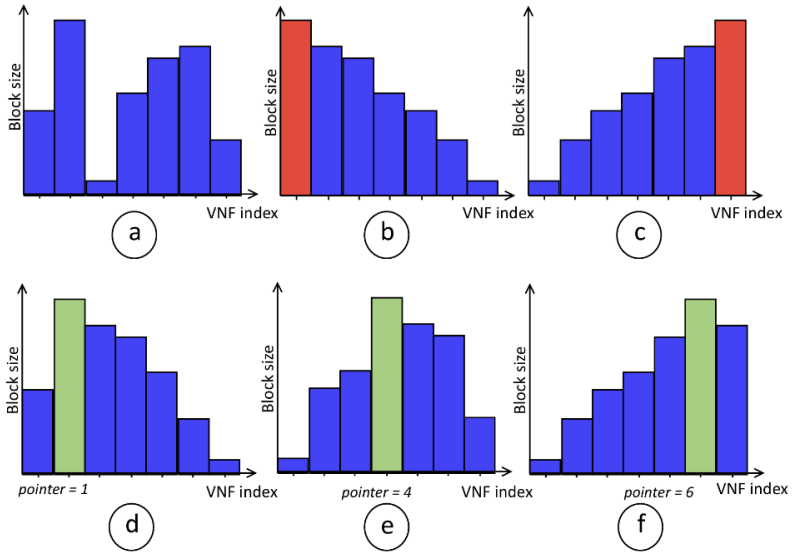


Figure 3. VNF reordering by In/Out bandwidth and CPU demand (block size). (a) Unsorted; (b) and (c) Decreasing and increasing order; (d)-(f) Pyramid sorting by shifting the peak.

Another objective of this research is resource efficiency. That is, a maximum number of SFCs can be embedded into a given physical substrate network and servers. This work takes into account two kinds of resources, which are *CPU of servers* and *bandwidth of physical links*. Thus, the resource-efficient SFC embedding can be formulated as the *2-dimensional bin*

packing problem, which is shown to be NP-hard [38]. Instead, we propose a resource-efficient heuristic SFC embedding strategy, which will be discussed below.

Network bandwidth is the first limiting resource. Packing VNFs onto the *fewest active servers* cuts power and, thanks to VNF proximity, slashes inter-server traffic. Heuristics such as *First-Fit Decreasing* therefore embed high-bandwidth VNFs first. This fails, however, when the chosen host is already CPU-bound, forcing the VNF onto a distant node and saturating links (Figures 3(b)–(c)). Spreading VNFs across the data-center eases link stress [22] but negates energy savings by waking idle machines. Since exact multi-resource placement is NP-hard, we propose the *Pyramid Pointer Arrangement (PPA)* heuristic, which balances traffic without scattering VNFs across many servers.

Firstly, we define the bandwidth metric of a VNF as the total bandwidth demand, including incoming and outgoing traffic of all logical links connected to that VNF. To balance bandwidth and CPU resources of each VNF request, the strategy will apply to both bandwidth demand and CPU demand, combined into a resource block in 2D space. The combined demand resource of a VNF is defined as the multiplication of bandwidth and CPU demand, which is called the *block size*. Our mapping approach adheres to the following principles:

- The VNF demands are ordered from largest to smallest and the mapping first follows the *First-Fit Decreasing* (FFD) strategy, which maps the demand with largest block size first.

- If the mapping fails, the original FFD stops, and the request is rejected. In contrast, our approach reorders the VNF demand sequence and applies PPA to seek an optimal solution. PPA explores all possible mappings by combining FFD and *First-Fit Increasing* (FFI), this method enhances the likelihood of acceptance.

Figures 3(d)–(f) sketch the PPA workflow. From the unsorted chain in (a), the scheduler first tries a straightforward FFD pass - placing VNFs from largest to smallest, right-to-left. (b) If that fails, PPA slides a “pointer” along the list and reorders it into a pyramid: VNFs to the left of the pointer rise in size (FFI), those to the right fall (FFD). Each pointer position (d–f) recenters demand, spreads bandwidth/CPU load more evenly across links, and increases the likelihood of a feasible 2-D packing.

4.2. SFC embedding algorithms

The *Heuristic Resource and Energy-Aware SFC Embedding Algorithm* consists of three procedures, we consolidate our strategy into **HRE-SFC**, a single pipeline:

(i) **PPA ordering.** Compute VNF block sizes (CPU and VNF’s neighbor-bandwidth), sweep a pointer to generate pyramid orderings that balance CPU/bandwidth, and try largest-first permutations to escape fragmentation.

(ii) **Energy-aware VNF mapping.** First-fit into minimal server groups (prefer already-ON machines) to maximize consolidation.

(iii) **Link embedding.** Route high-bandwidth virtual links first on shortest feasible paths that minimize incremental switch activations.

The heuristic progress of HRE-SFC is shown in algorithm 1 through four steps as follows.

- Step 1 (line 2): Upon the arrival of an SFC request R_j^v, the `PPA` routine produces a list of VNF permutations $orders$. Each permutation reshapes inter-VNF bandwidth patterns and will be examined independently.

- Step 2 (lines 3-4): For every order $ord \in orders$, `VNFMapping` enumerates all feasible server groups $groups$ to map VNFs on physical servers in the DC (details in Procedure 1)

- Step 3 (line 5): Given a particular map_{VNF}, the `GraphEmbedding` phase tries to route every virtual link to physical links (details in Procedure 2). It returns a pair (map_{VL}, ok); the Boolean flag ok is **true** only if all virtual links embed successfully.

