Holu: Power-Aware and Delay-Constrained VNF Placement and Chaining

Article *in* IEEE Transactions on Network and Service Management · January 2021

DOI: 10.1109/TNSM.2021.3055693

5 authors, including:

Amir Varasteh
Technische Universität München
26 PUBLICATIONS **137** CITATIONS
SEE PROFILE

Basavaraj Madiwalar
Technische Universität München
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Amaury Van Bemten
Technische Universität München
17 PUBLICATIONS **196** CITATIONS
SEE PROFILE

Carmen Mas Machuca
Technische Universität München
155 PUBLICATIONS **1,507** CITATIONS
SEE PROFILE

Holu: Power-Aware and Delay-Constrained VNF Placement and Chaining

Amir Varasteh, Basavaraj Madiwalar, Amaury Van Bemten, Wolfgang Kellerer, and Carmen Mas-Machuca
Chair of Communication Networks, Department of Electrical and Computer Engineering,
Technical University of Munich, Germany
Email:{amir.varasteh, basavaraj.madiwalar, amaury.van-bemten, wolfgang.kellerer, cmas}@tum.de

***Abstract*— Service function chains (SFCs) are an ordered set of virtual network functions (VNFs) which can realize a specific network service. Enabled by virtualization technologies, these VNFs are hosted on physical machines (PMs), and interconnected by network switches. In today networks, these resources are usually under-utilized and/or over-provisioned, resulting in power-inefficient deployments. To improve power-efficiency, SFCs should be deployed utilizing the minimum number of PMs and network equipment, which are not concomitant. Considering the existing PM and switch power consumption models and their resource constraints, we formulate the power-aware and delay-constrained joint VNF placement and routing (PD-VPR) problem as an Integer Linear Program (ILP). Due to the NP-completeness of the problem, we propose *Holu*, a fast heuristic framework that efficiently solves the PD-VPR problem in an online manner. Specifically, *Holu* decomposes the PD-VPR into two sub-problems and solve them sequentially: *i)* a *VNF placement* problem that consists of mapping the VNFs to PMs using a centrality-based ranking method, and *ii)* a *routing* problem that efficiently splits the delay budget between consecutive VNFs of the SFC, and finds a Delay-Constrained Least-Cost (DCLC) shortest-path through the selected PMs (hosting VNFs) using the Lagrange Relaxation based Aggregated Cost (LARAC) algorithm. Our simulation results indicate that *Holu* outperforms the state-of-the-art algorithms in terms of total power consumption and acceptance rate by 24.7% and 31%, respectively.**

***Index Terms*—power optimization, power efficiency, energy efficiency, VNF placement, service function chaining**

I. Introduction

In the modern telecommunication world, network providers have been deploying their network services using Virtual Network Functions (VNFs). A service is usually composed of an ordered list of VNFs, deployed on commercial off-the-shelf (COTS) equipment in different parts of the network. This ordered sequence of VNFs is referred to as a Service Function Chain (SFC) [1]. Upon receiving user requests with a specific SFC to traverse, network providers have to find the best server(s) to configure the required VNFs while taking the Network Function Virtualization (NFV) architecture and its available resources into the account. The NFV architecture consists of several interconnected nodes which consist of network switches and hosting Physical Machines (PMs). The former forward the traffic through the network, whereas the latter host VNFs, in form of Virtual Machine (VMs) or containers. Since PMs can host several VNFs, it enables the physical consolidation of networks. To serve the user requests, VNFs must be mapped to the PMs and the traffic should be routed through these VNFs that form the requested SFC.

Despite of its importance, the power-efficiency of the NFV-enabled networks has been only slightly considered. Today, around 7-12% of the total power consumption is required for Internet technologies [2]–[4]. Network providers are challenged to increase their connectivity and services while being sustainable. Bolla *et. al.* [5] have shown the aggressive increase of power consumption in the networks operated by the major Telecom operators worldwide (e.g., AT&T, Verizon). Moreover, the global annual Internet traffic is expected to reach 4,2 ZB, i.e., 4.2×10^{12} GB, in 2022 (with an increase of almost 400% with respect to 2017 [4]). As a result, more and more operators work on the reduction of their carbon footprint and greenhouse gas emission by aiming at decreasing their power consumption [6], [7].

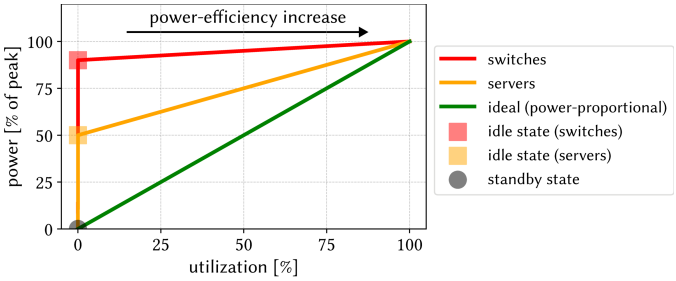


Fig. 1: Power consumption vs. utilization (power-proportionality) comparison for online PMs and network switches with respect to the theoretical ideal power-proportional case. We observe that both PMs and switches are not power-proportional, i.e., they consume a significant amount of power when they are in the idle state, while in the standby state, they consume a negligible amount of power. Therefore, to increase the power-efficiency, the number of online components should be reduced (increasing the resource utilization).

In order to reduce the power consumption, an analysis and modeling of the power consumed by the main NFV components needs to be done, particularly PMs and network equipment. According to Fig. 1, three power states can be considered for PMs and switches: *i)* standby: the server is in low-power (sleep) mode and consumes a negligible amount of power, *ii)* idle: the device is powered-on, however its utilization is almost 0% (no traffic load), *iii)* online: the device is powered-on and its utilization is higher than 0% (processing the traffic load). Starting with the PM power consumption, it has been shown that PMs can consume almost 50% of their maximum power when they are in idle state [8]–[10]. Also, they consume a negligible amount of power when are in standby mode [11] (also referred to as the offline state in some works). Similar to PMs, network resources are usually over-provisioned to support the maximum traffic. However, their utilization rarely reaches the peak network capacity [12], [13]. It has been observed that the online network components such as switch chips and fans consume a significant amount of power even with low workload [12], [14], [15]. Consequently,

idle networking devices are not power-efficient, since an idle network switch can consume up to 90% of the peak power consumption [13].

Accordingly, the relation between power consumption and the device utilization is depicted in Fig. 1 for PMs and switches, assuming a linear power profile [10], [16]. It can be observed that PMs and network devices are not power-proportional, i.e., they do not consume power proportional to their utilization, which differs significantly with the ideal power proportionality case [9], [10], [17]. This difference causes a waste of power consumed by under-utilized devices. Thus, minimizing the number of online PMs and switches can improve the power-proportionality (See Fig. 1) and hence, improving the power-efficiency of the service provider.

In this work, we study the power-aware and delay-constrained joint VNF placement and routing (PD-VPR) problem. Considering the capacitated network resources, the main goal is to minimize the number of online PMs and network switches required to allocate the requested SFCs, while meeting the end-to-end delay (i.e., sum of propagation and VNF processing delay) requirements. We first formulate this problem as an Integer Linear Program (ILP) based on our previous work [18]. The problem formulation has been improved by considering also the VNF processing time, which dynamically changes depending on the traffic. Considering the NP-completeness of the problem, the ILP is not usable for solving real-world problem sizes. Therefore, we propose *Holu*, an efficient heuristic that solves the PD-VPR problem in an online manner. In more detail, *Holu* decomposes the PD-VPR problem into two sub-problems which are solved in sequence: *i)* VNF placement, *ii)* routing. In the first sub-problem, we rank the PMs in the network according to their centrality and the requested VNF types in the SFC. Thereafter, we employ a Delay-Constrained Least-Cost (DCLC) shortest-path algorithm to find the path between the selected VNFs in the previous step using a routing heuristic proposed in our previous work [19]. As an important feature, our routing algorithm is able to efficiently split the end-to-end delay budget between subsequent VNFs in an SFC. This can significantly increase the acceptance rate of requests, even with very strict end-to-end delay requirement.

Therefore, the main contributions of this paper can be summarized as:

- Presenting the PD-VPR problem as an ILP optimization model to *i)* determine the optimal number of the VNFs and their mapping to the PMs, *ii)* allocate user requests to the VNF instances, *iii)* find a path to route the traffic through the allocated VNFs to form the SFC, *iv)* meet the resource capacities and end-to-end delay constraints including the link propagation and traffic-aware VNF processing delays, and *v)* minimize the total power consumption.
- Proposing *Holu*, an online heuristic framework to tackle the PD-VPR problem by dividing it into two sub-problems: VNF placement, and routing.
- Implementation and performance evaluation of the proposed heuristic and comparing with the state-of-the-art CPVNF and BCSP algorithms [18], [20] in terms of total power consumption, acceptance ratio, runtime, etc.

The rest of the paper is organized as follows: The related work is reviewed in Section II. Then, we present the system model and problem definition in Section III followed by the ILP formulation in Section IV. Thereafter, in Section V, we introduce the *Holu* framework. Finally, we present the performance evaluation of the work in Section VI, and conclude the paper in Section VII.

II. Related Work

Although power-efficient VM placement is a well-studied field in the cloud computing environment [37]–[39], VM placement and VNF placement in NFV/SFC paradigm problems differ in many ways. The former case is a problem that focuses on placing/packing a set of VMs on different PMs, while the latter, considers a specific ordered set of VNFs. In addition, the solution should contain routing and path allocation through these ordered VNFs, which makes it a fundamentally difficult problem to solve [40]. Therefore, the VM placement can be considered as a special case of the latter problem. There has been a large body of work that have investigated the VNF placement and routing problem with different objectives, such as minimizing total deployment cost [41]–[44], minimizing total end-to-end delay [45]–[47], minimizing network resources [48], [49] and routing costs [31], [32], maximizing reliability [50]–[53].

Let us summarize the most recent and relevant works addressing the power-aware VNF placement and routing problem [11], [21]–[29], [33]–[36]. These works have been grouped into three categories: *i)* VNF placement: place a set of VNFs in order to meet an objective, *ii)* SFC routing: these works assume the VNFs are already deployed in the network. Thus, they focus on finding the path for the traffic traversing through these VNFs, and *iii)* joint VNF placement and routing: in addition to VNF placement, the traffic path through these VNFs must be determined. In the first category, not concerned with the routing decisions, authors in [21], [22], [26] have tackled the VNF placement problem. In particular, Pham *et. al.*. [21] aimed at deploying VNFs by using as fewer number of online PMs, such that the communication cost between them is optimized. They proposed a fast solution by using a sampling-based Markov approximation method combined with matching theory. Further, Yang *et. al.* [22] studied VNF chain placement in data centers. They provided an algorithm to save power in servers and network switches. A step further was taken by authors in [26] and proposed a dynamic server consolidation approach using VM live migration to achieve power-efficiency by maximizing the number of PMs in standby state.

The second category belongs to the works which tackle the challenges brought by the routing problem. There are several dimensions to consider in this category. There are some works that have focused on the routing problem and have considered the power consumption of ternary content-addressable memory (TCAM) of network switches into the account [54]–[58]. However, in our work, we consider the network power consumption based on the online network switches and the number of active ports and their utilization.

References	Decisions		Power Consumption		Features		
	VNF Placement	Routing (chaining)	Physical Machines	Network	Delay-Constrained	Shared VNF Instance	Online
[19]	✓	✓	✗	✗	✓	✓	✓
[18]	✓	✓	✓	✓	✓	✓	✗
[20]	✓	✓	✓	✗	✓	✓	✗
[21]	✓	✗	✓	✗	✓	✗	✗
[22]	✓	✗	✓	✓	✗	✓	✗
[23]	✓	✓	✓	✓	✗	✓	✗
[24]	✓	✓	✗	✓	✗	✓	✗
[25]	✗	✓	✓	✗	✓	✓	✗
[26]	✓	✗	✓	✗	✗	✓	✓
[11]	✓	✓	✓	✗	✓	✓	✓
[27]	✓	✓	✓	✓	✗	✗	✓
[28]	✓	✓	✓	✗	✓	✓	✗
[29]	✓	✓	✓	✗	✗	✗	✗
[30]	✓	✓	✗	✗	✗	✓	✗
[31], [32]	✗	✓	✗	✗	✗	✗	✗
[33], [34]	✓	✓	✓	✗	✗	✓	✗
[35], [36]	✓	✓	✓	✗	✓	✗	✗
This paper	✓	✓	✓	✓	✓	✓	✓

TABLE I: Comparison of related work and the proposed solution.

