Network Service Embedding Across Multiple Resource Dimensions

Angelos Pentelas, George Papathanail, Ioakeim Fotoglou, Panagiotis Papadimitriou
Department of Applied Informatics, University of Macedonia, Greece
{*apentelas, papathanail, fotoglou, papadimitriou*}*@uom.edu.gr*

***Abstract*—Network Function Virtualization (NFV) poses the need for efficient embeddings of network services, usually defined in the form of service graphs, associated with resource and bandwidth demands. As the scope of NFV has been expanded in order to meet the requirements of virtualized cellular networks and emerging 5G services, the diversity of resource demands across dimensions, such as CPU, memory, and storage, increased. This requirement exacerbates the already challenging problem of network service embedding (NSE), rendering most existing NSE methods inefficient, as they commonly account for a single resource dimension (*i.e.,* typically, the CPU).**

In this context, we investigate methods for NSE optimization across multiple resource dimensions. To this end, we study a range of multi-dimensional mapping efficiency metrics and assess their suitability for heuristic and exact NSE methods. Utilizing the most suitable and efficient metrics, we propose two heuristics and a mixed integer linear program (MILP) for optimized multi-dimensional NSE. In addition, we devise a virtual network function (VNF) bundling scheme that generates (resource-wise) balanced VNF bundles in order to augment VNF placement. Our evaluation results indicate notable resource efficiency gains of the proposed heuristics compared to a single-dimensional counterpart, as well as a minor degree of sub-optimality in relation to our proposed MILP. We further demonstrate how the bundling scheme affects the embedding efficiency, when coupled with our most efficient heuristic. Our study also uncovers interesting insights and potential implications from the utilization of multi-dimensional metrics within NSE methods.

***Index Terms*—Network Function Virtualization, Orchestration, Mathematical optimization, Network service embedding.**

I. Introduction

Middleboxes, such as firewalls, proxies, network address translators (NAT), intrusion detection systems (IDS), and redundancy elimination devices, comprise an indispensable part of the network infrastructure [1], [2]. The various limitations of middleboxes in terms of resource efficiency (*i.e.,* middleboxes are typically provisioned for peak loads, meaning that they can remain under-utilized for long periods) and flexibility (middleboxes are specialized devices that cannot be re-purposed for other functionality) have sparked interest in software-based solutions, which are promoted by the Network Function Virtualization (NFV) paradigm [3], [4]. NFV is being materialized in enterprise datacenters (DCs), (micro-)DCs operated by telcos, and pubic cloud infrastructures, enabling the deployment of virtualized network functions (VNF) on commodity servers at unprecedented flexibility and lower cost.

The deployment and management of VNFs is far from straightforward, as these entail significant challenges in terms of resource allocation, service chaining, and elasticity, which are commonly studied within the context of NFV orchestration [5]. Network service embedding (NSE) comprises one of the most critical problems for NFV infrastructures, since efficient NSE methods can meet certain resource allocation policies and objectives, leading to increased revenue and potential reduction of operation costs (*e.g.,* by VNF consolidation) [6]. The problem of NSE consists in the mapping of service chains (*i.e.,* ordered sequences of VNFs) onto the NFV infrastructure, while ensuring resource efficiency and correctness. NSE can be optimized with specific objectives in mind, which can be tailored to the provider's policy (*e.g.,* footprint minimization, inter-rack traffic minimization) or the client's needs (*e.g.,* expenditure minimization). In this respect, a range of heuristic and exact methods have been proposed, which are applicable either within a single NFV infrastructure [7], [8] or across multiple domains [6], [9].

One limitation of most proposed NSE methods is that they account for a single resource dimension, which is highly restrictive especially in the case of nodes (*i.e.,* VNFs). In practice, VNF resource demands need to be expressed across multiple dimensions, such as CPU, memory, and storage. Furthermore, given the expansion of the scope of NFV, from packet processing functions to virtualized cellular network elements (*e.g.,* Serving Gateway, Mobility Management Entity) [8], [10], and specialized service elements (*e.g.,* for location-based services) [11], we anticipate a wide diversity in the requirements among the resource dimensions. Taking into account only the CPU demand (which is the common practice of the majority of NSE methods) for memory- or storage-intensive VNFs can lead to a significant resource wastage and, consequently, loss of revenue for the provider. The implications of single-dimensional NSE methods on resource efficiency can be more severe in resource-constrained NFV environments, such as edge clouds.

In this work, we tackle the NSE problem under a more pragmatic scope, by designing and evaluating methods for optimized NSE across multiple resource dimensions. To this end, we study the efficiency of relevant multi-dimensional mapping metrics, utilized for virtual machine placement. We further assess the suitability of these metrics for their utilization in multi-dimensional heuristic and exact service embedding methods. More precisely, we employ the metrics *L2 norm-based greedy* and the *cosine similarity* [12], [13], [14] into heuristic algorithms, whereas the *Manhattan distance* is deemed more appropriate for our proposed mixed integer linear program (MILP), because of its linearity. An additional metric, *i.e., resource skewness* [15], is used to quantify the balance in the resource consumption among the multiple dimensions.

Along these lines, the main goals of our study are the following: (i) quantify the gains of multi-dimensional NSE methods compared to counterparts that account only for CPU demands, and (ii) investigate to what extent a balanced resource utilization (across resource dimensions) can guarantee high resource efficiency. Our extensive evaluation results indicate notable resource efficiency gains from the use of multi-dimensional NSE methods. With respect to the second goal, we uncover that the integration of a multi-dimensional mapping metric into a heuristic is by no means straightforward; instead, they need to be very carefully coupled in order to ensure high efficiency.