Algorithm 1 Heuristic resource and energy aware SFC embedding algorithm

Require: substrate G^p, request R_j^v

1: $orders \leftarrow$ PPA(R_j^v) ▷ VNF-order generator
2: **for** $ord \in orders$ **do**
3: $groups \leftarrow$ `VNFMapping`(G^p, R_j^v, ord)
4: **for** $map_{VNF} \in groups$ **do**
5: $(map_{VL}, ok) \leftarrow$ `GraphEmbedding`(G^p, R_j^v, map_{VNF}) ▷ reserve resources
6: **if** ok **then**
7: `commit`(G^p, map_{VNF}, map_{VL})
8: **return true** ▷ embedding succeeded
9: **return false** ▷ no feasible placement found

- Step 4 (lines 6-9): At the first successful pair (map_{VNF}, map_{VL}), the algorithm `commit` the corresponding CPU, memory and bandwidth resources and terminates with (**return true**). If link embedding fails, the algorithm continues with the next server group; when all groups are exhausted, it falls back to the next VNF order. Should every order be tried without success, the routine exits with (**return false**).

4.2.1. Pyramid Pointer Arrangements – PPA

As described in Section 4.1, to improve the acceptance rate of SFC embedding, the PPA heuristic arranges VNFs in a specific order that balances both bandwidth and CPU usage across the network. The idea is to reorder the VNFs of an incoming SFC request based on *neighbor bandwidth* (B_n), which captures the sum of all incoming and outgoing virtual link bandwidths connected to each VNF. Formally, for a VNF vnf_i^j in the i^{th} SFC, B_n is computed as:

$$B_n(vnf_i^j) = \sum_{k \in N_j} vl_i^{j,k} \tag{11}$$

where N_j is the set of neighbor VNFs connected to vnf_i^j via virtual links $vl_i^{j,k}$. The block size of a VNF is defined as the product of its B_n and CPU demand C_d, reflecting the joint load the VNF imposes on both compute and network resources.

The PPA procedure then generates alternative VNF orderings by iteratively applying a pointer-based sorting strategy. Given a list of VNFs and their corresponding block sizes, a pointer p is first placed at the position of the VNF with the largest block size. The list is then

divided into two sub-lists: the left sub-list contains VNFs with smaller block sizes that precede p, and is sorted in ascending order; the right sub-list contains VNFs that follow p and is sorted in descending order. These two sub-lists are then concatenated, with the pointer element at the center, to form a new permutation of the VNF sequence (see Figures 3(d)-(f)).

This reordering process is repeated with different values of p across the list, producing a set of candidate VNF arrangements. Each arrangement is subsequently passed to the `VNFMapping` procedure to evaluate its feasibility. This mechanism increases the chances of finding valid embedding, especially under fragmented resource conditions.

4.2.2. Virtual network function mapping

Once VNFs have been sorted, this procedure performs VNF-CC that finds possible locations for VNFs on physical servers in the DC. The virtual link interconnecting two neighbor VNFs hosted by two servers are classified into: (i) *near link*, if the link traverses through one edge switch of the Fat-Tree (see Figure 2); (ii) *middle link*, in which the virtual link should span over an aggregation switch; and (iii) *far link*, in which the link should traverse through a core switch in the Fat-tree architecture. It is easy to observe that the far link interconnects two VNFs located in two different PODs, while near and middle links interconnect two VNFs residing in the same POD.

Furthermore, the servers that host VNFs of an SFC belong to *near* or *middle group* if the data between them are exchanged via near or middle links, respectively. In contrast, the servers hosting the VNFs are in a *far group* or *mixed group* if they belong to two or more than two separate PODs, respectively. The VNF-mapping phase (Procedure 1) proceeds in four sub-steps,

Procedure 1 `VNFMapping`(G^p, R_j^v, $order_{VNF}$)

1: $listPG \leftarrow \emptyset$ ▷ feasible placements
2: $DGR \leftarrow$ `sortByHop`(G^p) ▷ DCs nearest to ingress first
3: **for** $dc \in DGR$ **do**
4: **for** $gr \in$ `serverGroups`(dc) **do** ▷ Near → Middle → Far
5: $visited \leftarrow \emptyset,\ mapping \leftarrow \emptyset,\ ok \leftarrow$ **true**
6: **for** $v \in order_{VNF}$ **do**
7: $s \leftarrow$ `firstFit`$(v, gr, visited)$
8: **if** $s = None$ **then**
9: $ok \leftarrow$ **false**; **break**
10: $visited \leftarrow visited \cup \{s\}$
11: $mapping \leftarrow mapping \cup \{v \mapsto s\}$
12: **if** ok **then**
13: $listPG \leftarrow listPG \cup \{mapping\}$
14: **return** `sortByON`($listPG$)

shown schematically in the pseudocode below:

- Step 1 (lines 2-3): The helper `sortByHop` builds a list of data center candidates DGR that contains every DC whose ingress link can deliver the requested bandwidth of the first virtual link. The list is sorted in ascending hop count so that the nearest DC is examined first.

- Step 2 (lines 4-5): For each chosen DC, `serverGroups` returns three server groups ordered by proximity (*Near → Middle → Far*). Each group provides enough cumulative CPU

and memory to host all VNFs. Working sets $visited$ and $mapping$ are cleared at the start of every group.

- Step 3 (lines 6-11): VNFs are processed in the prescribed order. The call `firstFit` places a VNF on the first unused server in the current group that still meets its CPU/MEM demand. If no such server exists (line 8), the algorithm abandons the current group and proceeds to the next one.