Assuming that the VNFs are already placed in the network, some works have focused on finding a path going through the required VNFs (i.e., SFC) considering some constraints, e.g., delay, capacity. For example, the authors in [25] formulated a problem to allocate and schedule traffic flows with deadlines to VNFs while minimizing the total PM power consumption. Recently, the graph layering technique is proposed as an efficient way to find a path through an already placed set of VNFs [19], [30]–[32]. These works transform the network into a layered graph in which each VNF in the SFC is represented by a layer. The user traffic can be routed layer by layer from the top to the bottom layer. For instance, KARIZ, a local search heuristic proposed by [30], finds the path between two layers by solving the minimum cost flow problem. Disregarding the end-to-end delay constraint, the objective of KARIZ is minimizing the network resource costs. As another work, after the graph layering transformation, authors in [31] uses conventional shortest-path algorithms e.g., Dijkstra to calculate the path between the source and destination nodes. To reduce the computational time when using a shortest-path algorithm, Sallam *et. al.* in [32] propose a pruning algorithm to simplify the constructed layered graph. However, similar to [30] and [31], they did not consider the end-to-end delay constraint in their problem. Nevertheless, in our previous work [19], we proposed a heuristic to find a delay-constrained path, passing through a selected set of VNF nodes in a layered graph, achieving near-optimal performance.

Finally, as the third category, some works extend the problem to consider the routing jointly with the VNF placement decision [11], [23], [24], [27]–[29], [33]–[36]. In more detail, their main goal is to place the VNFs on PMs, allocate them to the user requests, and route the traffic through these VNFs. These decisions can be constrained to capacity and/or delay requirements, optimizing PM and/or network power consumption. Specifically, the authors in [35] focused on determining the required number and placement of VNFs to optimize the network utilization and operational costs (in terms of PMs power consumption), without violating service level agreements. They presented an efficient heuristic based on dynamic programming to solve this problem. Furthermore, Jang *et. al.* [29] formulated a multi-objective optimization model which maximizes the acceptance ratio and minimizes the power cost for multiple service chains. After transforming the model into a single-objective mixed integer linear programming (MILP) problem, they proved that the problem is NP-hard and proposed an algorithm based on linear relaxation and rounding to approximate the solution of the MILP in polynomial time. In addition to optimizing the VNF placement and routing, the authors in [11] presented online algorithms to reconfigure the network based on traffic changes.

However, in these three categories, there are a number of missing considerations that are addressed in this paper. For example, in the first and third categories, the necessity of coordination between VNF placement and routing decisions is disregarded. Thus, in this work, we tackle the PD-VPR problem. Moreover, opposed to this work, there are some approaches that do not consider both PM and network power consumption into account, which can increase the OPEX of service providers. Moreover, unlike some reviewed related works, we take the QoS constraints in terms of end-to-end delay into account. Also, some works [11], [26], [33] considered VNF migration and reconfiguration which we do not focus on it in this work. A comparison of different state-of-the-art solutions with respect to this work is summarized in Table I. Further, comprehensive surveys on VNF chain placement are available for interested readers [59]–[62]. In this work, we present *Holu*, an online heuristic framework that presents an efficient VNF placement approach coupled with a fast delay-constrained routing algorithm to solve the PD-VPR problem.

III. System Model and Problem Definition

Before presenting the mathematical modeling of the work, we note that the used notations through this paper and their definition are presented in Table II.

A. Network Model. In this paper, we focus on a Wide Area Network (WAN) scenario, where we represent the network as a unidirectional graph $G = (N, L)$, being N the set of nodes, and L the set of unidirectional links between pair of nodes. Each physical link $(i, j) \in L$ is characterized by the data rate

Sets and Parameters	
$G = (N, L)$	Physical network graph
$\mathcal{R}$	Set of requests
$\bar{G}_r = (\bar{N}_r, \bar{L}_r)$	Virtual network graph of the request r
F	Set of VNF types
$\mathcal{V}_r^s$	Source node of request r
$\mathcal{V}_r^d$	Destination node of request r
$\mathcal{B}_r$	Data rate of request r
$\mathcal{D}_r$	Maximum delay of request r
C_r	SFC of request r
$B_{(i,j)}$	Data rate capacity of physical link (i, j)
$d_{(i,j)}$	Propagation delay of physical link (i, j)
φ_f	Processing delay of VNF type f
Φ_f	Processing capacity of VNF type f
U	Set of VNFs/PMs resource types
$\Delta_{f,u}$	Required resource type $u \in U$ by VNF type f
$C_{i,u}$	Maximum capacity of resource type $u \in U$ in PM i
θ_i^{CPU}	CPU utilization of PM i
$P_{pm}^{idle}, P_{switch}^{idle}$	PM and network switch idle power consumption
P_{pm}^{max}	Maximum PM power consumption
P_{port}	Network switch port power consumption
P_{pm}^T, P_{net}^T	Total PM and network power consumption
Ψ	Large positive integer
$\gamma_n^{r,f}$	The ranking value of PM n with respect to VNF $f \in C_r$
$\alpha_n^{r,f}$	The centrality impact of PM n with respect to VNF $f \in C_r$
$\beta_n^{r,f}$	The power consumption impact of PM n with respect to VNF $f \in C_r$
$c_{(i,j)}$	Routing cost function assigned to link (i, j)
$\mathcal{P}_{(i,j)}$	Routing power impact of using link (i, j)
Q_j	Betweenness centrality of node j
S_r	Set of candidate PMs for request r
Decision Variables	
$x_i \in \{0, 1\}$	=1, if PM i is online
$y_i \in \{0, 1\}$	=1, if switch i is online
$q_{(i,j)} \in \{0, 1\}$	=1, if link (i, j) is online
$w_{(k,l),r}^{(i,j)} \in \{0, 1\}$	=1, if virtual link $(k, l) \in \bar{L}_r$ is mapped to $(i, j) \in L$
$a_{i,f,r} \in \{0, 1\}$	=1, if VNF f for request r is placed in PM i
$n_{i,f} \in \mathbb{N}$	Number of instances of VNF type f on PM i

TABLE II: Notation definition

capacity $B_{(i,j)}$ and the propagation delay $d_{(i,j)}$ (based on the distance of nodes i and j). Every node $n \in N$ consists of a switch and a co-located PM (or a cluster of PMs). The switch can interconnect any input port with any output port as well as forwarding the connection to the PM when required. On the other hand, the PM is characterized by a set of resources $u \in U$, such as CPU, RAM, and storage. Also, each VNF type $f \in F$ has a resource requirement $\Delta_{f,u}$, processing delay φ_f, and processing capacity Φ_f. A deployed instance of VNF f can be shared among user requests as long as its maximum processing capacity φ_f is not surpassed. Otherwise, the new instance of VNF f should be placed either on an online PM with enough available resources, or on a PM which must be powered on. These three alternatives have different impact on the power consumption as introduced in the later sections.

B. User Request: We define a user request $r \in \mathcal{R}$ as following 5-tuple:

$$r = (\mathcal{V}_r^s, \mathcal{V}_r^d, \mathcal{D}_r, \mathcal{B}_r, C_r) \tag{1}$$

where $\mathcal{V}_r^s$ and $\mathcal{V}_r^d$ are the source and the destination nodes, $\mathcal{D}_r$ is the maximum allowed end-to-end delay, $\mathcal{B}_r$ is the requested data rate, and $C_r = \{f_1, f_2, ..., f_{|c_r|}\}$ is the requested SFC, an ordered set of VNFs that the traffic should be routed through.

SFCs can be modeled as a virtual network represented as a graph $\bar{G}_r = (\bar{N}_r, \bar{L}_r)$, where for request r, $\bar{N}_r$ are the set of virtual nodes which are $C_r = \{f_1, f_2, ..., f_{|c_r|}\}$; and $\bar{L}_r$ are the set of links where the virtual link $(i, i+1)$ interconnects f_i with f_{i+1}. In particular, for each request r, the virtual nodes $\bar{N}_r$ (i.e., the VNFs) and their interconnecting links $\bar{L}_r$ should be mapped to the physical network G. Hence, the link $(i, j) \in \bar{L}_r$ between two consecutive VNFs must be assigned to a path connecting physical links in L. Also, the ingress and egress virtual nodes match the physical source and destination nodes in the substrate physical network. In this way, $\bar{G}_r$ has to be mapped over G such that the requirements of request r are met in terms of delay and capacity while the consumed total power is minimized.

C. Power Consumption Models: Let us introduce the power consumption models considered in this problem.

1) Network Power Consumption Model: The network power consumption refers to the amount of power consumed for the transmission, which includes the power consumed by the switches at the network nodes as well as the active interconnecting links. In particular, when a network switch is powered on, it consumes a base power P_{switch}^{idle} Watts which is independent of the traffic load [15], [16], [63], [64]. Similarly, the network ports consume P_{port} Watts if the port is powered on (otherwise, 0 Watts). Therefore, the power of a network switch can be computed as [15], [64]:

$$P_{switch} = \begin{cases} P_{switch}^{idle} + \sum_{i \in ports} P_i & \text{, if it is } \textit{online} \\ 0 & \text{, otherwise} \end{cases} \tag{2}$$

2) PM Power Consumption Model: The most power-consuming factor of a PM has been shown to be the CPU [65]–[68]. Hence, the power consumption model for the PM is based on its CPU utilization, denoted by θ_i^{CPU}. The power consumption of a PM can be calculated as below [69]–[71]:

$$P_{pm} = \begin{cases} P_{pm}^{idle} + (P_{pm}^{max} - P_{pm}^{idle})\ \theta_{pm}^{CPU} & \text{, if it is } \textit{online} \\ 0 & \text{, otherwise} \end{cases} \tag{3}$$

where P_{pm}^{idle} and P_{pm}^{max} is the consumed power when the CPU utilization is 0% and 100%, respectively. Also, θ_{pm}^{CPU} is the CPU utilization of the PM.

D. Problem Statement: Given a network and a set of requested SFCs: *i)* determine the optimal number of the VNF instances to be deployed, *ii)* Mapping of these VNF instances to the PMs, *ii)* allocating user requests to the deployed VNFs, *iv)* find a path to route the user traffic through the allocated VNFs (i.e., forming the SFC), *v)* guarantee the end-to-end delay required by the user requests, which should not exceed the sum of the network propagation delay and the processing delay of VNFs, *vi)* meet the PM and network capacity constraints, and finally *vii)* minimize the total PM and network power consumption.

IV. Optimization Formulation

In this section, we mathematically formulate the PD-VPR problem as an Integer Linear Program (ILP) optimization model. *Total Power Consumption:* As mentioned before, the objective function is to minimize the total power consumption which is defined as the sum of the network and PM consumed power. According to the network power consumption model in Eq. 2, the total network power consumption denoted by P_{net}^T can be calculated as:

$$P_{net}^T = P_{switch}^{idle} \sum_{i \in N} y_i + 2P_{port} \sum_{(i,j) \in L} q_{(i,j)} \tag{4}$$

where $y_i \in \{0, 1\}$ is a variable indicating if switch i is online. Also, $q_{(i,j)} \in \{0, 1\}$ is defined as a variable that is equal to

1 if the physical link between nodes i and j is online. Note that an active physical link $l_{(i,j)} \in L$ (connecting node i to j) requires two online ports (one per source/destination node).