This paper extends our previous work [16], by expanding the scope of NSE methods from two dimensions to an arbitrary number of resource dimensions[1], thereby, enhancing the applicability and potential of the proposed methods. In this version, we also devise and evaluate a VNF bundling mechanism that seeks to generate VNF bundles out of individual VNFs with unbalanced resource demands. More precisely, the main contributions of this paper are the following:

- We investigate the suitability of multi-dimensional mapping efficiency metrics for NSE across multiple resource dimensions.
- We propose both heuristic and exact methods for the embedding of network services with demands across multiple resource dimensions.
- We assess the efficiency of multi-dimensional metrics for NSE, and further quantify the resource efficiency gains compared to a NSE method with a single resource dimension.
- We perform a comparison between our most prominent heuristic and a MILP, in terms of NSE optimality and solver run-time.
- We propose a VNF bundling scheme and assess its efficiency by coupling it with our most efficient heuristic.
- We shed light on challenging aspects and potential implications from the utilization of multi-dimensional metrics within NSE heuristics and exact methods.

The remainder of the paper is organized as follows. In Section II, we elaborate on the NSE and the multi-dimensional resource allocation problems. In Section III, we initially present service and substrate network models required for our formulations; subsequently, we study four relevant multi-dimensional mapping efficiency metrics and comment upon their suitability for heuristic and exact methods. In Section IV, we introduce a MILP formulation, a heuristic for multi-dimensional embedding, as well as our VNF bundling method. In Section V, we quantify the efficiency of the multi-dimensional metrics in comparison with a baseline heuristic that accounts only for CPU demands. In addition, we compare the most efficient heuristic variant and our proposed MILP, and assess the impact of the VNF bundling scheme on the embedding efficiency. Section VI provides an extensive discussion of related work. In Section VII, we discuss critical aspects of multi-dimensional NSE and highlight our conclusions.

[1]CPU, memory, storage, and I/O are regarded as the most common resource dimensions.

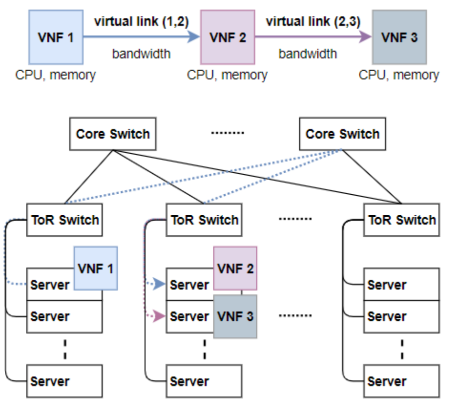

Fig. 1: Network service embedding onto a two-layer datacenter network topology.

II. Problem Description

In this section, we elaborate on the NSE and the multi-dimensional virtual machine (VM) allocation problems. In particular, we aim at pointing out how methods from the latter can be incorporated into the former and, further explain the integration of multi-dimensional allocation principles into a heuristic algorithm that we later present and evaluate.

A. Network Service Embedding

NSE exhibits certain similarities with the Virtual Network Embedding (VNE) problem. VNE consists the main resource allocation challenge in network virtualization [17], since both virtual nodes and virtual links have to be mapped to the respective substrate components, subject to capacity constraints. Nonetheless, VNE was initially decoupled into two sub-problems; namely, the *virtual node mapping* and the *virtual link mapping* [18]. This decomposition, though, may lead to sub-optimal embeddings. This problem was rectified by subsequent VNE methods, which coordinate node and link assignment (*e.g.*, [19]).

NSE differentiates itself from VNE in that the virtual networks are expressed as undirected graphs composed of vertices and edges, whereas network services (NSes) are specified as directed graphs (more precisely, as directed acyclic graphs) [20], [21]. According to [6], a service chain represents the exact sequence of VNFs traversed by one or multiple flows. That is, NSes are defined as directed graphs composed of vertices and edges, which represent VNFs and virtual links (for VNF communication), respectively, as shown on the top of Fig. 1. The essence of the problem lies in the efficient assignment of both VNFs and virtual links to the respective counterparts of the substrate topology, while adhering to capacity constraints. Schematically, according to Fig. 1, each VNF must be placed at exactly one server, while each virtual link must be mapped onto a sequence of physical links that,

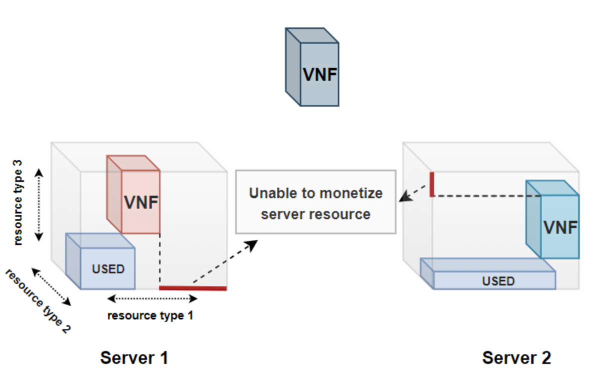

Fig. 2: VNF-to-server mapping with *3-dimensional* nodes.

ultimately, forms a path that connects servers hosting adjacent VNFs.

The range of different allocation possibilities, both for VNFs and virtual links, corroborates the combinatorial nature of NSE. More specifically, NSE belongs in the class of *NP-complete* combinatorial optimization problems, being, as already mentioned, a particular instance of the *NP-complete* VNE [22]. Furthermore, NSE can be tackled from a variety of optimization objectives, such as footprint minimization, revenue maximization, or load balancing.

B. Multi-dimensional VM allocation

Research on single-dimensional VM allocation has led to several algorithms that exhibit a high degree of optimality [12]. However, by considering multi-dimensional VMs and servers, the complexity of the problem increases significantly. That is, the notion of a VM *best fitting* a server can not be inherently extended to many dimensions. In practice, the multi-dimensional VM allocation problem is approached as a variant of the Vector Bin Packing (VBP) problem and algorithms implemented for the former draw upon research on the latter.