- Step 4 (lines 12-14): Whenever every VNF is successfully mapped, the placement is appended to $listPG$. After all groups and DCs have been scanned, $listPG$ is sorted by the number of servers already in the `ON_State` (`sortByON`). The chosen mapping, therefore, reuses active machines and minimizes additional power consumption.

4.2.3. VNF forwarding graph embedding

To reduce the consumed energy of network function virtualization infrastructure, a forwarding graph embedding in procedure 2 has been developed based on the concept of *Elastic Tree* [39]. Elastic Tree reduces the power consumption of the data center network by maintaining a minimal logical topology on top of the Fat-Tree based on actual traffic demands.

Procedure 2 `GraphEmbedding`(G^p, R_j^v, $mapping_{VNF}$)

1: $L_j^v \leftarrow$ `sortDescBW`(L_j^v) ▷ high-BW links first
2: $result \leftarrow \emptyset$
3: **for** $vl \in L_j^v$ **do**
4: $(s, d) \leftarrow$ `endpoints`$(vl, mapping_{VNF})$
5: $path \leftarrow$ `findPath`$(s, d, bw(vl), G^p)$ ▷ shortest feasible route
6: **if** $path = None$ **then**
7: **return** (∅, **false**) ▷ embedding fails
8: $result \leftarrow result \cup \{vl \mapsto path\}$
9: **return** ($result$, **true**)

Firstly, the matrix of virtual links L_j^v belonging to request R_j^v is sorted in non-increasing bandwidth order (`sortDescBW`). Routing the heaviest flows first helps reserve residual capacity for lighter ones. For each virtual link vl (line 3), the helper `endpoints` retrieves the physical servers that host its source and destination VNFs (as decided by the VNF-mapping phase). The routine `findPath` then selects the shortest residual-capacity route whose extra active switches add the least incremental power. When no feasible path exists (line 6), the algorithm aborts immediately and returns **false**; control will pass to the next candidate server group produced by VNF-mapping. If every virtual link is routed successfully, the procedure returns the set $result$ that maps each $vl \in L_j^v$ to its physical path, together with the flag **true**. The residual network, therefore, forms the minimal logical topology required to carry the SFC traffic, mirroring the elastic-tree philosophy.

5. PERFORMANCE EVALUATION

The proposed HRE-SFC and other state-of-the-art algorithms are evaluated under different distributed and centralized scenarios in this section.

5.1. Simulation scenarios

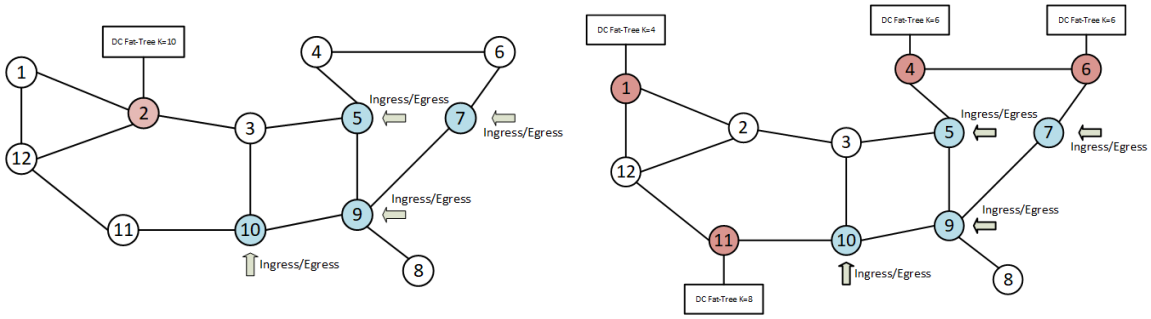
5.1.1. Performance criteria

The performance of the SFC embedding strategy is evaluated based on three criteria as follows:

- *Resource efficiency*: We use *acceptance ratio* for evaluation as it indicates how many incoming SFC requests can be accommodated within given limited resources, such as bandwidth, CPU, and memory. It is measured by the ratio of the number of *accepted SFCs* over *requested SFCs* in the period of simulation time.

- *Energy efficiency*: The total power consumption of the edge-cloud substrate, the average power consumption of an SFC, and of a VNF are investigated according to the system utilization, which reflects how much system resources (CPU) have been consumed at a given time.

5.1.2. Experimental parameters



(a) Centralized topology (b) Distributed topology

Figure 4. Two topology cases - Abilene topology [40].

We use the abilene topology provided by SNDLib [40], a library of test instances for survivable fixed telecommunication network design, which reflects a real network widely adopted in research for its realism and publicly available configuration data [41]. The abilene topology is shown in Figures 4(a)–(b), which has 11 physical nodes.

The topology of an SFC request is randomly generated using Waxman algorithm [42] with the probability that there exists a virtual link between two VNFs u and v of an SFC is $p(u,v) = \alpha e^{-d(u,v)/(\beta L)}$. The parameters α and β are in the range of (0,1), d is the distance in Cartesian coordinates among VNF u and v, and L is the maximum distance between any two nodes in the graph. The parameter α presents the probability of a link existing between any two nodes in a graph, and β represents the ratio of long links to short links. In this simulation, we set $\alpha = \beta = 0.5$ [43] for average connectivity and link distance.

The arrival of SFC requests is modeled as a Poisson process with an average rate of λ SFCs per hour, a common assumption grounded in prior empirical studies [15, 21, 44]. While seemingly simplistic, this approach has proven effective in capturing the inherent randomness and burstiness of traffic patterns observed in real-world network environments. The average service time of an SFC is exponentially distributed with $T_s = 2$ hours. The number of VNFs per SFC request uniformly varies from 4 to 20. All traffic demands between VNF pairs are also

uniformly distributed between 10 Mbps and 90 Mbps. The computing utilization usage per VNF is distributed randomly in the range of 15 – 30 %.