Moreover, according to the PM power consumption model Eq. 3, the total PM power consumption P^T_{pm} can be calculated as:

$$P^T_{pm} = \sum_{i \in N} x_i \left(P^{idle}_{pm} + (P^{max}_{pm} - P^{idle}_{pm})\ \theta^{CPU}_i \right) \tag{5}$$

where x_i is a binary variable indicating if the PM i is powered on. θ^{CPU}_i is the CPU utilization of PM i, which is calculated as the ratio between the all required CPU resources by the hosted VNFs and the available CPU resources on PM i, i.e., $\theta^{CPU}_i = \sum_{f \in F} \Delta_{f,u}/C_{i,u}$ for $u = CPU$. In addition, there are some constraints that need to be met by the ILP model.

Capacity Constraints: There are three capacity constraints. First, the set of resources on a PM is limited, and therefore, these resources cannot be exceeded by the hosted VNFs, i.e.,:

$$\sum_{f \in F} \Delta_{f,u} n_{i,f} \leq C_{i,u},\ \forall i \in N, \forall u \in U, \tag{6}$$

where $\Delta_{f,u}$ is the amount of resource type u used by the placed VNF f, $n_{i,f}$ is an integer variable indicating the number of placed VNFs type f on PM i, and $C_{i,u}$ represents the maximum capacity of resource type u in PM i.

Secondly, each VNF f has a limited processing capacity, which is denoted by Φ_f. Therefore, the sum of the rates of all requests served by function f in the PM i must not exceed the processing capacity of VNF f, i.e.,

$$\sum_{r \in \mathcal{R}} a_{i,f,r} \mathcal{B}_r \leq n_{i,f} \Phi_f,\ \forall i \in N, \forall f \in F, \tag{7}$$

where $a_{i,f,r} \in \{0,1\}$ is a variable which equals to 1 if VNF f of request r is assigned to the PM i, and $\mathcal{B}_r$ is the requested data rate of r. The last set of capacity constraints belong to the physical link capacities. In fact, the sum of the data rate required by all requests served by link (i,j) should not be larger than the capacity of the link (i,j):

$$\sum_{r \in \mathcal{R}} \sum_{(k,l) \in \bar{L}_r} w^{(i,j)}_{(k,l),r} \mathcal{B}_r \leq B_{(i,j)},\ \forall i \in N, \forall j \in N, \tag{8}$$

where $w^{(i,j)}_{(k,l),r} \in \{0,1\}$ is a variable that equals to 1 if the physical link (i,j) is used by the virtual link (k,l) of r.

As introduced in Section III, the nodes of the virtual network graph $\bar{G}_r = (\bar{N}_r, \bar{L}_r)$ demanded by the request r must be embedded in the physical graph $G = (N, L)$. Firstly, the source and destination nodes of $\bar{G}_r$ must be mapped to the physical source and destination nodes in the G. Hence, the source and destination nodes of request r should be mapped to the same node on the substrate physical node:

$$a_{i,f,r} = 1, \text{ if } i = f = \mathcal{V}^s_r,\ \forall r \in \mathcal{R}, \tag{9}$$

$$a_{i,f,r} = 1, \text{ if } i = f = \mathcal{V}^d_r,\ \forall r \in \mathcal{R}. \tag{10}$$

Moreover, the VNFs in SFC C_r must be mapped to a node $n \in N$ that actually host an instance of VNF $f, \forall f \in C_r$:

$$a_{i,f,r} \leq n_{i,f},\ \forall i \in N, \forall f \in F, \forall r \in \mathcal{R}. \tag{11}$$

The flow conservation law is expressed in flow states such that for each network switch $i \in N$, the difference of all outgoing and incoming physical links that are used for the virtual link between virtual nodes k and l, and request r must be equal:

$$\sum_{(i,j) \in L} w^{(i,j)}_{(k,l),r} - \sum_{(j,i) \in L} w^{(j,i)}_{(k,l),r} = a_{i,k,r} - a_{i,l,r},$$
$$\forall i \in N,\ \forall k \in \bar{N}_r,\ \forall l \in \bar{N}_r,\ \forall r \in \mathcal{R}, \tag{12}$$

The total delay of the embedded request r is modeled as the sum of the VNF processing time of each VNF $f \in C_r$, denoted by $\varphi_{f,r}$, and the propagation delay $d_{(i,j)}$ in the physical links. This summation must be limited to the required end-to-end delay $\mathcal{D}_r$ of each request $r \in \mathcal{R}$, thus:

$$\sum_{i \in N} \sum_{f \in F} \varphi_f a_{i,f,r} + \sum_{(i,j) \in L} \sum_{(k,l) \in \bar{L}_r} d_{(i,j)} w^{(i,j)}_{(k,l),r} \leq \mathcal{D}_r,\ \forall r \in \mathcal{R}, \tag{13}$$

where $\varphi_{f,r}$ denotes the processing delay of the VNF f for request r, which is considered proportional to the data rate of request r (i.e., $\varphi_{f,r} = \mathcal{B}_r/\Phi_f$).

Finally, let us introduce three indicator variables to control the operation status (i.e., *online* or *standby*) of PMs, physical links, and network switches):

$$\sum_{r \in \mathcal{R}} \sum_{(k,l) \in \bar{L}_r} w^{(i,j)}_{(k,l),r} \leq q_{(i,j)} \Psi,\ \forall i \in N, \forall j \in N, \tag{14}$$

$$\sum_{j \in N} (q_{(i,j)} + q_{(j,i)}) \leq y_i \Psi,\ \forall i \in N, \tag{15}$$

$$\sum_{f \in F} n_{i,f} \leq x_i \Psi,\ \forall i \in N, \tag{16}$$

where Ψ is a large positive number. Considering the aforementioned definitions and constraints, we can formulate the ILP optimization model as below:

$$\begin{aligned}
&\min\ \left(P^T_{pm} + P^T_{net}\right), && (17)\\
&\text{s.t. constraints } (6) - (16),\\
&\text{vars: } x_i, y_i \in \{0,1\}, \forall i \in N,\\
&\quad q_{(i,j)} \in \{0,1\}, \forall (i,j) \in L,\\
&\quad w^{(i,j)}_{(k,l),r} \in \{0,1\}, \forall (i,j) \in L, \forall (k,l) \in \bar{L}_r, \forall r \in \mathcal{R},\\
&\quad a_{i,f,r} \in \{0,1\}, \forall i \in N, \forall f \in F, \forall r \in \mathcal{R},\\
&\quad n_{i,f} \geq 0, \forall i \in N, \forall f \in F.
\end{aligned}$$

In this formulation, we jointly map a set of virtual nodes $\bar{N}_r, \forall r \in \mathcal{R}$ to PMs together with the mapping of virtual links $\bar{L}_r, \forall r \in \mathcal{R}$ onto paths in the underlying network G connecting the respective servers. Also, this embedding must meet capacity and path length (i.e., delay) constraints. This problem can be reduced to the graph embedding problem which is generally NP-hard and inapproximable [40], [72]. Thus, the presented ILP model (17) does not apply to practical-sized problems as it cannot be solved in a reasonable period. Therefore, in the following sections, we propose an online heuristic algorithm to solve the aforementioned problem in polynomial time.

V. Holu: The Online Heuristic Framework

In this section, we propose an efficient online heuristic framework named *Holu* to solve the PD-VPR problem with much lower and more manageable complexity. Many of the heuristics presented in the state-of-the-art divide the problem based on each VNF, they solve the resulting sub-problems

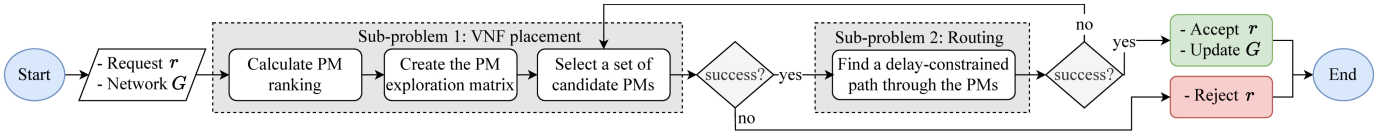


Fig. 2: Flowchart of the *Holu* framework. Given the user request r (Eq. 1) and the network graph G, *Holu* solves the PD-VPR problem by collaboratively performing VNF placement (Sub-problem 1) and routing (Sub-problem 2). In Sub-problem 1, using a node ranking mechanism, the PM exploration matrix is formed. Then, a set of candidate PMs are picked using a PM selection mechanism. Thereafter, in Sub-problem 2, a routing algorithm attempts to realize the requested SFC by routing the traffic from the source node to the destination node through the candidate PMs returned by Sub-problem 1. If the path meets the required delay, the algorithm accepts the request r and registers it in the network. Otherwise, in the next iteration, it attempts to find a new delay-constrained path using the second candidate PMs set. After a limited number of unsuccessful iterations, *Holu* rejects the user request r.

sequentially per VNF [18], [20], [24], [28], [34], [73]–[75]. In more details, considering a user request r with $C_r = \{f_1, f_2, f_3\}$, the referred state-of-the-art papers would consider four sub-problems to determine a solution to the VNF placement and routing problems, one for each VNF and last one for routing from the last VNF to the destination, without having a fallback mechanism after a decision for a VNF is made. They sequentially solve these sub-problems, i.e., determining the VNF placement and the routing first for f_1, then for VNF f_2, and finally f_3 and they build the SFC from the $\mathcal{V}_r^s$ to $\mathcal{V}_r^d$. Considering the previous example, the algorithm tries to place f_3 after placing the f_1, f_2, and find a path from the placed f_2 to the potential f_3 PMs. This might not be feasible anymore, since the constraint budgets (e.g., delay) might have been already consumed by the previous VNFs f_1 and f_2. Therefore, they cannot efficiently distribute the cost/constraint budgets between different sections of a SFC (budget-division problem). This can potentially reduce the quality of found solutions dramatically as well as the acceptance rate. In contrast, *Holu* employs an end-to-end decision-making strategy to solve the PD-VPR problem. This strategy consists on placing all the VNFs at once, and then find the best route, both considering the power consumption. In this way, we can solve the budget-division problem which exists in the state-of-the-art sequential solutions. Moreover, a fallback mechanism improves the quality of solution in an iterative way.

To do so, we divide the PD-VPR in two sub-problems, regardless of the length of the requested SFC: *i)* Sub-problem 1: VNF placement to minimize P_{pm}^T, and *ii)* Sub-problem 2: routing aiming at minimizing P_{net}^T while meeting the delay constraints. As depicted in the flowchart in Fig. 2, having a request r in hand, the VNF placement sub-problem finds a set of PMs to host the VNFs $f \in C_r$. Particularly, *Holu* uses a ranking mechanism to assign a score to the candidate PMs for hosting the VNFs requested in the C_r. Thereafter, it builds a matrix (referred as *exploration matrix*) to represent the candidate PMs that can host VNFs $f \in C_r$ and sorts them based on the calculated ranking values. According to this sorting, a potential VNF placement solution (the PMs to host VNFs $f \in F$) is selected and passed as an input to Sub-problem 2, which attempts to find a routing through these PMs, considering the delay constraint $\mathcal{D}_r$. To do so, we employ our previous work [19] to find a delay-constrained path to route the traffic through the candidate PMs in a power-aware manner. If the routing is not successful (no path is found with respect to delay/capacity constraints), another set of candidate PMs is selected by Sub-problem 1 and the routing is recomputed. Note that since it is not straightforward to select a *good* set of PMs at once (as it would imply solving the whole problem at once), there is an iteration in order to gradually improve the solution. This loop is repeated either until a successful routing is found, which leads to accepting the request r, or a stopping condition (e.g., VNF placement returns no result), in which case the request r is rejected. An illustrative example is presented in Fig. 3.

As a result, by identifying the VNF hosting PMs for the SFC (Sub-problem 1) and finding a path passing through these PMs (Sub-problem 2), we are able to distribute the cost (power) and the delay budget between different segments of the SFC. In below, the details of these two sub-problems and their respective solutions are presented.