The VBP problem considers the allocation of n-dimensional items to n-dimensional bins by using as few bins as possible. It has been proved to be an *NP-hard* combinatorial optimization problem (*e.g.*, [23]); hence, a variety of heuristic algorithms have been developed as more appropriate means of tackling it, *e.g.*, [12], [13], [24], [25]. Assuming that items and bins correspond to VNFs and servers, respectively, VBP algorithms can be directly utilized to handle the *virtual node placement* phase discussed in Section II-A, although without guaranteeing success on our particular problem, since they cannot incorporate the bandwidth demands of a NS. We propose solutions for this later in the paper.

The limitations of a naive allocation policy, with three resource type requirements for VNFs, are merely illustrated in Fig. 2. In essence, if the *VNF* on top is placed in *Server 1*, a significant portion of its first resource type remains untapped, since the other two resource types of the server are fully occupied and, therefore, no additional VNFs can be accommodated. Instead, the placement of the *VNF* in *Server 2* seems more suitable for the exact opposite reason, *i.e.*, more efficient server resource utilization.

Essentially, the corresponding resources marked with the red line are implicitly bounded; the fact, however, that they are not assigned to any particular VNF prevents the infrastructure providers (InPs) from monetizing them. Techniques addressing this problem can be deemed efficient only when they minimize such resource wastage.

NSE deals with NSes that commonly consist of a multitude of VNFs. As a consequence, typical VBP algorithms that are driven by holistic criteria for the allocation of items to bins cannot be directly applied to the problem that we investigate. In this respect, our work aims at exploring potential ways of leveraging VBP insights within the NSE problem space. This is anticipated to yield more efficient resource utilization in servers, compared to other relevant methods that do not exercise any notion of multi-dimensional resource mapping.

TABLE I: Notations in the network model and the NS embedding formulations and algorithms.

Symbol	Description
V	the set of physical nodes within the substrate topology
V_F	the set of virtual nodes comprising a network service
E	the set of physical edges within the substrate topology
E_F	the set of virtual edges between virtual nodes
N	the set of all resource types
d_i^k	demand of virtual node i on resource k
d^{ij}	bandwidth demand of edge (i,j) in *Mbps*
r_u^k	residual capacity of physical node u on resource k
r_{uv}	residual capacity of physical edge (u,v) in *Mbps*
x_u^i	assignment of virtual node i to physical node u
M_u	the sum of the remaining capacities of each resource type for server u
f_{uv}^{ij}	amount of bandwidth assigned to link (u,v) for virtual edge (i,j) in *Mbps*
z_u	usage indicator of server u
s_u^i	suitability value for the mapping of VNF i to server u

III. Models and Mapping Efficiency Metrics

In this section, we initially introduce models for the NS and the substrate network, *i.e.*, datacenter (DC). Next, we describe in detail the various mapping efficiency metrics that we use.

A. Service and Network Models

Network Service (NS) model. We use a directed graph $G_F = (V_F, E_F)$ to express a NS. The set of nodes V_F includes all VNFs i that are associated with a demand value on resource k, denoted by d_i^k. Edges between nodes i and j, $i,j \in V_F$, are expressed as $(i,j) \in E_F$, whereas their respective bandwidth demands are denoted by d^{ij}.

Datacenter (DC) model. We model a DC as an undirected graph $G = (V,E)$, where V is the set of all physical nodes within the datacenter, *i.e.*, servers and switches. Furthermore, r_u^k is assigned to every physical node u to express its residual capacity on resource k. Likewise, $(u,v) \in E$ denotes the link between nodes u and v, $(u,v \in V)$. Each link $(u,v) \in E$ is associated with its residual bandwidth capacity, r_{uv}.

Table I summarizes the notations for the models explained above. We point out that all switches within a DC are assigned with zero resource capacities in order to inhibit the mapping of VNFs onto them.

B. Mapping Efficiency Metrics

Let the vector $d_i = [d_i^1, d_i^2, ..., d_i^n]$ indicate the requirements of a VNF $i \in V_F$, on a finite set of resource types $N = \{1, ..., n\}$. Similarly, we denote with $r_u = [r_u^1, r_u^2, ..., r_u^n]$ the vector of available resources of a server $u \in V$. According to these formulations, we now present the way each metric is computed, as well as its broader scope.

L2 norm-based greedy. Based on the *L2 norm-based greedy* metric, the server u that hosts VNF i in the most efficient way will be the one that minimizes the quantity:

$$s_u^i = \sqrt{\sum_{k=1}^{n} (r_u^k - d_i^k)^2} \tag{1}$$

Eq. 1 computes the euclidean distance between the $n-$ *dimensional* points expressed by the coordinates of d_i and r_u. Thus, the *L2 norm-based greedy* selects the server that minimizes this distance.

Dot product - Cosine similarity. Given the coordinates of all vectors r_u, and d_i, the respective dot-products can be computed with the analytical expression, *i.e.*, Eq. 2:

$$s_u^i = r_u \cdot d_i = \sum_{k=1}^{n} r_u^k \cdot d_i^k \tag{2}$$

As per the *dot-product* metric, the most suitable server u for the mapping of VNF i is the one that maximizes Eq. 2. The dot-product of two vectors can be also calculated as:

$$s_u^i = r_u \cdot d_i = ||r_u|| \cdot ||d_i|| \cdot \cos\theta \tag{3}$$

where $||r_u||$ and $||d_i||$ indicate the norms of the respective vectors, while $\cos\theta$ expresses the cosine of their angle (which is illustrated in Fig. 3). Consequently, and given that d_i is fixed for a specific VNF i, the dot-product in Eq. 3 increases if the value of $||r_u||$, $\cos\theta$ or both also increase.

Nevertheless, the effect of $||r_u||$ on the final selection of a server may outweigh that of $\cos\theta$. Since we deem the latter term as more appropriate for finding aligned (according to available and required resources) server-VNF pairs, we utilize the *cosine similarity* metric, shown in Eq. 4.

$$s_u^{i\,\prime} = \cos\theta = \frac{r_u \cdot d_i}{||r_u|| \cdot ||d_i||} \tag{4}$$

The coordinates of all vectors participating in Eq. 4 are positive[2] and thus their angle lies within $[0, \frac{\pi}{2})$. Therefore, the domain of $s_u^{i\,\prime}$ will be (0,1] and, apparently, the closer this value to one, the better.