5.1.3. Experimental scenarios

The advantages and disadvantages of the two following scenarios are to be investigated: (i) *centralized cloud*, where a larger data center is placed at the center of the network; and (ii) *distributed cloud*, where several smaller data centers are moved nearer to users, at the edge of the network.

The performance of the system with the same computing capacity is compared. In both scenarios, 4 ingress nodes are defined, which are Nodes 7, 5, 9, 10 in the Abilene topology (see Figures 4(a)–(b)). For the centralized DC scenario, a large fat tree topology with k = 10, equivalent to 250 servers residing at the center in Node 2. On the other hand, in the distributed DC scenario, 4 different DCs are placed near the ingress nodes with a total of 252 servers. Each server has 8 CPUs and 64 GB of memory.

Furthermore, to see the impact of the core network topology on the cloud system performance, two network setups are considered: (i) *limited link capacity*, in which the capacity of the physical links of the core network is limited to 100 Mbps; and (ii) *infinite link capacity* (BW=Inf), which is the ideal case, where the capacity of the physical link is infinite.

5.1.4. Baselines

Our proposed Heuristic Resource and Energy Aware SFC Embedding Algorithm (HRE-SFC) is compared with the two meta-heuristic baselines:

- *VNF splitting*, in which a VNF can be split up into multiple sub-VNFs based on the available CPU resources of physical servers. The representative of this strategy is the joint online composition and embedding of the VNF Chain (VNFG) [8, 16]. Doing so increases the chance that VNF, or SFC, can fit into the available resources, thus increasing the acceptance rate. However, some services cannot be processed in parallel by multiple functions.

- *Virtual link splitting* that splits a large virtual link demand into multiple virtual links with smaller bandwidth demands in case the physical link is bottlenecked. Some work follows this direction, such as online parallelized SFC Orchestration (ONP) [9, 15].

Splitting functions or links offers clear benefits for SFC embedding, as it allows the corresponding SFC to be more easily accommodated within the physical substrate. However, as mentioned earlier in Section 2, splitting VNF or virtual links comes with a trade-off of high complexity for the scheduler. In addition, sub-SFCs may not be applicable in most applications, such as video streaming.

For a comprehensive evaluation, we consider eight experimental scenarios that combine two data-center architectures (distributed vs. centralized) with three SFC-embedding strategies:

(i) Dist-HRE-SFC and (ii) Cent-HRE-SFC apply the HRE-SFC algorithm in distributed and centralized clouds, respectively.

(iii) Dist-HRE-SFC - *BW=Inf* and (iv) Cent-HRE-SFC - *BW=Inf* repeat those two cases while assuming unlimited core-link bandwidth to isolate network-congestion effects.

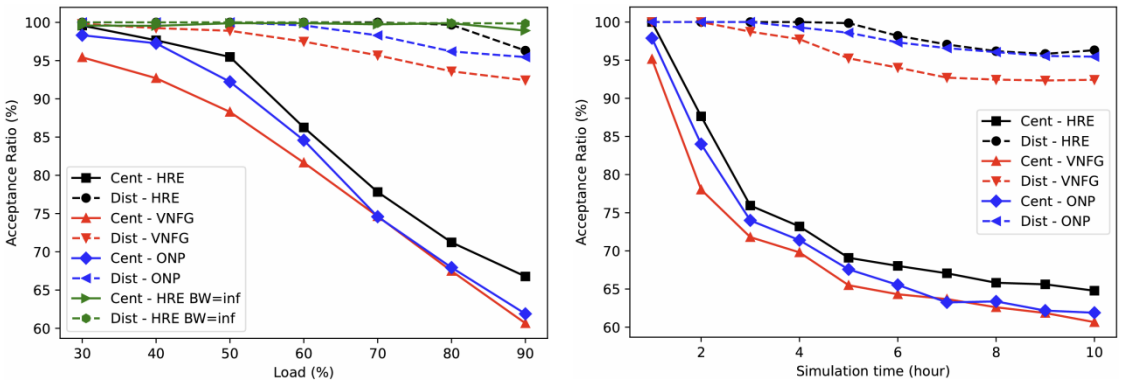
(v) Dist-VNFG and (vi) Cent-VNFG use the joint online VNF chain composition & embedding heuristic [8, 16] under the two cloud architectures.

(vii) Dist-ONP and (viii) Cent-ONP test the online paralleled SFC orchestration method [9, 15] with a splitting threshold $k_{sub} = 30$ Mbps in distributed and centralized settings.

5.2. Simulation results

This section discusses our performance analysis of the proposed strategy with some algorithms in terms of resources, energy consumption.

5.2.1. Resource efficiency



(a) Acceptance ratio versus system load

(b) Acceptance ratio varied with simulation time with λ = 9 (load = 90 %)

Figure 5. Acceptance ratio under varying system load and over time.