A. Sub-problem 1: VNF Placement

As mentioned before, the VNF placement process identifies to map each of VNFs in the requested C_r on a PM in network with the goal of minimizing the first element of the objective function, P_{pm}^T (See the ILP model (17)). In order to improve power-efficiency, we focus on increasing the reusability of VNFs by sharing them between more SFCs, thus, increasing their utilization and minimizing the number of online PMs (i.e., improving power-proportionality, See Fig. 1). Therefore, it is critical for the VNF placement process to identify the PMs that maximize the VNF reusability factor. We achieve this goal by introducing a PM ranking strategy, as explained below.

1) PM Ranking Mechanism: Given the request r, we determine a set of PMs that are the *best* candidates for hosting each VNF $f \in C_r$. To do so, we employ a mechanism that assigns a ranking value to each PM, according to the requested SFC C_r. We define the score of PM $n \in N$ to host VNF $f \in C_r$ as $\gamma_n^{r,f}$:

$$\gamma_n^{r,f} = \alpha_n^{r,f} + \beta_n^{r,f}, f \in C_r, n \in N, \tag{18}$$

In following, we define $\alpha_n^{r,f}$ and $\beta_n^{r,f}$ in detail.

Node Centrality Impact ($\alpha_n^{r,f}$): Given a network topology, a wide spectrum of centrality metrics can be defined in order to find the most *critical/important* node(s) in a network [76], [77]. Metrics relying only on the node degree have been shown to not improve the resource reusability as they incur higher network power consumption or too long delays [77]. Hence, in this work, three centrality metrics relying on the length of shortest-paths between all node pairs are used to determine the value of $\alpha_n^{r,f}$: Betweenness (BC), Closeness (CC), and Katz Centrality (KC) [76], [77]. Using one of the three chosen metrics, we calculate the centrality value for all the nodes

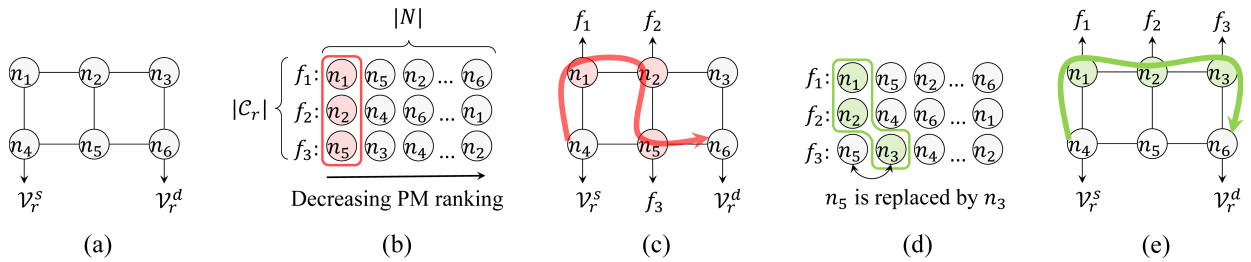


Fig. 3: A high-level example that illustrates the steps that the *Holu* framework takes to find a solution for a request r. The request r is defined with source node $\mathcal{V}_r^s = n_4$, destination node $\mathcal{V}_r^s = n_6$, and the requested SFC as $C_r = \{f_1, f_2, f_3\}$ (in this example we do not consider/show the data rate $\mathcal{B}_r$). (a) Physical network graph G with $|N|$=6 nodes, where the request r must be embedded. (b) Considering the graph G and the request r, the PM exploration matrix $\mathbb{M}^r$ is formed in this sub-figure. There are $|N|$=6 columns and $|C_r|$=3 rows. Each row contains the candidate PMs for the respective VNF, sorted by their ranking value $\gamma_n^{r,f}$ (e.g., considering the first row, $\gamma_{n_1}^{r,f_1} > \gamma_{n_5}^{r,f_1} > \gamma_{n_2}^{r,f_1} > \ldots > \gamma_{n_6}^{r,f_1}$). It also shows the first set of candidate PMs that are chosen to host the VNFs in C_r is $\mathcal{S}_r = \{n_1, n_2, n_5\}$. (c) As determined by the PM exploration matrix, the PMs in the first column n_1, n_2, n_5 are the candidate PMs to host f_1, f_2, and f_3, respectively. In the first iteration, the delay-constrained shortest-path algorithm (Sub-problem 2) should find a path meeting all the routing constraints. In this example, there is not such a path (e.g., the path $(n_4 - n_1 - n_2 - n_5 - n_6)$ does not meet the delay constraint). (d) After the unsuccessful path search in the first iteration, the PM selection mechanism selects the next best PM candidate set from the matrix $\mathbb{M}^r$ by replacing n_5 by n_3 for hosting f_3, having $\mathcal{S}_r = \{n_1, n_2, n_3\}$. (e) In the second iteration, the routing algorithm successfully finds a path $(n_4 - n_1 - n_2 - n_3 - n_6)$ which meets the delay and capacity requirements. Finally, this path as well as the VNF configuration on PMs n_1, n_2, and n_3 are registered in the network.

$n \in N$ and then normalize the obtained values between 0 and 1 to obtain $\alpha_n^{r,f} \forall n \in N$.

Power Consumption Impact ($\beta_n^{r,f}$): This metric aims at prioritizing PMs which causes the minimum increase in the power consumption to process a new SFC request r (either by creating a new VNF instance or using an existing one). The $\beta_n^{r,f}$ can take three different values based on the state of a PM $n \in N$ with respect to VNF $f \in C_r$ (the power consumption caused by hosting f is increasing from first to the third case):

- $\beta_n^{r,f} = 1$: The PM n is *online* and already hosts an instance of VNF f with enough available capacity to process the data rate $\mathcal{B}_r$.
- $\beta_n^{r,f} = 0.1$: The PM n is *online* and has sufficient resources to launch a new instance of VNF f.
- $\beta_n^{r,f} = 0$: The PM n is *standby* and must be turned on to launch a new instance of VNF f.

The $\beta_n^{r,f}$ values can be tuned by the operator e.g., based on the power-proportionality of PMs and/or the size (impact) of specific VNF types that is being used.

After determining the values of $\alpha_n^{r,f}$ and $\beta_n^{r,f}$, upon receiving a SFC request r, for each $f \in C_r$, the values of $\gamma_n^{r,f}$ can be computed. Note that the weights of $\alpha_n^{r,f}$ and $\beta_n^{r,f}$ metrics can be changed according to the operator preferences/priorities.

2) PM Exploration Matrix: As mentioned before, given a request r, we would prefer to place the VNF $f \in C_r$ on a PM with the highest $\gamma_n^{r,f}$ value. Here, the challenge is how to select this set of candidate PMs, which is denoted by $\mathcal{S}_r$, where $|\mathcal{S}_r| = |C_r|$. To do so, we define a PM exploration matrix $\mathbb{M}^r_{|C_r| \times |N|}$ which represents the search space of the candidate PMs: each row represents a set of candidate PMs to host a VNF $f \in C_r$, which are sorted in decreasing order according to their $\gamma_n^{r,f}$ value. Potentially, since all the PMs in the network are candidate nodes, there are N columns in the matrix $\mathbb{M}^r$. Fig. 3b shows an example of a PM exploration matrix $\mathbb{M}^r$ for a user request r. In this exemplary matrix, there are three (i.e., $|C_r|$) rows, one per VNF in C_r, and four (i.e., $|N|$) columns.

3) PM Selection Mechanism: The VNF placement problem consists in selecting one PM from each row in the matrix $\mathbb{M}^r$. Since the PMs in each row are sorted based on their $\gamma_n^{r,f}$ value, the first (and the *best*) choice of PM candidates is the first PM from each row in $\mathbb{M}^r$. For example, in Fig. 3b, the best (first) candidate PMs to host VNFs $\{f_1, f_2, f_3\}$ are supposed to be n_1, n_2, and n_5, respectively.

Having the set of candidate PMs $\mathcal{S}_r$, a path from $\mathcal{V}_r^s$ to $\mathcal{V}_r^d$ passing through the sequence of PM nodes $n \in \mathcal{S}_r$ should be calculated. However, such a path might not be found because of the routing algorithm incompleteness or capacity/delay constraint violations. If this happens, we retry to find a path for an updated set of candidate PMs. In our approach, we select a PM from the current $\mathcal{S}_r$ and replace it by the next PM with the highest ranking value from the respective row in matrix $\mathbb{M}^r$. For example, in Fig. 3d, after no solution is found for the first candidate PM set n_1, n_2, n_5 (Fig. 3b), the PM n_5 is chosen to be replaced by the next PM n_3 to form the new candidate PM set n_1, n_2, n_3.

The question here is which PM from the current candidate PMs set $\mathcal{S}_r$ should be replaced. Since different candidate PM sets can lead to different solutions with different qualities, it is important to determine how to select the next set of candidate PMs, which is expected to reduce the power consumption while allowing a path through them guaranteeing the constraints. Having the PM exploration matrix in hand, we propose three different candidate PM selection mechanisms to determine the next candidate PM set:

- *The Highest Node Ranking (HNR):* The first approach uses the $\gamma_n^{r,f}$ metric to select and replace a PM from the candidate PMs set $\mathcal{S}_r$. HNR selects the candidate PM with the highest $\gamma_n^{r,f}$ value among the current set of PMs. Then the selected candidate PM is replaced by the next PM in the corresponding row of $\mathbb{M}^r$. Thereby, this approach relaxes the power savings to meet the end-to-end delay and in-turn, increases the acceptance ratio.
- *The Highest Remaining Capacity (HRC):* HRC selects the PM with the highest remaining capacity and replaces it by the next PM from its corresponding row in matrix $\mathbb{M}^r$. The idea is to use up PMs fewer remaining compute resources over PM with more compute resources so that the power-efficiency is higher (See Fig. 1).
- *The Largest Sub-path Delay (LSD):* The third approach uses the sub-path delay metric to identify and prune a

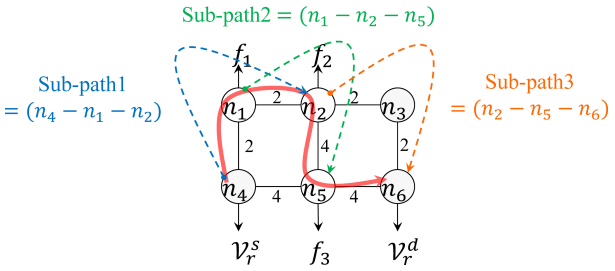


Fig. 4: Example of the LSD PM selection method. Considering the failed solution from the example in Fig. 3c, a total $|C_r|$ sub-paths are formed, each corresponding to a VNF $f \in C_r$. Moreover, each sub-path has a certain total delay based on the propagation delay of each of its links. In this example, the delays of sub-paths 1, 2, and 3 are assumed to be 4, 6, and 8 ms respectively. As the third sub-path (corresponding to VNF f_3) has the highest delay among the sup-paths, the LSD removes the PM at n_5 hosting f_3, and looks for another PM, according to the matrix $\mathbb{M}^r$ (See Fig. 3d).

PM from the candidate PM set. To do so, we first divide the shortest-path from $\mathcal{V}_r^s$ to $\mathcal{V}_r^d$ through candidate PMs of $\mathcal{S}_r$ into $|C_r|$ sub-paths: $\mathcal{V}_r^s$ to f_2 (here f_1 is the intermeddiate VNF), f_1 to f_3, an$f_{|C_r|-1}$ to $\mathcal{V}_r^d$. Each sub-path contains a single PM hosting a VNF as an intermediate PM. Then, for each sub-path, we calculate the shortest-path from the start to the end node of the sub-path with the delay as the cost function. Afterward, we prune the PM which is an intermediate PM of the sub-path with the largest sub-path delay from the current PM set. As a result, the candidate PM with a high contribution to the end-to-end delay of the original path is replaced with the next PM from the respective row in the matrix $\mathbb{M}^r$. Fig. 4 shows an example where a path and its delay from $\mathcal{V}_r^s$ to $\mathcal{V}_r^d$ through f_1, f_2, and f_3 VNFs is shown.