Resource skewness. The notion of resource skewness is introduced in [15] as a means to quantify the fairness of the utilization of a server's different resource types. In accordance with the applicability of the other two metrics, we can compute the skewness of each server using:

$$s_u^i = \sqrt{\sum_{k=1}^{n} (\frac{r_u^k - d_i^k}{\bar{r_u}} - 1)^2} \tag{5}$$

[2]We only consider servers with sufficient resources for hosting the VNF.

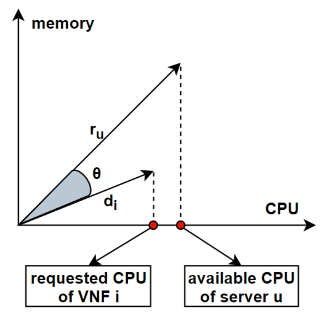

Fig. 3: The angle θ between the vector of a server's residual capacities, r_u, and the vector of a VNF's resource requirements, d_i, for a 2-dimensional scenario.

where $\bar{r_u}$ expresses the average consumption of all resources of server u. In this way, Eq. 5 can be used to assign a cost to each VNF-to-server mapping alternative, for a specific VNF i and the available servers u. The smaller the skewness of a server the more balanced its resource utilization is. Therefore, we would opt for the server u that minimizes Eq. 5.

Nevertheless, we compute skewness as shown in Eq. 6. Specifically, we omit the terms d_i^k from the nominators of the previous equation, and compute the average resource skewness of all servers according to their current consumption levels in order to obtain a single value for an individual DC. Hence, we adopt this metric as an embedding efficiency measure.

$$skewness = \frac{\sum_{u \in V} \sqrt{\sum_{k=1}^{n} (\frac{r_u^k}{\bar{r_u}} - 1)^2}}{|V|} \tag{6}$$

Manhattan distance. In the MILP that we later present, we aim at incorporating a metric that acts similarly to those already presented. More specifically, this metric will act as a penalty assigned to the servers selected during the NS embedding, in the sense that the more balanced their resource consumption is the better (and vice versa). It becomes apparent, though, that the metrics presented thus far are computed with non-linear terms. Hence, by incorporating them into a MILP, the element of linearity would be lost. Furthermore, the time complexity of the resulting solver would explode. To circumvent this, we have decided to incorporate the *Manhattan distance* into our MILP. According to Eq. 7, this distance is computed as the sum of the absolute differences of the respective coordinates of two vectors.

$$s_u^i = \sum_{k=1}^{n} |r_u^k - d_i^k| \tag{7}$$

Despite the fact that the *Manhattan* metric seems incapable of expressing aspects, such as the angle of two vectors, its simplicity and linearity make it computationally efficient. Thus, it stands out as the most appropriate metric for our MILP. Instead, the *L2 norm-based greedy* and the *cosine similarity* metrics are incorporated into the heuristic algorithm, which is presented in the next section.

Note that each resource dimension $k \in N$ can be assigned a weight (*e.g.*, by multiplying an α_k value with terms formed by d_i^k and r_u^k in previous equations). Such weights can adjust the impact of a resource dimension on the embedding computation. For instance, someone can assign the CPU dimension a weight larger than memory, which, in turn, may be given a larger weight than storage. In our evaluations (Section V-A), such adjustments are implied by different resource scarcities, mandated by VNF resource demands and server resource capacities.

IV. NS Embedding Methods

For the NSE problem that we study, we formulate, implement and evaluate three different embedding methods. The first one is a mixed-integer linear program (MILP), which exhibits the known drawback of exact methods, *i.e.,* high solver runtime. Therefore, we provide two heuristics in order to quickly obtain efficient solutions; the first one (*i.e.*, multi-dimensional heuristic) leverages information about the geometry of virtual and physical nodes, as opposed to the second one (*i.e.*, heuristic-cpu) which does not exhibit this feature. We further propose a service graph pre-processing method, which aims at the composition of resource-balanced VNF bundles, prior to their embedding.

A. Mixed Integer Linear Program

We initially introduce a MILP, which is essentially an extension of the MILP employed in [6] for *multi-dimensional* nodes. Besides the notations for the abstraction of a NS and a DC (Table I), we further use the binary variables x_u^i and z_u. The former indicates the placement of the virtual node $i \in V_F$ in the physical node $u \in V$, while the latter represents the allocation of a physical node $u \in V$ from the computed embedding. In addition, we utilize the variable f_{uv}^{ij} to denote the amount of flow of the virtual edge $(i,j) \in E_F$ that is mapped onto the physical edge $(u,v) \in E$, whereas M_u expresses the *Manhattan distance* between the vector of residual capacities of server u and the vector of (cumulative) assigned resources to server u.

We define the objective function (8), at which the value of α acts as a weight parameter that signifies the importance of efficient VNF-to-server allocations (for a more thorough discussion on the adjustment of this parameter, see Section VII). For a given NS, the objective function gives preference to solutions that utilize fewer servers, while, at the same time, minimize the volume of network traffic in the DC. Our MILP formulation is as follows:

Minimize

$$\sum_{u \in V} (z_u + \alpha M_u) + \frac{1}{\sum\limits_{(i,j) \in E_F} d^{ij}} \sum_{(u,v) \in E} \sum_{(i,j) \in E_F} f_{uv}^{ij} \quad (8)$$

subject to:

$$\sum_{u \in V} x_u^i = 1 \quad \forall i \in V_F \quad (9)$$

$$\sum_{\substack{v \in V \\ (u \neq v)}} (f_{uv}^{ij} - f_{vu}^{ij}) = d^{ij}(x_u^i - x_u^j)$$
$$i \neq j, \forall (i,j) \in E_F, \forall u \in V \quad (10)$$

$$M_u = \sum_{k \in N} (z_u r_u^k - \sum_{i \in V_F} d_i^k x_u^i) \quad \forall u \in V \quad (11)$$

$$\sum_{i \in V_F} d_i^k x_u^i \leq r_u^k \cdot z_u \quad \forall u \in V, \forall k \in N \quad (12)$$

$$\sum_{(i,j) \in E_F} f_{uv}^{ij} \leq r_{uv} \quad \forall (u,v) \in E \quad (13)$$

$$x_u^i, z_u \in \{0,1\} \quad \forall i \in V_F, \forall u \in V \quad (14)$$

$$f_{uv}^{ij}, M_u \in R_+ \quad \forall (u,v) \in E, \forall (i,j) \in E_F, \forall u \in V \quad (15)$$