Figure 5 shows the distributed architecture consistently achieves higher acceptance ratios than centralized ones, especially under high load. This improvement stems from reduced link congestion due to the proximity of edge data centers. Additionally, the HRE-SFC algorithm maintains a superior acceptance rate thanks to the PPA strategy, which increases flexibility without requiring service function splitting. In Figure 5(a), under varying offered loads (with difference λ) distributed deployments outperform centralized ones across all embedding algorithms. This is explained as distributed DCs allow SFC to be placed scattering over the topology, thus preventing the bandwidth bottleneck that usually occurs when most of the traffic concentrates at one place as in the case of centralized DC. The effect is evident when comparing Cent-HRE and Dist-HRE: with 100 Mbps core links, Dist-HRE supports nearly double the number of SFCs, while under unlimited bandwidth, both scenarios converge. This confirms that, given equal computing capacity, distributed placement alleviates backbone bottlenecks and accepts more SFCs.

When comparing the SFC algorithms, HRE-SFC outperforms the others even without dividing the SFC into smaller ones. This is because HRE employs the pyramid pointer arrangement (Procedure 1), which shuffles the order of VNFs to reduce interconnections within the SFC, thereby increasing the likelihood of a successful mapping. The PPA strategy not only improves the acceptance rate but also avoids splitting the SFC - a process that can be unsuitable or even detrimental for certain applications, such as stateful services or video streaming.

Figure 5(b) illustrates the decline in acceptance ratio over time as the system continues to operate. Initially, Dist-HRE maintains an acceptance ratio close to 100 % for the first five hours. As can be observed, in general, as SFCs dynamically join and leave the system, resources

become increasingly fragmented, making it more difficult to accommodate new VNF requests under the same load. HRE attempts multiple VNF placement strategies to maximize acceptance. Compared to Dist-ONP, Dist-HRE demonstrates slightly higher efficiency, even though ONF splits the original SFC into multiple sub-SFCs. On the other hand, although the number of servers between the centralized and distributed scenarios, distributed DCs employ more switches (190 vs. 125). This provides greater link capacity for graph embedding (Section 4.2), which further mitigates bandwidth fragmentation and raises acceptance ratios in distributed deployments.

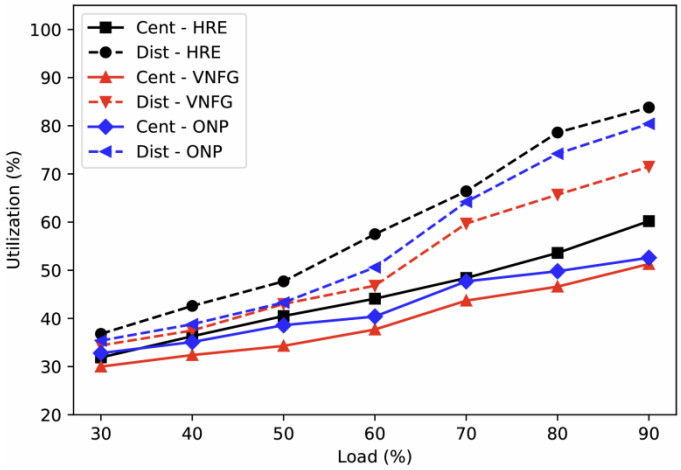
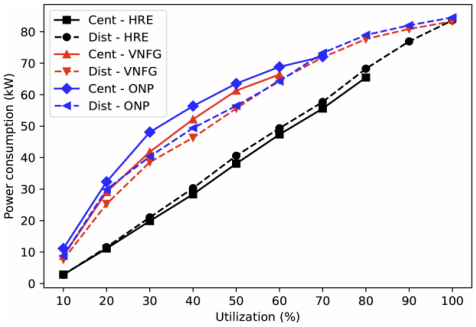


Figure 6. Utilization of the system under different loads (%).

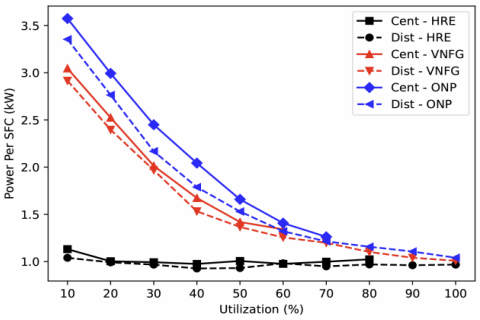
Figure 6 further presents the utilization of the system at different load levels. As the load increases, distributed approaches yield better resource utilization due to more flexible VNF placement and reduced bandwidth fragmentation. HRE-SFC outperforms other methods in both centralized and distributed setups, validating its effectiveness in balancing resource usage. As the system load increases from 30% to 90%, utilization also increases correspondingly. The Dist-HRE approach achieves the highest utilization, followed by Dist-ONP and Dist-VNFG, demonstrating that distributed architectures can more effectively utilize resources than their centralized counterparts. Among centralized methods, Cent-HRE outperforms Cent-ONP and Cent-VNFG, indicating that our heuristic PPA resource allocation strategies enhance utilization. In addition, as system load increases, centralized approaches generally exhibit lower resource utilization than distributed methods, reinforcing the benefits of distributing workload across multiple data centers. This observation has the same reason that the number of switches in the distributed fat-tree-based data centers is higher than those of the centralized one, so that more VNFs can be accepted.

5.2.2. Energy efficiency

In terms of energy efficiency, we assess the total power consumption of the entire system as well as the power consumption of each SFC across different utilization levels, thereby quantifying the power required for a given amount of resource.



(a) Total system power consumption under different utilization levels



(b) Power consumption per serving SFC of the substrate network with different utilization levels

Figure 7. Energy efficiency simulation results.