The heuristic continues to try different candidate PM sets $\mathcal{S}_r$ until a solution meeting the delay and capacity constraints is found or a stopping condition is reached (See Fig 2). This condition can be chosen by the operator based on the network size and the service provider's decision time constraints.

B. Sub-problem 2: Routing

In this section, we present the solution to the routing sub-problem which aims at minimizing the second element of the objective function, P_{net}^T (Eq. 17). At this stage, the user request r and an ordered set of candidate PMs $\mathcal{S}_r$ are given by Sub-problem 1. The problem now, is to find a path from source $\mathcal{V}_r^s$ to destination $\mathcal{V}_r^d$, that traverses the candidate PMs in $\mathcal{S}_r$, and that satisfies the delay constraint $\mathcal{D}_r$. Given a cost function that models the power consumption of a path, the problem is to find a Delay-Constrained Least-Cost (DCLC) path from a given source src to destination dst, which can be modeled as:

$$z^*(src, dst) = \min_{\mathfrak{p} \in \mathbb{P}_{src,dst}} \sum_{(i,j)\in\mathfrak{p}} c_{(i,j)} \quad \text{s.t.} \sum_{(i,j)\in\mathfrak{p}} d_{(i,j)} \leq \mathcal{D}_r \tag{19}$$

where $\mathbb{P}_{src,dst}$ is the set of paths from node src to dst, and $c_{(i,j)}$ and $d_{(i,j)}$ are the cost (i.e., network power consumption) and delay functions of link (i, j) respectively.

In the following, we show how to solve Sub-problem 2 by using the Lagrange Relaxation based Aggregated Cost (LARAC) algorithm [78] and an extension of it, LARAC-SN [19]. Let us first shortly describe these two algorithms.

1) LARAC Algorithm: The DCLC problem is NP-hard [79]. Accordingly, numerous heuristics have been proposed to quickly find close to optimal solutions [80]. The LARAC algorithm finds a DCLC path in a graph by running several times the Dijkstra shortest-path algorithm. It is commonly considered as one of the best performing heuristics for the DCLC problem [19], [80]. The algorithm is based on the Lagrange relaxation, a mathematical optimization technique for solving constrained optimization problems. Using the Lagrange relaxation principle, a heuristic solution $z^R(s,t)$ to the DCLC problem (Eq. 19) can be found by optimally solving [80]:

$$z^R(\mathcal{V}_r^s, \mathcal{V}_r^d) = \max_{\lambda\in\mathbb{R}^+} \left(\min_{\mathfrak{p}\in\mathbb{P}_{\mathcal{V}_r^s,\mathcal{V}_r^d}} \sum_{(i,j)\in\mathfrak{p}} c_{(i,j)} + \lambda\left(\sum_{(i,j)\in\mathfrak{p}} d_{(i,j)} - \mathcal{D}_r \right) \right). \tag{20}$$

Solving this maximization problem requires to solve several times the inner minimization problem (called *relaxed problem*) by varying the λ parameter. Interestingly, for this problem, the relaxed problem corresponds to a shortest-path problem with a modified cost function $c^\lambda = c_{(i,j)} + \lambda \times d_{(i,j)}$ [78]. As a result, LARAC finds a heuristic solution to the DCLC problem by successively running Dijkstra with the modified cost function c^λ and by adapting the λ at each iteration to converge to the maximum of Eq. 21.

LARAC starts by finding the least-delay and least-cost paths in the network from a given source node src to destination dst. If the least-delay path does not exist, it returns null since there no path exists that meets the delay constraint. If the least-cost path ($\lambda = 0$) does not violate the delay constraint, the path is returned as the solution (which is the optimal solution for sub-problem 2). Otherwise, if the delay of the least-cost path is higher than $\mathcal{D}_r$, it performs subsequent Dijkstra runs with the modified cost function c^λ. At each iteration, λ is adapted in order to give more or less importance to the cost and/or the delay. More details on the rationale behind the convergence strategy are available in the original reference [78] and the survey [80]. The algorithm is *complete*, i.e., it always finds a solution if it exists, but not optimal. However, it was shown that its optimality gap is quite low [80] while keeping a relatively low runtime. Thus, it becomes a perfect candidate to be used as part of the solution to Sub-problem 2 that we solve repeatedly to obtain a solution to our original problem.

2) LARAC-SN Algorithm: LARAC-SN builds on top of LARAC to force the traversal of an ordered set of nodes [19]. To do so, the algorithm adapts the underlying Dijkstra procedure used by LARAC. Instead of running Dijkstra from the source to the destination node, LARAC-SN splits the Dijkstra run from the source node to the first VNF, from the second VNF to the next one, and so on until the destination. By doing so, the algorithm ensures that LARAC only considers paths that traverse the set of candidates PMs and hence that the final path satisfies this constraint. As we ensure that p_c, p_d and p always traverse the VNFs of C_r (i.e., the PM nodes in

$\mathcal{S}_r$), the path returned by LARAC-SN will also traverse the VNFs. Splitting the underlying Dijsktra run rather than the LARAC run removes the need for splitting the delay constraint between the different VNFs. That is automatically handled by the LARAC procedure by adapting the weights of the aggregated cost function, which can significantly increase the acceptance rate of the requests. Inheriting the properties of LARAC, LARAC-SN is complete and close-to-optimal [19].

3) Routing Metrics: Delay, Cost, and Capacity: The first metric is the delay. It is a global constraint metric. The delay of a link corresponds to its propagation delay (the VNF processing delay can be pre-computed and subtracted from the delay constraint budget $\mathcal{D}_r$), thus, its value for a link is static for a given routing request. The second metric is the cost which is a global optimization metric. Our cost function essentially considers the power consumption impact of taking a given path. The cost of a link (i, j) is defined as

$$c_{(i,j)} = \hat{\mathcal{P}}_{(i,j)} + \hat{\mathcal{Q}}_j, \tag{21}$$

where the $\hat{\cdot}$ notation represents the fact that both values are normalized to a value between 0 and 1. As it can be seen, $c_{(i,j)}$ consists of two cost elements. The first one is the impact of power consumption $\hat{\mathcal{P}}_{(i,j)}$ for using a given link (i, j) and it is computed as

$$\hat{\mathcal{P}}_{(i,j)} = \frac{1}{2} P^{idle}_{switch} \times (2 - y_i - y_j) + 2P_{port} \times (1 - q_{(i,j)}), \tag{22}$$

where y_i and $q_{(i,j)}$ are binary variables reporting whether the switch i or link (i, j) are already powered on, respectively. In fact, variables y and q allow to penalize the usage of a switch or link that is not yet used (i.e., that is in the *standby* state). The second term forming the $c_{(i,j)}$ is the centrality metric $\hat{\mathcal{Q}}_j$ of the destination node j of the link. The reason for adding this term is that the power consumption of links can often be equal, as only six different values are possible (based on the y and q variables). As a result, the routing algorithm will often face equal-cost paths. Instead of letting the algorithm pseudo-randomly choose among these paths, the centrality term acts as a tie breaker to direct the algorithm towards more central nodes that tend to be used more often and hence reduce the power consumption of subsequent requests. We note that we use the BC value of the node j for the $\hat{\mathcal{Q}}_j$. Computing the value of our cost function $c_{(i,j)}$ requires to know all the links previously visited by the routing algorithm. Indeed, since the state $s_{(k,l)}$ of a link depends on whether it is used already (i.e., whether we already visited it), knowing all the previously visited links is necessary to compute $\hat{\mathcal{P}}_{(i,j)}$ and hence $c_{(i,j)}$. In [81], it is shown that such a metric as global optimization metric increases the optimality gap of an algorithm. Yet, the algorithm stays complete.

The third and last metric we use is the capacity of links. It is a local constraint metric. A link can only be used if it still has sufficient capacity to host the new flow. Similar to the cost function, checking if the new flow can be accommodated by the remaining capacity at a link requires to know whether we visited this link already. In traditional routing, it is not the case, because we know the remaining capacity of all links in advance and we do not route into loops. However, when routing through a set of VNFs, we can potentially route through loops. Unfortunately, such a metric as a local constraint metric makes the routing algorithm *incomplete* [81]. Recovering the completeness of algorithms in such situation leads to intractable runtime [81]. In fact, this is one of the reasons for which we have to iterate back and forth between sub-problems 1 and 2. Indeed, once $\mathcal{S}_r$ has been selected in Sub-problem 1, the algorithm used in Sub-problem 2 cannot guarantee (because it is not complete) to find a solution for the $\mathcal{S}_r$, even if it exists. We iterate between the two sub-problems until the routing algorithm finds a solution. In the evaluation section, we show that the incompleteness of the algorithm is not very high, and hence solutions can be found in relatively few iterations and hence, short runtime.

Type	SFC Set	Data Rate	Delay	Share
Web	NAT-FW-TM-WOC-IDPS	U[0.6-1] Mbps	500 ms	18.2%
VoIP	NAT-FW-TM-FW-NAT	U[0.384-0.64] Mbps	100 ms	11.8%
Streaming	NAT-FW-TM-VOC-IDPS	U[24-40] Mbps	100 ms	69.9%
Gaming	NAT-FW-VOC-WOC-IDPS	U[0.24-0.5] Mbps	60 ms	0.1%

TABLE III: Overview of service types and their SFC set, data rate, end-to-end delay, and share of the total requests. We have considered request aggregation by considering a range of data rate to mimic the traffic coming from multiple users at the same time from a single network node. (NAT: network address translator, FW: Firewall, TM: traffic monitor, WOC: WAN optimization controller, IDPS: intrusion detection prevention system, VOC: video optimization controller)

VI. Performance Evaluation

A. Simulation Setup

This section provides details about the network topologies and the simulation parameters considered for the evaluation. The results presented in this section correspond to: NobelEU [82], and Internet2 [83]. Similar results have been obtained for Geant topology [84] but not included due to space limitations. Each network node is equipped with a switch and a PM for hosting VNFs. The PM resources are characterized by the number of CPU cores, which has been set to 16. The PM power consumption is computed in accordance to Eq. 3. The idle and maximum power consumption of the PM is chosen as P^{pm}_{idle}=299 Watts and P^{pm}_{max}=521 Watts, respectively [85]. On the other hand, the network switch power consumption consists of a static hardware power and network interface power (See Eq. 2). The static hardware power of the switch P^{idle}_{switch} is set to 315 Watts [86]. The network interface power consumption of switch port varies based on the data rate configuration, which is modeled to operate at three different configurations: 100 Mbps, 1 Gbps, or 10 Gbps with 26 Watts, 30 Watts, and 55 Watts of power consumption, respectively [64], [87]. The physical link data rate capacity is set to a randomly generated value between 6 to 10 Gbps. To generate the user requests, the source and destination pairs are generated randomly, such that $\mathcal{V}^s_r \neq \mathcal{V}^d_r$. We have considered commonly used service types: video streaming, web service, voice-over-IP (VoIP), and online gaming [18], [24]. As it is presented in Table III, each of these service types requires a specific SFC, data rate, maximum end-to-end delay, and the share of the service type in the set of user requests. Additionally, the VNFs forming the SFCs have considered with different resource requirements and processing capacities as listed in Table IV [35], [88].

VNF Type	Num. CPU Cores	Processing Capacity
NAT	2	500 Mbps
FW	8	400 Mbps
TM	1	200 Mbps
VOC	2	580 Mbps
WOC	2	300 Mbps
IDPS	8	600 Mbps

TABLE IV: The number of CPU cores and their processing capacity (maximum throughput) of each VNF type.