With respect to expressions (9) to (15), constraint (9) ensures that each virtual node is mapped at exactly one substrate node (*i.e.*, server), while (10) enforces flow conservation, (*i.e.*, the amount of ingress traffic in a switch equals the amount of egress traffic). The penalty values calculated by the *Manhattan distance* are assigned through constraint (11). Notice that M_u is zero for all servers u that do not participate in the current embedding. Constraints (12) and (13) are used to prevent infeasible placements, *i.e.*, placements that violate node and link residual capacities. Another property of constraints (12) is that they bind x and z variables, *e.g.*, if $x_u^i = 1$ (for some $i \in V_F, u \in V$), then $z_u = 1$. Finally, constraints (14) and (15) enforce the variables' domains.

B. Multi-Dimensional Heuristic

Based on [6], we implement a heuristic that initially ranks the datacenter's racks in descending order, according to the average available capacity of their top-of-the-rack (ToR) to core switch links (***line 5*** in Algorithm 1). The attempt of embedding a NS within the substrate topology initiates from the first ordered rack and ends on the last (***line 6***). In case that the whole NS cannot be placed within a single rack, the *min-cut* algorithm is applied (***line 39***) to partition the request into segments, striving to minimize the amount of generated inter-rack traffic via the coordinated VNF and link assignment.

The aspect that we investigate appears during the placement of VNFs within a certain rack. Let $V_R \subset V$ be the set of servers within this rack. Then, for a specific VNF i of a given NS, the corresponding s_u^i values are computed in accordance to the utilized metric (***line 11***). The most suitable server for accommodating VNF i is expressed by $u = argmin_u(s_u^i)$ or $u = argmax_u(s_u^i)$, $u \in V_R$, depending on the metric (***line 12***). Thus, the substance of each metric lies on the way they sort the servers of each rack. Nevertheless, if the VNF can be placed in the same server as its predecessor, this placement is prioritized since, in this way, the generated traffic within the substrate network is reduced (***line 13***). Regarding the respective flow mappings, for adjacent VNFs within the same rack the mapping possibilities are limited, since there are unique paths that can establish their connection. In the case where such VNFs are accommodated in servers within different racks, though, the algorithm assigns the corresponding flow to inter-rack links that are the least loaded.

Algorithm 1 Heuristic for VNF placement

```
1: **Input**: $V, racks, V_F, E_F$
2: **Output**: mapping, rejected_service
3: $rejected\_service := True$
4: $N := |V_F|$
5: $s\_racks = sort(racks)$
6: **for** $R \in s\_racks$ **do**
7:     $prevServer := "None"$
8:     $placed\_vnfs := 0$
9:     **for** $i \in V_F$ **do**
10:        $placed\_vnf := False$
11:        compute $s^i_u, \forall u \in V_R$
12:        $sorted\_V_R = argsort_u(s)$
13:        **if** $prevServer \in sorted\_V_R$ and $i$ fits $prevServer$ **then**
14:            update $mapping$
15:            $placed\_vnf := True$
16:            $placed\_vnfs += 1$
17:            **if** $placed\_vnfs == N$ **then**
18:                $rejected\_service := False$
19:        **else**
20:            **for** $u \in sorted\_V_R$ **do**
21:                **if** $i$ fits $u$ **then**
22:                    update $mapping, prevServer$
23:                    $placed\_vnf := True$
24:                    $placed\_vnfs += 1$
25:                    **if** $placed\_vnfs == N$ **then**
26:                        $rejected\_service := False$
27:                        **break**
28:                **else**
29:                    **continue**
30:        **if** $placed\_vnf$ and $rejected\_service$ **then**
31:            **continue**
32:        **else**
33:            **break**
34:    **if** $rejected\_service$ **then**
35:        **continue**
36:    **else**
37:        **break**
38: **if** $rejected\_service$ and $|V_F| > 1$ **then**
39:    $V_{F_1}, V_{F_2} = minCut(V_F, E_F)$
40:    $mapping1, rejected1 = vnPlace(V, racks, V_{F_1}, E_F)$
41:    $mapping2, rejected2 = vnPlace(V, racks, V_{F_2}, E_F)$
42:    **if** $rejected1$ or $rejected2$ **then**
43:        **return** $\{\}, True$
44:    **else**
45:        **return** $mapping1 \cup mapping2, False$
46: **else if** $rejected\_service$ and $|V_F| = 1$ **then**
47:    **return** $\{\}, True$
48: **else**
49:    **return** $mapping, False$
```

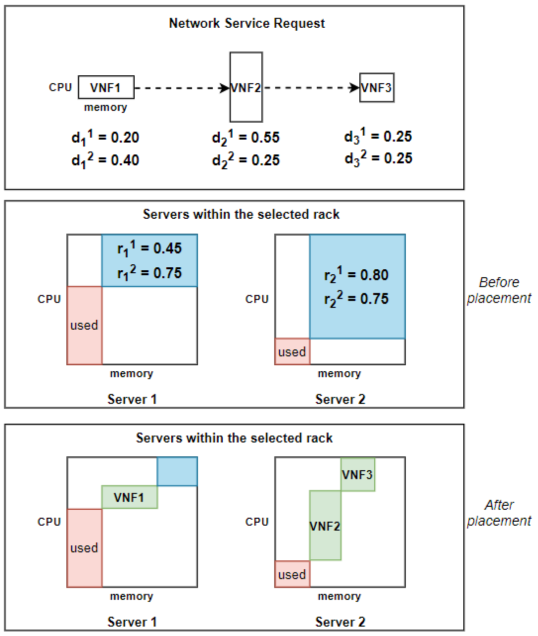

Fig. 4: NSE example with the multi-dimensional heuristic using the *Manhattan* metric.