Figure 7(a) reports total power across utilization levels, from low (utilization = 30 %, λ = 3) to high (utilization = 90 %, λ = 9). HRE-SFC is the most energy-efficient in both total power and power per SFC, as depicted in Figures 7(a)–(b). Furthermore, Figure 7(b) shows that HRE-SFC maintains consistent power consumption per SFC regardless of utilization, whereas other algorithms exhibit higher power consumption per SFC under low utilization conditions. The reason is that HRE-SFC consolidates VNFs onto fewer servers (see line 5 of procedure 2) and embeds links on the shortest feasible paths (as indicated in line 7 of procedure 3), further reducing energy consumption across links and preventing the activation of unnecessary network switches. Thus, HRE-SFC maintains the energy proportional property by consuming only the energy required to accommodate the incoming SFC demands, whereas other approaches consume more energy at the same utilization levels, particularly under low-load conditions.

In addition, Figure 7(a) shows that the total power grows quasi-linearly with the number of active servers, while Figure 5(a) indicates how many SFCs can be packed for a given CPU utilization U. Combining the two curves, we observe an inflection at $U \approx 70$ %: below this, each additional 1 % of CPU load raises the total power by 0.6 %, but above it the slope nearly doubles as extra VNFs force new servers and top-of-rack switches to turn on. Thus, higher utilization helps until the knee, after which energy per SFC rises sharply, motivating the hybrid policy (Section 5) that targets 65–75% utilization, consolidates at low load, and delays edge-DC activation until the knee is reached.

Table 2. Total energy consumption throughout the entire simulation period, utilization 90 %.

Scenarios	Cent-HRE	Dist-HRE	Cent-VNFG	Dist-VNFG	Cent-ONP	Dist-ONP
Total energy consumption (kWh)	6.77	8.53	9.67	10.84	10.36	11.48

Furthermore, as shown in Figure 7(a) and Table 2, for a given utilization level, although distributed setups yield higher acceptance ratios, centralized data centers demonstrate better energy efficiency compared to distributed data centers. This is due to distributed scenarios require more active network devices for virtual links interconnecting VNFs, leading to higher energy consumption of network devices.

5.3. Discussions

Based on the evaluation criteria and test scenarios, centralized and distributed DC strategies outperform each other in specific cases, as summarized below:

- *Resource efficiency*: Figures 5(a) and 6 show that *Dist* outperforms *Cent* thanks to the proximity of the DC to the edge, reducing link bottlenecks. With sufficient core bandwidth, *HRE-SFC* achieves peak efficiency in utilization and acceptance ratio in both centralized and distributed scenarios. However, under constrained bandwidth, distributed approaches excel by minimizing core reliance, improving load distribution and scalability. Notably, the *Dist-HRE* surpasses other algorithms in SFC acceptance. This suggests that the distributed approach of HRE-SFC is more effective at handling SFC requests, maintaining a higher acceptance ratio compared to alternative methods.

- *Energy efficiency*: The results represented in Figures 7(a)–(b) and Table 2 show that the energy consumption of distributed scenarios is higher than centralized ones. The HRE-SFC outperforms other algorithms thanks to its energy-aware strategy. Moreover, the system energy consumption of HRE-SFC is in proportion to system utilization, while the energy consumption under low utilization of other algorithms is much higher as they do not exhibit the energy-proportion property.

Overall, HRE-SFC in distributed cloud offers the best balance between resource and energy efficiency. However, the study has limitations: it uses a single backbone topology (Abilene) and Poisson arrivals; it omits costs such as VNF migration, state synchronization, and wake-up delays; and it focuses on acceptance and power, leaving latency, SLA compliance, and orchestration cost in large-scale, irregular settings for future work.

6. CONCLUSIONS

This work discusses the advantages and disadvantages of centralized and distributed edge-cloud paradigms. A heuristic SFC embedding mechanism is proposed in the work, which can accommodate the edge-cloud system under network dynamics when SFC requests join and leave the system over time, and demonstrates its effectiveness in improving both energy consumption and resource efficiency.

The evaluation revealed several key insights. First, data centers near the edge of the network help remarkably improve system performance with respect to resource efficiency. With limited physical resources, a careful plan of data center locations can improve the system's performance significantly. The physical link capacity is one of the important factors that decides the performance of the edge-cloud infrastructure. The centralized data center paradigm can accommodate incoming demands well if sufficient capacities are provided. However, it might be difficult for an operator to over-provision its network capacity to meet this requirement.

Second, our pyramid pointer arrangement strategy deals with the resource efficiency problem better than other approaches. In all centralized and distributed scenarios, the proposed strategy generally has higher performance in terms of resources and energy.

In future research, edge-cloud data center planning and placement strategy can be studied and developed to improve the overall system performance in terms of both energy consumption, QoS guarantees, latency and resource efficiency, targeting hybrid, dynamically reconfigurable DC topologies that couple a large core cloud with elastic micro-DCs at the edge. Also, it is

interesting to see how different SFC embedding strategies and algorithms can work in a real edge-cloud environment. This embedding strategy can be practically deployed in Kubernetes-based NFV orchestration frameworks to improve energy-performance trade-off, especially in multi-access edge computing (MEC) deployments for 5G/6G networks.

Acknowledgements. This work is funded by the Hanoi University of Science and Technology (HUST) under project T2023-PC-038.

CRediT authorship contribution statement. Ma Viet Duc: Formal analysis, Writing – original draft. Nguyen Trung Kien: Methodology, Software. Dao Dai Hiep: Data curation, Validation. Nguyen Tai Hung: Investigation, Validation. Nguyen Huu Thanh: Conceptualization, Methodology, Supervision, Writing – review & editing.

Declaration of competing interest. The authors declare that there is no conflict of interest in this article.