To choose the best centrality metric and PM selection mechanism for *Holu*, we have extensively evaluated them in terms of power consumption and acceptance rate. Based on the results (omitted due to the lack of space), we have selected the best-performing settings to evaluate *Holu*. In particular, for each node $n \in N$, we use CC as centrality metric $\alpha_n^{r,f}$ and the LSD as the PM candidate exploration strategy (supportive evaluations are presented in the results section). The proposed ILP optimization model (17) has been implemented using Gurobi solver [89] in Python. Besides, *Holu* framework and the state-of-the-art approaches are simulated using Java. The simulations have been performed on a machine equipped with Intel Core i7-6700HQ @2.60 GHz, 16 GB of RAM, running Ubuntu 18.04.

B. Algorithms to Compare

We compare the proposed ILP model and the *Holu* heuristic with two state-of-the-art algorithms: *i)* A state-of-the-art approach named CPVNF [20], and *ii)* The BCSP algorithm, which was presented in our previous work [18] (referred as the BC-Based algorithm). These two algorithms are briefly explained in below:

1) CPVNF: The CPVNF algorithm [20] addresses VNF placement and routing problem with the objective of minimizing the number of online PMs and communication costs, while meeting end-to-end delay and resource capacities. CPVNF act in a sequential way (solving VNF per VNF). To select *high-quality* PMs in the network to host each VNF, it uses a ranking algorithm based on the Google PageRank method. PageRank is a variant of the Eigen Vector centrality measure [90] and assigns ranks to a web page based on both the quality and quantity of the web pages linked to it. Comparatively, the authors of CPVNF algorithms assign ranks to PMs based remaining capacity of PM itself, aggregated remaining bandwidth of its outgoing links, and rank of its directly connected nodes [20]. Additionally, nodes are assigned personalized PageRank to bias towards PMs with VNF instance under question. For every request, the CPVNF algorithm uses the computed rank metric along with a delay function to select the VNF hosting PMs. CPVNF uses the *k*-shortest-path algorithm to compute the routing at each stage, and selects the path with the lowest delay value. If the end-to-end delay is less than the $\mathcal{D}_r$, the user request is accepted in the network. Otherwise, a new set of VNF hosting PMs is selected by increasing the importance of aggregate outgoing link bandwidth over the remaining CPU resource in node rank computation. The search continues until the iteration reaches the predefined search limit.

2) BCSP: This algorithm represents the BC-based algorithm in our previous work [18]. This algorithm performs the VNF placement and routing sequentially for each VNF in the requested SFC. BCSP works based on the BC centrality metric for VNF placement and shortest-path for routing. It first calculates the BC metric for all the nodes in the graph. Then, for each request r, it calculates the shortest-path between source $\mathcal{V}_r^s$ and destination $\mathcal{V}_r^d$ nodes using Dijkstra with delay as the cost function. Then, the BCSP algorithm tries to place the required VNFs of the $\mathcal{C}_r$ on the nodes with higher BC value on the shortest-path in a greedy manner. According to the BC definition, a higher BC value means a higher number of shortest-paths that are passing through a node n and hence, a higher probability of VNF re-usability for future requests, which can lead to power-efficiency.

C. Simulation Results

This subsection summarizes the performance evaluation on two topologies. Before presenting the results, we note that due to the scalability issues, we were able to run the ILP model for up to 25 user requests (more than 48 hours of computation time for a single problem instance with 50 requests).

1) Power Consumption: Let us start comparing the total, PM, and network power consumption per accepted user request for the optimal case and the three heuristics for different number of user requests. Fig. 5 shows the mean power consumption per accepted request value. Among all the heuristics, it can be observed that the *Holu* has the lowest total power consumption per request values in all the topologies. In more detail, *Holu* is performing on average 19.3% worst than the optimal solution, and outperforming CPVNF and the BCSP algorithms by 19% and 24.7%, respectively. However, for higher number of requests, the network gets saturated and hence, the gap between the different heuristics decreases.

All the three heuristics utilize centrality measures to determine the PMs to place VNFs. However, it is evident from the results that the effectiveness of all the centrality measures is not the same. The performance of the CPVNF algorithm decreases when the network contains densely connected regions away from the graph center (e.g., more connected at the edges of the network). That is, unlike the ranking method of *Holu*, the PMs with higher ranks have more distance from other high rank nodes, making it difficult to meet the required SFC delay constraint. For instance, Internet2 is from this group of graphs, which the power consumption of CPVNF is even worse than the BCSP algorithm, because BCSP is always using the shortest-path (cost is considered as delay) between source and destination (See Fig. 5a). On the bright side, the *Holu* algorithm first ranks the PMs by using the CC and their power consumption impacts (i.e., α_n^r and β_n^r, respectively) The CC value of a node depends on its position in the network for all other nodes. As a result, *Holu* can identify the PMs that increase the reusability of resources, leading to lower power consumption. The average number of online PMs per request have been summarized in Table V for 25 user requests. The lower number of online PMs per user request for *Holu* compared to the other heuristics reflects the effectiveness of the proposed PM ranking and selection approach. Concerning BCSP, the disadvantage when solving the VNF placement

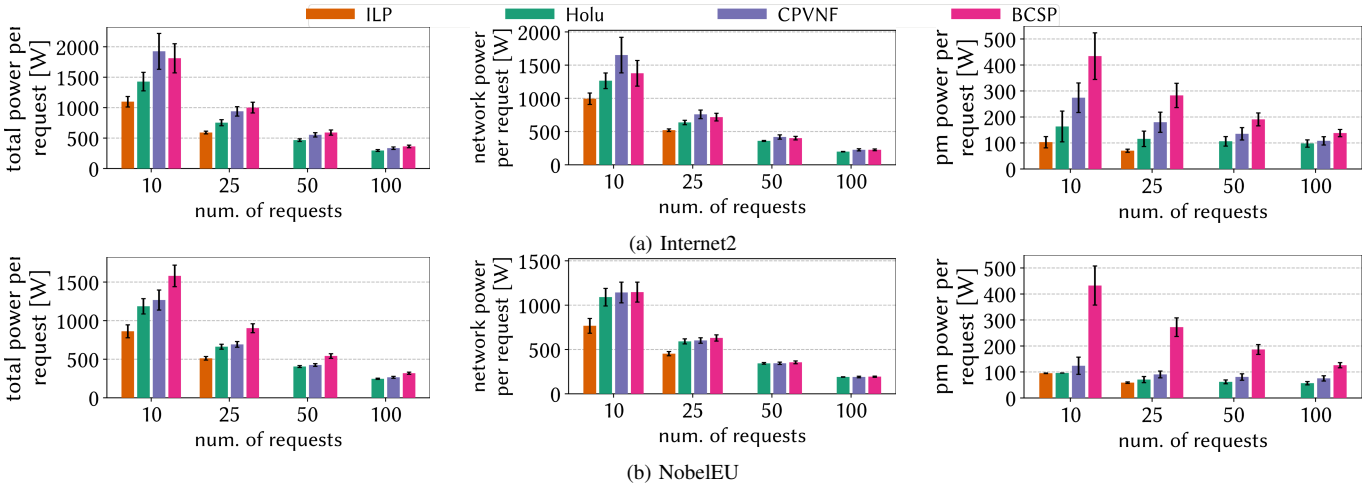


Fig. 5: Comparison of total, network, and PM power consumption per user request for (a) Internet2 and (b) NobelEU topologies. It can be seen that the proposed *Holu* heuristic outperforms the achieved total power consumption achieved by CPVNF and BCSP algorithms up to 19% and 24.7% on average.

Topology	ILP	*Holu*	CPVNF	BCSP
Internet2	0.14	0.25	0.38	0.62
NobelEU	0.12	0.14	0.19	0.6

TABLE V: Comparison of the average number of online PMs per request for 25 user requests. *Holu* utilizes a fewer number of online PMs to serve the user requests compared to state-of-the-art approaches, thus improving the power-proportionality of the network.

problem, is that the search space is limited to the nodes along the shortest-path between request's source and destination. As a result, the resource reusability decreases, which leads to an increase of online PMs in both topologies.

2) Acceptance Rate: In our next study, we run *Holu*, CPVNF, and BCSP for up to 500 user requests to compare the acceptance rate for the two topologies. As we were able to run the ILP for a maximum of 25 requests, the feasibility of the problem for the larger inputs is unknown to us and requires further investigations. Also, we note that the ILP model either accepts all the requests or nothing, therefore it is omitted from the plots. As it can be seen in Fig. 6, *Holu* can accept 31% and 20% more requests for Internet2, and 16.3% and 14.1% for NobelEU topology compared to the CPVNF and BCSP, respectively. An important parameter in successfully accepting a request is meeting its delay requirement. The results in Fig. 6 indicate that opposed to the other algorithms, by employing LARAC, *Holu* is able to divide the delay budget between the VNFs in the chain. In this way, it can increase its delay-awareness which leads to a significant improve in acceptance ratio.

The CPVNF algorithm sorts the PMs of the network based on their ranking (i.e., according to the PageRank metric), ignoring the delay role. Thus, it is possible that the node with a high ranking is located far from the request's source and/or destination. As a result, the delay requirement of the user request may not be guaranteed and hence, the request is rejected. Besides, the BCSP algorithm uses the BC metric for the PMs along the shortest-path to select the candidate nodes for VNFs. Therefore, a node with a high BC value may be located closer to the user request's destination node. As a result, the BCSP algorithm needs to make a loop along the shortest-path to realize the SFC. Under such circumstances,

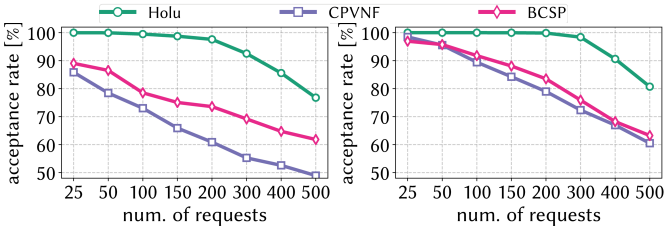
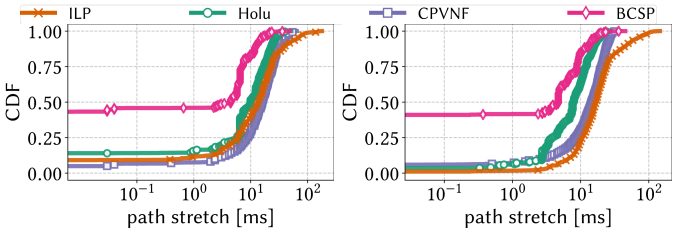


Fig. 6: Comparison of the acceptance rate of the algorithms for different number of user requests. This figure shows that on average, *Holu* achieves 31% and 20% higher acceptance rate for Internet2, and 16.3% and 14.1% for NobelEU topology compared to the CPVNF and BCSP, respectively.

the actual SFC path computed by the BCSP is longer than the shortest-path which can lead to missing the delay requirement, and rejecting the request. Also, we can see that the acceptance rate for BCSP is higher than CPVNF for the Internet2 topology and almost similar for the NobelEU. The Internet2 topology contains high ranked nodes (according to PageRank metric) far away from each other and edges with large lengths (i.e., higher delay). As a result, the CPVNF algorithm performs poorly as compared to the BCSP algorithm in meeting the delay requirement, hence reducing the user request acceptance rate. Therefore, it can be seen that the coordination of an efficient VNF placement and a power and delay-aware routing can play an important role in the acceptance delay. Thereby, *Holu* is able to balance the power and delay requirement, achieving a higher acceptance rate.