Fig. 4 exemplifies the logic behind the proposed algorithm, in an attempt to map a NS to a datacenter rack. In this example, we consider the embedding of a NS (composed of three 2-dimensional VNFs) to a specific rack that consists of two servers. d^k_i- and r^k_u-values are used for the purposes described in Table I. The heuristic seeks to map the VNFs sequentially, *i.e.*, it will commence with the mapping of *VNF 1* and terminate with *VNF 3*. For the sake of simplicity, let us assume that the heuristic incorporates the *Manhattan* metric. Thus, we compute the following:

$$s^1_1 = |0.45 - 0.20| + |0.75 - 0.40| = 0.60$$
$$s^1_2 = |0.80 - 0.20| + |0.75 - 0.40| = 0.95$$

and we derive that *VNF 1* will be assigned to *Server 1*. In more detail, this decision is made based on the following two conditions: (i) *Server 1* has sufficient capacity to accommodate *VNF 1* and (ii) $s^1_1 = 0.60 < 0.95 = s^1_2$ (*i.e.*, assigning *VNF 1* to *Server 1* is more efficient than assigning it to *Server 2*, according to the utilized metric). Subsequently, r^k_1-values are recomputed, since the available resources of *Server 1* have changed. Ideally, *VNF 2* would also be placed on *Server 1*; however, it is apparent that it does not fit due to lack of available CPU resources. Thereby, *VNF 2* is mapped onto *Server 2* and r^k_2-values are recomputed. We note that although *VNF 3* fits *better* in *Server 1* (*i.e.*, $s^3_1 = 0.10 < 0.25 = s^3_2$, considering the updated r^k_u-values), it will be placed on *Server 2*, where its predecessor has been assigned in order to avoid the generation of additional intra-rack traffic.

This example demonstrates a critical aspect of the problem concerned. That is, striving for a balanced resource consumption may lead to larger traffic volumes and, thus, the inherent capability of an algorithm to skew in favour of the former against the latter (or the opposite) will significantly affect the overall NSE efficiency.

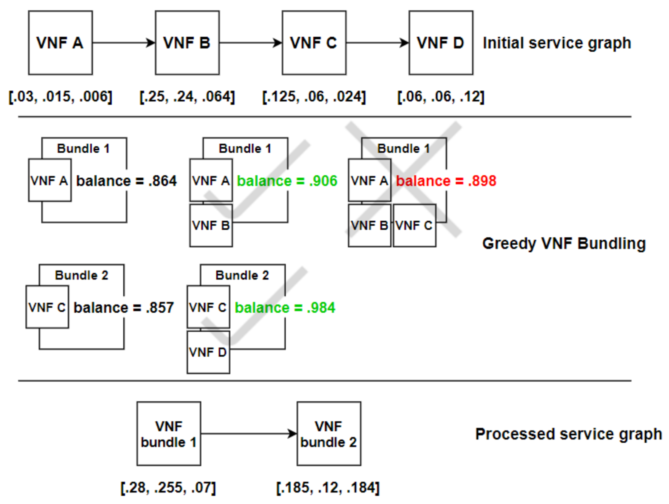

Fig. 5: The *Greedy VNF Bundling* (*GVB*) service graph pre-processing method.

C. Heuristic Baseline

Furthermore, we have implemented another heuristic that we term as *heuristic-cpu*. This heuristic serves as a baseline, empowering us to quantify the potential gains from NSE with multiple resource dimensions. In particular, this heuristic is similar to the one presented in Section IV-B. However, instead of using any of the mapping efficiency metrics defined in the previous sections, it ranks the servers within a rack according to their available CPU capacity, in descending order.

D. Greedy VNF Bundling (GVB)

We devise a VNF-graph pre-processing method, namely *Greedy VNF Bundling* (*GVB*), which seeks to generate bundles out of VNFs, while maintaining the same resource requirements with the sum of individual VNFs that comprise each bundle. The primary aim of *GVB* is to create VNF bundles with more balanced resource demands (across the resource dimensions), thereby, augmenting the underlying NSE method in VNF placement. The main intuition behind this approach is that bundling VNFs with resource intensity in different dimensions (*e.g.,* a CPU-intensive VNF with a memory-intensive one) can generate more balanced VNF bundles. One restriction here is that bundling should be applied only to consecutive VNFs (in the service chain) in order to enforce correctness in the processing of traffic.

A key-feature of *GVB* is the way that it groups adjacent VNFs. More specifically, it commences with an empty bundle, and places the first VNF of the chain within it. Subsequently, it assigns the following VNF in the same bundle, if the sum of the respective VNF resource requirements results in a more resource-balanced (bundled) VNF, provided that its requirements do not exceed the capacity of a server; otherwise, it opens up a new bundle and repeats the same procedure. The level of resource balance is quantified using Eq. 4, where $r_u = \vec{1}$, and d_i expresses the resource requirements of the i^{th} bundle.