REFERENCES

1. Nguyen Huu T., Pham Ngoc N., Truong Thu H., Tran Ngoc T., Nguyen Minh D., Nguyen V. G., Nguyen Tai H., Ngo Quynh T., Hock D., Schwartz C. - Modeling and experimenting combined smart sleep and power scaling algorithms in energy-aware data center networks. Simul. Model. Pract. Theory, **39** (2013) 20-40. https://doi.org/10.1016/j.simpat.2013.05.011.
2. Bolla R., Bruschi R., Davoli F., Cucchietti F. - Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures. IEEE Commun. Surv. Tutor., **13**(2) (2010) 223-244. https://doi.org/10.1109/surv.2011.071410.00073.
3. Bonomi F. - The eighth ACM international workshop on vehicular inter-networking (VANET), (2011) 13-15.
4. Bonomi F., Milito R., Zhu J., Addepalli S. - Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, (2012) 13-16. https://doi.org/10.1145/2342509.2342513.
5. Gil Herrera J., Botero J. F. - Resource allocation in NFV: A comprehensive survey. IEEE Trans. Netw. Serv. Manage., **13**(3) (2016) 518-532. https://doi.org/10.1109/tnsm.2016.2598420.
6. Cohen R., Lewin-Eytan L., Naor J. S., Raz D. - 2015 IEEE conference on computer communications (INFOCOM), IEEE, (2015) 1346-1354. https://doi.org/10.1109/infocom.2015.7218511.
7. Zhao D., Liao D., Sun G., Xu S. - Towards resource-efficient service function chain deployment in cloud-fog computing. IEEE Access, **6** (2018) 66754-66766. https://doi.org/10.1109/access.2018.2875124.
8. Li J., Shi W., Ye Q., Zhuang W., Shen X., Li X. - 2018 IEEE global communications conference (GLOBECOM), IEEE, (2018) 1-6. https://doi.org/10.1109/glocom.2018.8647700.
9. Wang R., Yu X., Wu Q., Yi C., Wang P., Niyato D. - Efficient deployment of partial parallelized service function chains in CPU+DPU-based heterogeneous NFV platforms. IEEE Trans. Mob. Comput., **23**(10) (2024) 9090-9107. https://doi.org/10.1109/tmc.2024.3357796.
10. Pham T.-M. - Optimizing service function chaining migration with explicit dynamic path. IEEE Access, **10** (2022) 16992-17002. https://doi.org/10.1109/access.2022.3150352.
11. Erbati M. M., Tajiki M. M., Schiele G. - Service function chaining to support ultra-low latency communication in NFV. Electronics, **12**(18) (2023) 3843. https://doi.org/10.3390/electronics12183843.

12. Wang X., Wang X., Shi Y., Wu D., Ma L., Huang M. - Core-selecting auction-based mechanisms for service function chain provisioning and pricing in NFV markets. Comput. Netw., **222** (2023) 109557. https://doi.org/10.1016/j.comnet.2023.109557.
13. Lin R., He L., Luo S., Zukerman M. - Energy-aware service function chaining embedding in NFV networks. IEEE Trans. Serv. Comput., **16**(2) (2022) 1158-1171. https://doi.org/10.1109/tsc.2022.3162328.
14. Chintapalli V. R., Partani R., Tamma B. R., C S. R. M. - Energy efficient and delay aware deployment of parallelized service function chains in NFV-based networks. Comput. Netw., **243** (2024) 110289. https://doi.org/10.1016/j.comnet.2024.110289.
15. Sun G., Chen Z., Yu H., Du X., Guizani M. - Online parallelized service function chain orchestration in data center networks. IEEE Access, **7** (2019) 100147-100161. https://doi.org/10.1109/access.2019.2930295.
16. Azhdari A., Ebrahimzadeh A., Afrasiabi S. N., Szabó R., Mouradian C., Li W., Glitho R. H. - GLOBECOM 2023 - 2023 IEEE global communications conference, IEEE, (2023) 5384-5390. https://doi.org/10.1109/globecom54140.2023.10437838.
17. Xiao Y., Zhang Q., Liu F., Wang J., Zhao M., Zhang Z., Zhang J. - Proceedings of the International Symposium on Quality of Service, ACM, (2019) 1-10. https://doi.org/10.1145/3326285.3329056.
18. Dolati M., Hassanpour S. B., Ghaderi M., Khonsari A. - IEEE INFOCOM 2019 - IEEE conference on computer communications workshops (INFOCOM WKSHPS), IEEE, (2019) 879-885. https://doi.org/10.1109/infcomw.2019.8845171.
19. Hantouti H., Benamar N., Taleb T. - Service function chaining in 5G & beyond networks: Challenges and open research issues. IEEE Netw., **34**(4) (2020) 320-327. https://doi.org/10.1109/mnet.001.1900554.
20. Adoga H. U., Pezaros D. P. - Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges. Future Internet, **14**(2) (2022) 59. https://doi.org/10.3390/fi14020059.
21. Sun G., Li Y., Yu H., Vasilakos A. V., Du X., Guizani M. - Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks. Future Gener. Comput. Syst., **91** (2019) 347-360. https://doi.org/10.1016/j.future.2018.09.037.
22. Pei J., Hong P., Xue K., Li D. - Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system. IEEE Trans. Parallel Distrib. Syst., **30**(10) (2018) 2179-2192. https://doi.org/10.1109/tpds.2018.2880992.
23. Kaur K., Garg S., Aujla G. S., Kumar N., Rodrigues J. J. P. C., Guizani M. - Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. IEEE Commun. Mag., **56**(2) (2018) 44-51. https://doi.org/10.1109/mcom.2018.1700622.
24. Wang L., Zhang F., Aroca J. A., Vasilakos A. V., Zheng K., Hou C., Li D., Liu Z. - GreenDCN: A general framework for achieving energy efficiency in data center networks. IEEE J. Sel. Areas Commun., **32**(1) (2013) 4-15. https://doi.org/10.1109/jsac.2014.140102.
25. Lin B., Huang Y., Zhang J., Hu J., Chen X., Li J. - Cost-driven off-loading for DNN-based applications over cloud, edge, and end devices. IEEE Trans. Ind. Inform., **16**(8) (2019) 5456-5466. https://doi.org/10.1109/tii.2019.2961237.
26. Zhang Y., Zhang F., Tong S., Rezaeipanah A. - A dynamic planning model for deploying service functions chain in fog-cloud computing. J. King Saud Univ. Comput. Inf. Sci., **34**(10) (2022) 7948-7960. https://doi.org/10.1016/j.jksuci.2022.07.012.
27. Liang W., Cui L., Tso F. P. - Low-latency service function chain migration in edge-core networks based on open Jackson networks. J. Syst. Archit., **124** (2022) 102405. https://doi.org/10.1016/j.sysarc.2022.102405.