3) Path Stretch: The path stretch metric represents the end-to-end delay difference between the path computed by each algorithm and the actual shortest-path (based uniquely on delay) between the source and destination request. This metric is important because the lower path stretch can be translated to lower network resource consumption, i.e., lower OPEX costs. Fig. 7 shows the CDF of path stretch values for ILP, *Holu*, CPVNF, and BCSP approaches for 25 user requests. We omit to show these results for other numbers of user requests, since their behavior is similar. It can be seen that in both topologies, the proposed *Holu* heuristic achieves the minimum-maximum path stretch compared to all the approaches, followed by CPVNF, BCSP, and the ILP. The



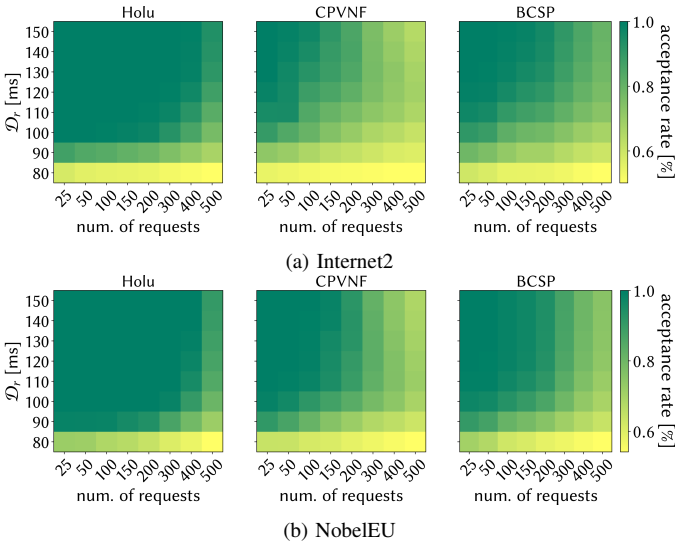
(a) Internet2 (b) NobelEU

Fig. 7: Path stretch comparison in case of 25 user requests. It can be seen that by outperforming the CPVNF and ILP, the average path stretch of *Holu* stands higher than the BCSP algorithm (which places the VNFs on the shortest-path). In both topologies, ILP produces the largest path stretch, since it focuses on maximizing the reusing of resources, which can lead to taking longer paths. Also, due to the properties of Internet2 topology, the maximum path stretch value is higher for Internet2 compared to NobelEU.

reason is that the delay-aware routing algorithm can efficiently balance the importance of the power vs. the delay metrics. That is, the ILP shows an extreme high path stretch, due to focusing on power minimization, while *Holu* causes a reasonable path stretch, while having a lower power consumption compared to the other heuristics. Also, the plots indicate that the ILP solution has many instances with a high path stretch values. The reason is that the ILP, in an effort to reduce the total power, makes the requests to traverse through VNF hosting nodes which are not necessarily close to the source and/or destination of the request.

4) Delay Tolerance: This study aims to compare the acceptance rate of *Holu*, CPVNF, and BCSP algorithms for user requests with varying delay requirements. The goal is to explore how different algorithms can handle user requests with tight or loose delay requirements. For this purpose, we run the simulation for different sets of user requests, ranging from 25 to 500. Each of these user requests are generated with random source and destination pairs, 4 Mbps data rate, and a specific SFC type (See Table III). Moreover, we select the delay values of user requests from 80 to 150 ms in steps of 10 ms. The goal is to check which algorithm is more sensitive to the delay variations in terms of acceptance rate. Fig. 8 shows the heatmap of the acceptance rate for a different number of user requests with varying delay requirements for both topologies. These plots exhibit that *Holu* has a generally higher acceptance rate for all the delay ranges and for different number of requests. This is because *Holu* puts emphasis on the delay parameter during the selection of VNF hosting nodes and routing process. Thereby, it succeeds in achieving a greater acceptance rate even under strict delay conditions. The CPVNF has the second-best performance after *Holu* and BCSP has the worst performance. The reason is CPVNF applies a more sophisticated ranking method and also explores a larger solution space compared to the BCSP approach.

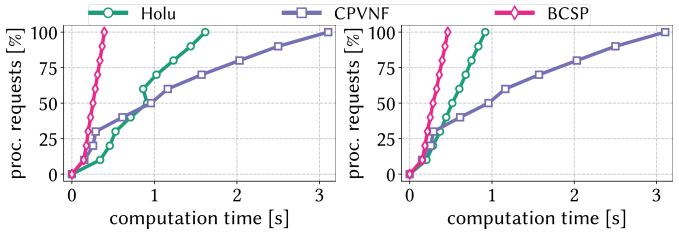
5) Runtime: In this subsection, we compare the runtime of the algorithms. To do this, we record the computation time of each algorithm at the end of processing of every 10% of the user requests. Due to space limitations, we only show the comparison of the runtime of different approaches for 100 user requests. We note that we omitted the ILP results from the plot, because it runtime is in order of hours. Fig. 9 shows that *Holu* can do the admission and solve the PD-VPR for



(a) Internet2

(b) NobelEU

Fig. 8: Comparison of acceptance rate of different algorithms with respect to the number of user requests and the value of the delay constraint. This figure indicates that compared to others, *Holu* accepts more requests, even the ones with tight delay requirements. Also, due to the higher graph connectivity, more requests can be embedded into the NobelEU topology, compared to Internet2.



(a) Internet2 (b) NobelEU

Fig. 9: The comparison of the runtime efficiency for *Holu*, CPVNF, and BCSP for processing 100 user requests.

a given user request in 10 ms for NobelEU, and 15 ms for Internet2 network topology. We note that the implementation of *Holu* can be optimized e.g., by using more efficient data structures to achieve even lower runtime for practical use-cases. Besides, the results in Fig. 9 indicate that the BCSP algorithm is the fastest approach. The main reason behind it is the lower number of times that requires to run Dijkstra (only once per user request compared with several times for the other heuristics). For instance, *Holu* needs to compute the shortest-path several times iterative for certain requests (depending on the PM selection mechanism).

On the other hand, the CPVNF and *Holu* approaches require to calculate PM ranking values before processing a user request resulting in higher runtime. After processing some of the initial 40-50% of the user requests, the CPVNF algorithm requires more time to process the remaining user requests compared to the other approaches. This is because it saturates the the high ranked PMs for serving the initial requests. As a result, it requires more iterations to find the VNF hosting PMs which meets the delay requirements for remaining user requests. Moreover, unlike our approach, the CPVNF needs to recompute the node ranks at every iteration, which leads to a higher runtime.

VII. Conclusions and Future Work

Motivating by low power-proportionality of both Physical Machines (PMs) and network switches, we formulate the power-aware and delay-constrained joint VNF placement and service function chain (SFC) routing problem (PD-VPR). Since the resource are usually under-utilized and/or over-provisioned, minimizing the number of online devices can lead to improved resource utilization and more power-proportionality. To achieve this goal, we first proposed an Integer Linear Program (ILP) model. Due to its scalability issues, we presented an online heuristic framework named *Holu* which tackles the problem by breaking it into two sub-problems which are solved sequentially: *i) Virtual Network Function (VNF) placement*, and *ii)* routing. The VNF placement suggests a set of candidate PMs to host the requested SFC. It uses a PM ranking mechanism considering the centrality of the PM and the power consumption impact by hosting a VNF. We show that centrality is an important metric to consider for PM ranking, since it can improve the resource utilization and power-efficiency in long-term. Also, to find a path traversing through the set of suggested candidate PMs (returned by the VNF placement sub-problem), we employed a complete algorithm based on Lagrange Relaxation based Aggregated Cost (LARAC) heuristic to solve a Delay-Constrained Least-Cost (DCLC) shortest-path routing problem. We showed that our algorithm is able to efficiently split the delay budget between two consecutive VNF in an SFC, leading to a high acceptance ratio compared to state-of-the-art algorithms.

Our simulation results indicated that compared to existing approaches, the end-to-end SFC placement and routing approach, employed by *Holu*, is able to determine the VNF to PM mapping and also accurately split the delay budget among the routing paths between the consecutive VNFs in the SFC, improving its power consumption and acceptance rate up to 24.7% and 31%, respectively.

Future work. We believe there are many interesting open challenges to be tackled, especially in improving the coordination of the VNF placement and routing sub-problems. For example, one can introduce a distance metric from the source and/or destination to the PM ranking calculation (See Eq. 18). Using that, similar to the A* algorithm [91], this information can be used to prune the candidate PMs that are far from source and/or destinations. This can improve the completeness of *Holu*, hence, its performance, especially in terms of acceptance rate. In addition, PM selection methods can be improved to generate *better* candidate PM sets. For example, one can replace multiple PMs based on more complex PM selection methods at each iteration to improve the search space exploration. These improvements can also lead to reducing the average number of VNF placement iterations, thus, reducing the runtime. Finally, a resource (PM and network) consolidation module can be added to *Holu*, which can be triggered periodically or in case of resource over/under-load. For example, in case of resource overload, it can use live Virtual Machine (VM) migration to migrate the workload away from the *hot* spots, thus, potentially improving the acceptance rate. Also, in the case of under-load, it can move the load away from an under-utilized region to further improve the total power-efficiency.

Acknowledgment

The authors would like to thank Nemanja Deric and the anonymous reviewers for their fruitful comments. This work is funded by Germany Federal Ministry of Education and Research under project OptiCON (grant IDs #16KIS0989K and #16KIS0991).

References

[1] J. Halpern and C. Pignataro, "RFC 7665: Service Function Chaining (SFC) Architecture," *IETF Datatracker*, 2015.

[2] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, pp. 338–347.

[3] P. Corcoran and A. Andrae, "Emerging trends in electricity consumption for consumer ICT," *National University of Ireland, Galway, Connacht, Ireland, Tech. Rep*, 2013.

[4] IEA, "Digitalization and energy," in *Int. Energy Agency*, 11 2017.

[5] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, D. Suino, C. Vassilakis, and A. Zafeiropoulos, "Cutting the energy bills of internet service providers and telecoms through power management: An impact analysis," *Computer Networks*, vol. 56, no. 10, pp. 2320–2342, 2012.

[6] M. F. Tuysuz, Z. K. Ankarali, and D. Gözüpek, "A survey on energy efficiency in software defined networks," *Computer Networks*, vol. 113, pp. 188–204, 2017.

[7] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2015.

[8] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," vol. 37, no. 1. ACM New York, NY, USA, 2009, pp. 157–168.

[9] C. Jiang, Y. Wang, D. Ou, B. Luo, and W. Shi, "Energy proportional servers: Where are we in 2016?" in *2017 IEEE 37th Int. Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1649–1660.

[10] S. Malla and K. Christensen, "The effect of server energy proportionality on data center power oversubscription," *Future Generation Computer Systems*, vol. 104, pp. 119–130, 2020.

[11] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2018.

[12] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks." in *Nsdi*, vol. 10, 2010, pp. 249–264.

[13] P. Mahadevan, S. Banerjee, P. Sharma, A. Shah, and P. Ranganathan, "On energy efficiency for enterprise and data center networks," *IEEE Communications Magazine*, vol. 49, no. 8, pp. 94–100, 2011.

[14] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "Energy aware network operations," in *IEEE INFOCOM Workshops 2009*. IEEE, 2009, pp. 1–6.

[15] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, "Carpo: Correlation-aware power optimization in data center networks," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 1125–1133.

[16] J. C. Cardona Restrepo, C. Gruber, and C. Mas-Machuca, "Energy profile aware routing," in *First Int. Workshop on Green Communications IEEE Int. Conference on Communications (ICC)*, Jun 2009.

[17] Z. Abbasi, G. Varsamopoulos, and S. K. Gupta, "Tacoma: Server and workload management in internet data centers considering cooling-computing power trade-off and energy proportionality," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 9, no. 2, pp. 1–37, 2012.

[18] A. Varasteh, M. De Andrade, C. Mas-Machuca, L. Wosinska, and W. Kellerer, "Power-aware virtual network function placement and routing using an abstraction technique," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[19] A. Van Bemten, J. W. Guck, P. Vizarreta, C. Mas-Machuca, and W. Kellerer, "LARAC-SN and mole in the hole: Enabling routing through service function chains," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 298–302.

[20] M. Dieye, S. Ahvar, J. Sahoo, E. Ahvar, R. Glitho, H. Elbiaze, and N. Crespi, "CPVNF: Cost-efficient proactive VNF placement and chaining for value-added services in content delivery networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 774–786, 2018.