To exemplify *GVB*, Fig. 5 illustrates a service graph comprising four VNFs, with their respective resource demand d_i vectors (*i.e.*, normalized CPU, memory, and storage demands, respectively). For instance, $d_A = [.03, .015, .006]$ indicates that *VNF A* requires 3% of a server's CPU, 1.5% of its memory, and 0.6% of its storage capacity. According to the *GVB* procedure, an empty bundle (*i.e.*, *Bundle 1*) is spawned, and *VNF A* is assigned to it. Eq. 4 computes the level of resource-balancing of *Bundle 1*, as follows:

$$balance(Bundle1) = \cos\theta = \frac{\vec{1} \cdot [.03, .015, .006]}{||\vec{1}|| \cdot ||[.03, .015, .006]||} = .864$$

Subsequently, we examine whether *VNF B* should be placed into *Bundle 1*. This will result in *Bundle 1* requiring

$$[.03 + .25, .015 + .24, .006 + .064] = [.28, .255, .07]$$

resources, and, thus:

$$balance(Bundle1) = \cos\theta = \frac{\vec{1} \cdot [.28, .255, .07]}{||\vec{1}|| \cdot ||[.28, .255, .07]||} = .906$$

Since the generated bundled VNF is more balanced, we assign *VNF B* to *Bundle 1*. Next, we attempt to assign *VNF C* to *Bundle 1*, also. In this case, we obtain:

$$balance(Bundle1) = \cos\theta = \frac{\vec{1} \cdot [.405, .315, .094]}{||\vec{1}|| \cdot ||[.405, .315, .094]||} = .898$$

which implies a degraded level of resource balance. Hence, we do not assign *VNF C* to *Bundle 1* and, instead, a new bundle (*i.e.*, *Bundle 2*) is spawned. Following a similar approach, we obtain that *Bundle 2* comprises *VNF C* and *VNF D*. Finally, this pre-processing phase results in a new service graph (depicted at the bottom of Fig. 5), which consists of two VNF bundles connected with a virtual link (*i.e.*, the edge between *VNF B* and *VNF C*).

V. Evaluation

In this section, we initially present the evaluation environment, *i.e.*, the simulation parameters for the DCs and the NS requests, as well as our evaluation metrics (Section V-A). Subsequently, in Section V-B we examine the impact of incorporating the *L2 norm-based greedy* and the *cosine similarity* metrics into the heuristic presented in Section IV-B. The resulting heuristics are termed as *heuristic-L2* and *heuristic-cos*, respectively, in the remaining of the paper. Both of these algorithms are compared against the single-dimensional version of the same heuristic, *i.e.*, that merely accounts for the CPU resource dimension. Next, in Section V-C we discuss the comparative evaluation of the heuristic that stands out (from the previous comparison) and the MILP presented in Section IV-A. Finally, we assess the impact of the *GVB* pre-processing scheme on the embedding efficiency in Section V-D.

A. Evaluation Environment

Each one of the evaluated algorithms is handling the embedding of NS requests to a DC with a two-layer hierarchical topology. More specifically, the simulated DC contains 200 servers, organized in 10 racks. The corresponding 10 top-of-the-rack (ToR) switches communicate via the topology's core layer, which includes 5 core switches. Each server has a computing capacity of 64 vCPUs, 512 GB of main memory, and 8 TB of storage. Lastly, each server is connected to its ToR switch with a 4 Gbps link, whereas the links connecting the ToR with the core switches have capacity of 16 Gbps, as shown in Table II.

Prominent cloud providers, such as Microsoft [26], Amazon [27], Google [28] and IBM [29], offer the deployment of VM instances classified, among others, to *compute-*, *memory-* and *storage*-optimized. This provides evidence for the resource demand diversity of workloads executed in cloud environments, as well as the versatility that the cloud provides for the accommodation of the various resource demands. Based on these observations, we distinguish between three types of VNFs; namely *compute-*, *memory-* and *storage*-optimized VNFs. Subsequently, each VNF instance may require 2, 4, 8, 16 or 32 vCPUs. The required capacities of the other resource types, though, depend on the VNF type. More specifically:

Compute-optimized VNF instances are specified by a constant memory-to-vCPU ratio of 4, as well as a constant storage-to-vCPU ratio of 25.
Memory-optimized VNF instances are associated with a constant memory-to-vCPU ratio of 8 and a constant storage-to-vCPU- ratio of 32.5.
Storage-optimized VNF instances have a constant memory-to-vCPU ratio of 8 and a constant storage-to-vCPU ratio of 232.5.

Therefore, if a VNF requires 8 vCPUs and is classified as a memory-optimized instance, the required memory and storage capacities will be 64GB and 260GB, respectively. Apparently, these VNF instances yield a *positive correlation* between the various resource dimensions (pairwise), irrespective of their type. Although this correlation may influence the behavior of our algorithms, such instances are deemed more realistic.

Furthermore, the inbound traffic of a NS is set to a random value within the range [10, 100] (Mbps), whereas the inbound-to-outbound traffic ratio for each VNF in the service chain is adjusted to a random value within [0.5, 1.5]. This traffic ratio essentially captures potential implications of VNFs on the volume of outbound traffic (compared to inbound traffic). This stems from the fact that VNFs which perform packet inspection, redundancy elimination or caching (amongst other processing operations) may suppress or amplify the inbound traffic [6].

In terms of incoming NSE requests, we consider the more realistic case of expiring NSes with a lifetime of three to nine time intervals. Such NSE requests arrive at the DCs with a Poisson distribution (30 on average per time interval). Furthermore, each NS comprises three to ten VNFs (Table IV).

Our simulations are conducted on a server with 8 CPU cores at 2.1 GHz and 8 GB of RAM, using the Linux-based Ubuntu 16.04 (LTS) operating system. The simulation environment is implemented in Python, while for the MILP we rely on the Gurobi [30] solver.

TABLE II: DC Model Parameters

Number of Core Switches	5
Number of Racks	10
Number of Servers per Rack	20
Capacity of server vCPUs	64
Capacity of server memory	512GB
Capacity of server storage	8TB
ToR-to-Server link capacity	4Gbps
Inter-rack link capacity	16Gbps

TABLE III: NSR Model Parameters

Inbound NS traffic	uniform distr. [10,100]
Outbound-to-inbound traffic ratio	uniform distr. [0.5,1.5]
Splittable flow	Yes

TABLE IV: Evaluation Environment Parameters

Number of NSes per time interval	Poisson(30)
Number of VNFs per NS	uniform distr. [3,10]
NS type	expiring
NS running duration	random [3,9] intervals

For the comparison between the NSE methods, we use the following metrics:

Request Acceptance Rate, which is computed as the ratio of the successfully embedded NSes over the total number of incoming NSE requests.