28. Ros S., Ryoo I., Kim S. - DRL-driven intelligent SFC deployment in MEC workload for dynamic IoT networks. Sensors, **25**(14) (2025) 4257. https://doi.org/10.3390/s25144257.
29. Poltronieri F., Stefanelli C., Suri N., Tortonesi M. - Value is king: The mecforge deep reinforcement learning solution for resource management in 5G and beyond. J. Netw. Syst. Manage., **30**(4) (2022) 63. https://doi.org/10.1007/s10922-022-09672-6.
30. Nam T. M., Thanh N. H., Hieu H. T., Manh N. T., Huynh N. V., Tuan H. D. - Joint network embedding and server consolidation for energy–efficient dynamic data center virtualization. Comput. Netw., **125** (2017) 76-89. https://doi.org/10.1016/j.comnet.2017.06.007.
31. Al-Fares M., Loukissas A., Vahdat A. - A scalable, commodity data center network architecture. Comput. Commun. Rev., **38**(4) (2008) 63-74. https://doi.org/10.1145/1402946.1402967.
32. Niranjan Mysore R., Pamboris A., Farrington N., Huang N., Miri P., Radhakrishnan S., Subramanya V., Vahdat A. - Proceedings of the ACM SIGCOMM 2009 conference on data communication, ACM, (2009) 39-50. https://doi.org/10.1145/1592568.1592575.
33. Huong T., Schlosser D., Nam P., Jarschel M., Thanh N., Pries R. - 11th würzburg workshop on IP: Joint ITG and euro-NF workshop visions of future generation networks (EuroView2011), (2011)
34. Thanh N. H., Cuong B. D., Thien T. D., Nam P. N., Thu N. Q., Huong T. T., Nam T. M. - 2013 international conference on advanced technologies for communications (ATC 2013), IEEE, (2013) 312-317. https://doi.org/10.1109/atc.2013.6698128.
35. Mahadevan P., Sharma P., Banerjee S., Ranganathan P. - INFOCOM workshops 2009, IEEE, (2009) 1-6. https://doi.org/10.1109/infcomw.2009.5072138.
36. Abdul-Minaam D. S., Al-Mutairi W. M. E. S., Awad M. A., El-Ashmawi W. H. - An adaptive fitness-dependent optimizer for the one-dimensional bin packing problem. IEEE Access, **8** (2020) 97959-97974. https://doi.org/10.1109/access.2020.2985752.
37. Raj P. H., Ravi Kumar P., Jelciana P., Rajagopalan S. - 2020 4th international conference on intelligent computing and control systems (ICICCS), IEEE, (2020) 1107-1110. https://doi.org/10.1109/iciccs48265.2020.9120929.
38. Hartmanis J. - Computers and intractability: a guide to the theory of NP-completeness (michael R. Garey and david S. Johnson). SIAM Rev., **24**(1) (1982) 90-91. https://doi.org/10.1137/1024022.
39. Heller B., Seetharaman S., Mahadevan P., Yiakoumis Y., Sharma P., Banerjee S., McKeown N. - NSDI '10: 7th USENIX symposium on networked systems design and implementation, (2010) 249-264.
40. Instance - Atlanta Network Problem - SNDlib-Library of test instances for Survivable fixed telecommunication Network Design.
41. Zhou H., Tan L., Zeng Q., Wu C. - Traffic matrix estimation: A neural network approach with extended input and expectation maximization iteration. J. Netw. Comput. Appl., **60** (2016) 220-232. https://doi.org/10.1016/j.jnca.2015.11.013.
42. Waxman B. M. - Routing of multipoint connections. IEEE J. Sel. Areas Commun., **6**(9) (2002) 1617-1622. https://doi.org/10.1109/49.12889.
43. Fischer A., Botero J. F., Beck M. T., de Meer H., Hesselbach X. - Virtual network embedding: A survey. IEEE Commun. Surv. Tutor., **15**(4) (2013) 1888-1906. https://doi.org/10.1109/surv.2013.013013.00155.
44. Zhu Z., Lu H., Li J., Jiang X. - GLOBECOM 2017 - 2017 IEEE global communications conference, IEEE, (2017) 1-6. https://doi.org/10.1109/glocom.2017.8254441.
45. Agarwal S., Cai Q., Agarwal R., Shmoys D., Vahdat A. - 21st USENIX symposium on networked systems design and implementation, (2024) 329-343.