[21] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient VNF placement for service chaining: Joint sampling and matching approach," *IEEE Transactions on Services Computing*, 2017.

[22] K. Yang, H. Zhang, and P. Hong, "Energy-aware service function placement for service function chaining in data centers," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[23] B. Farkiani, B. Bakhshi, and S. A. Mirhassani, "A fast near-optimal approach for energy-aware SFC deployment," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1360–1373, 2019.

[24] N. Huin, A. Tomassilli, F. Giroire, and B. Jaumard, "Energy-efficient service function chain provisioning," *Journal of Optical Communications and Networking*, vol. 10, no. 3, pp. 114–124, 2018.

[25] N. El Khoury, S. Ayoubi, and C. Assi, "Energy-aware placement and scheduling of network traffic flows with deadlines on virtual network functions," in *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*. IEEE, 2016, pp. 89–94.

[26] V. Eramo, M. Ammar, and F. G. Lavacca, "Migration energy aware reconfigurations of virtual network function instances in NFV architectures," *IEEE Access*, vol. 5, pp. 4927–4938, 2017.

[27] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Generation Computer Systems*, vol. 91, pp. 347–360, 2019.

[28] B. Kar, E. H.-K. Wu, and Y.-D. Lin, "Energy cost optimization in dynamic placement of virtualized network function chains," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 372–386, 2017.

[29] I. Jang, D. Suh, S. Pack, and G. Dán, "Joint optimization of service function placement and flow distribution for service function chaining," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2532–2541, 2017.

[30] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2479–2489, 2017.

[31] A. Dwaraki and T. Wolf, "Adaptive service-chain routing for virtual network functions in software-defined networks," in *Proceedings of the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*, 2016, pp. 32–37.

[32] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest path and maximum flow problems under service function chaining constraints," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2132–2140.

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

[34] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "Energy efficient algorithm for VNF placement and chaining," in *2017 17th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2017, pp. 579–588.

[35] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 725–739, 2016.

[36] S. Kim, S. Park, Y. Kim, S. Kim, and K. Lee, "VNF-EQ: dynamic placement of virtual network functions for energy efficiency and QoS guarantee in NFV," *Cluster Computing*, vol. 20, no. 3, pp. 2107–2117, 2017.

[37] F. L. Pires and B. Barán, "A virtual machine placement taxonomy," in *2015 15th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2015, pp. 159–168.

[38] A. Varasteh and M. Goudarzi, "Server consolidation techniques in virtualized data centers: A survey," *IEEE Systems Journal*, vol. 11, no. 2, pp. 772–783, 2015.

[39] A. Varasteh, F. Tashtarian, and M. Goudarzi, "On reliability-aware server consolidation in cloud datacenters," in *2017 16th Int. Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 2017, pp. 95–101.

[40] M. Rost and S. Schmid, "Virtual network embedding approximations: Leveraging randomized rounding," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 2071–2084, 2019.

[41] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1346–1354.

[42] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[43] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably efficient algorithms for placement of service function chains with ordering constraints," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 774–782.

[44] F. Tashtarian, M. F. Zhani, B. Fatemipour, and D. Yazdani, "CoDeC: a cost-effective and delay-aware SFC deployment," *IEEE Transactions on Network and Service Management*, 2019.

[45] R. Gouareb, V. Friderikos, and A.-H. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2346–2357, 2018.

[46] Y. Zhang, B. Anwer, V. Gopalakrishnan, B. Han, J. Reich, A. Shaikh, and Z.-L. Zhang, "Parabox: Exploiting parallelism for virtual network functions in service chaining," in *Proceedings of the Symposium on SDN Research*, 2017, pp. 143–149.

[47] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal VNF placement at the network edge," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 693–701.

[48] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "A scalable approach for service chain mapping with multiple SC instances in a wide-area network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 529–541, 2018.

[49] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "Joint virtual network function placement and routing of traffic in operator networks," *UC Davis, Davis, CA, USA, Tech. Rep*, 2015.

[50] M. Karimzadeh-Farshbafan, V. Shah-Mansour, and D. Niyato, "A dynamic reliability-aware service placement for network function virtualization (NFV)," *IEEE Journal on Selected Areas in Communications*, 2019.

[51] D. Li, P. Hong, K. Xue, and J. Pei, "Availability aware VNF deployment in datacenter through shared redundancy and multi-tenancy," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1651–1664, 2019.

[52] L. Qu, C. Assi, M. Khabbaz, and Y. Ye, "Reliability-aware service function chaining with function decomposition and multipath routing," *IEEE Transactions on Network and Service Management*, 2019.

[53] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan, "Reliability-aware virtualized network function services provisioning in mobile edge computing," *IEEE Transactions on Mobile Computing*, 2019.

[54] K. Kannan and S. Banerjee, "Compact tcam: Flow entry compaction in tcam for power aware sdn," in *International conference on distributed computing and networking*. Springer, 2013, pp. 439–444.

[55] H. Huang, S. Guo, J. Wu, and J. Li, "Green datapath for tcam-based software-defined networks," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 194–201, 2016.

[56] P. T. Congdon, P. Mohapatra, M. Farrens, and V. Akella, "Simultaneously reducing latency and power consumption in openflow switches," *IEEE/ACM Transactions On Networking*, vol. 22, no. 3, pp. 1007–1020, 2013.

[57] S. S. Vegesna, A. C. Nara, and N. M. Sk, "A novel rule mapping on tcam for power efficient packet classification," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 5, pp. 1–23, 2019.

[58] Y.-H. Chen, T.-L. Chin, C.-Y. Huang, S.-H. Shen, and R.-Y. Huang, "Time efficient energy-aware routing in software defined networks," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. IEEE, 2018, pp. 1–7.

[59] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on Service Function Chaining," *Journal of Network and Computer Applications*, vol. 75, pp. 138–155, 2016.

[60] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[61] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2018.

[62] W. Chen, X. Yin, Z. Wang, and X. Shi, "Placement and routing optimization problem for service function chain: State of art and future opportunities," *arXiv preprint arXiv:1910.02613*, 2019.

[63] D. Boru *et al.*, "Energy-efficient data replication in cloud computing datacenters," *Cluster Computing*, vol. 18, no. 1, pp. 385–402, 2015.

[64] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," in *Int. Conference on Research in Networking*. Springer, 2009, pp. 795–808.

[65] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH computer architecture news*, vol. 35, no. 2, pp. 13–23, 2007.

[66] J. Chase *et al.*, "Managing energy and server resources in hosting centers," *ACM SIGOPS operating systems review*, vol. 35, no. 5, pp. 103–116, 2001.

[67] G. Chen *et al.*, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services." in *NSDI*, vol. 8, 2008, pp. 337–350.

[68] A. Berl *et al.*, "Energy-efficient cloud computing," *The computer journal*, vol. 53, no. 7, pp. 1045–1051, 2010.

[69] Y. C. Lee *et al.*, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, pp. 268–280, 2012.

[70] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[71] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," in *Advances in computers*. Elsevier, 2011, vol. 82, pp. 47–111.

[72] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 2018, pp. 1–9.

[73] W. Fang, M. Zeng, X. Liu, W. Lu, and Z. Zhu, "Joint spectrum and it resource allocation for efficient vnf service chaining in inter-datacenter elastic optical networks," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1539–1542, 2016.

[74] P. Vizarreta, M. Condoluci, C. Mas-Machuca, T. Mahmoodi, and W. Kellerer, "QoS-driven function placement reducing expenditures in NFV deployments," in *IEEE Int. Conference on Communications*, 2017.

[75] S. Ahvar, H. P. Phyu, S. M. Buddhacharya, E. Ahvar, N. Crespi, and R. Glitho, "Ccvp: Cost-efficient centrality-based vnf placement and chaining algorithm for network service provisioning," in *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2017, pp. 1–9.

[76] S. P. Borgatti and M. G. Everett, "A graph-theoretic perspective on centrality," *Social networks*, vol. 28, no. 4, pp. 466–484, 2006.

[77] F. A. Rodrigues, "Network centrality: an introduction," in *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*. Springer, 2019, pp. 177–196.

[78] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 2. IEEE, 2001, pp. 859–868.

[79] M. R. Garey and D. S. Johnson, *Computers and intractability*. freeman San Francisco, 1979, vol. 174.

[80] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for sdn: A comprehensive survey and performance evaluation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 388–415, 2017.

[81] A. Van Bemten, J. W. Guck, C. Mas-Machuca, and W. Kellerer, "Routing metrics depending on previous edges: The mn taxonomy and its corresponding solutions," in *2018 IEEE Int. Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[82] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[83] R. Summerhill, "The new internet2 network," in *6th GLIF Meeting*, 2006.

[84] "Geant topology map," *https://www.geant.org/Networks/Pan-European_network/Pages/GEANT_topology_map.aspx*, 2018.

[85] "SpecPOWER," *https://www.spec.org/power_ssj2008/results/power_ssj2008.l* Online accessed 2020.

[86] W. Van Heddeghem, F. Idzikowski, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, "Power consumption modeling in optical multilayer networks," *Photonic Network Communications*, vol. 24, no. 2, pp. 86–102, 2012.

[87] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE, 2008, pp. 457–465.

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

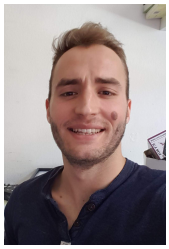
[89] G. Optimization, "Gurobi Optimizer 5.0," *Gurobi: http://www. gurobi. com*, 2013.

[90] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," 1998.

[91] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

Amir Varasteh received his M.Sc. degree in Information Technology from Sharif University of Technology, Tehran, Iran in 2015. He joined the Chair of Communication Networks at Technical University of Munich (TUM) as a PhD candidate in February 2017. His current research focuses on resource management and reconfiguration algorithms in dynamic programmable networks.

Basavaraj Madiwalar received his B.E. degree in Telecommunication Engineering from Visvesvaraya Technological University, India, in 2013. He graduated with a Masters degree in Communications Engineering from Technical University of Munich (TUM), Germany in 2019. He currently works as a full-time Software Engineer in IP/Optical Networks (ION) division at Nokia, Stuttgart, Germany.



Amaury Van Bemten received the B.Sc. degree in engineering, in June 2013, and the M.Sc. degree in computer science and engineering, in June 2015, both from the University of Liège, Liège, Belgium. He joined the Chair of Communication Networks at the Technical University of Munich (TUM) as a PhD candidate in September 2015. His current research is focused on data plane isolation techniques for predictable latency in software-defined networks.

Wolfgang Kellerer (M′96 – SM′11) is a Full Professor with the Technical University of Munich (TUM), heading the Chair of Communication Networks at the Department of Electrical and Computer Engineering. Before, he was for over ten years with NTT DOCOMO's European Research Laboratories. He received his Dr.-Ing. degree (Ph.D.) and his Dipl.-Ing. degree (Master) from TUM, in 1995 and 2002, respectively. His research resulted in over 200 publications and 35 granted patents. He currently serves as an associate editor for IEEE Transactions on Network and Service Management and on the Editorial Board of the IEEE Communications Surveys and Tutorials. He is a member of ACM and the VDE ITG.

Carmen Mas-Machuca (M'96–SM'12) is a Privat Dozent/Adjunct Teaching Professor at the Chair of Communication Networks, Technical University of Munich (TUM), Germany. She received her Dipl.-Ing. degree (Master) from Universitat Politècnica de Catalunya (UPC, Spain) in 1995 and her Dr.-Ing. degree (Ph.D.) from École Polytechnique Fédérale de Lausanne (EPFL, Switzerland) in 2000. Dr. Mas-Machuca has published more than 150 peer reviewed papers. Her main research interests are in the area of techno-economic studies, network planning and resilience, SDN/NFV optimization problems and next generation converged access networks. She is currently NoF'21General Co-Chair, ONDM'21 TPC Co-Chair, and IEEE TNSM Associate Co-Editor of the special issue on "Design and Management of Reliable Communication Networks".