Resource Utilization, which corresponds to the sum of the physical resources (*i.e.*, CPU, memory, storage) that have been allocated for the VNFs.

Resource Skewness, which is computed according to Eq. 6 (Section III-B).

Intra-, Inter-rack traffic, which is the mean volume of traffic observed in the respective links in various snapshots of the simulated network infrastructure.

B. Evaluation of Metrics

We hereby study the efficiency of the two multi-dimensional mapping metrics (*i.e.*, *L2 norm-based greedy* and *cosine similarity*), as they have been incorporated into the proposed heuristic. Fig. 6 illustrates that the two heuristics that incorporate the metrics yield more efficiency than the *heuristic-cpu*. More specifically, we observe a higher acceptance rate for the *heuristic-L2* compared to the *heuristic-cos*, while both of them outperform the third algorithm. Bearing in mind that a 1% difference in the acceptance rate at the end of the experiment corresponds to the rejection of more than 70 NSes, the improvement by the proposed heuristics is deemed considerable.

To gain more insights on embedding efficiency, we further examine the resource utilization achieved by the three methods. To this end, we depict the percentage-wise difference of the resource utilization obtained with the *heuristic-L2* and the *heuristic-cos* (computed as a moving average) compared to the one obtained by the *heuristic-cpu*. According to Fig. 7,

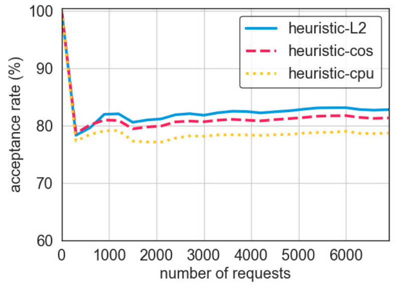

Fig. 6: Request acceptance rate.

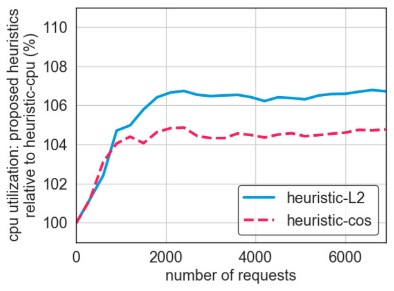

Fig. 7: CPU utilization of the multi-dimensional heuristics, in relation to the single-dimensional.

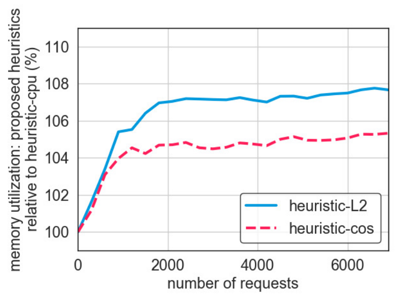

Fig. 8: Memory utilization of the multi-dimensional heuristics, in relation to the single-dimensional.

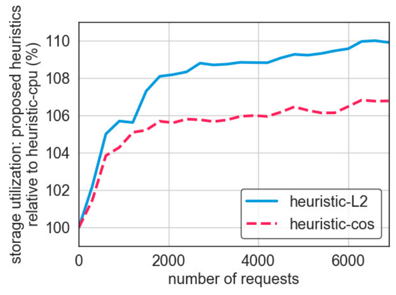

Fig. 9: Storage utilization of the multi-dimensional heuristics, in relation to the single-dimensional.

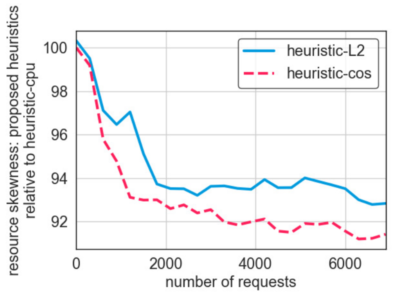

Fig. 10: Resource skewness of the multi-dimensional heuristics, in relation to the single-dimensional (lower is better).

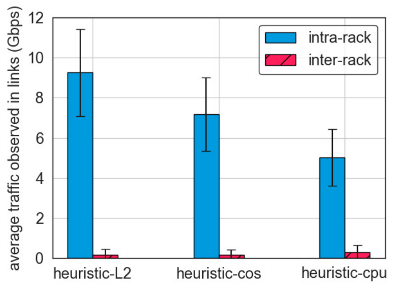

Fig. 11: Intra-rack and inter-rack traffic distribution.

the percentage of the excess CPU that is assigned to NSes (more precisely, to VNFs) by the multi-dimensional heuristics is equal or lower than the respective percentages concerning the rest resource dimensions (Figs. 8, 9); this pertains to both *heuristic-L2* and *heuristic-cos*. This outcome is expected, since the *heuristic-cpu* accounts for the CPU resource dimension and, thus, assigns it to VNFs in a less naive manner, compared to the way it allocates memory and storage. However, with respect to the allocation efficiency of all resource types, we observe significant gains of at least 4%, for the *heuristic-cos*, and at least 7%, for the *heuristic-L2*.

To further comprehend the benefits of the multi-dimensional heuristics, we investigate the extent to which resource balance comprises a key aspect of their efficiency. According to Fig. 10, the proposed algorithms yield a more balanced consumption of the different resource types than the *heuristic-cpu*. Specifically, by quantifying this balance with the *resource skewness* metric, the two heuristics lead to almost 8% more balanced servers on average. In addition, the algorithm that employs the *cosine similarity* metric yields the best skewness, since its respective skewness values are typically lower than those of the *heuristic-L2*. Nevertheless, we cannot infer that the improved resource utilization stems exclusively from skewness (*i.e.*, balanced consumption of server resource types). If that was the case, the *heuristic-cos* would yield at least similar (if not better) efficiency than the *heuristic-L2*. However, our previous results provide the opposite indication.

This finding essentially raises the need for a deeper investigation of the way that the two metrics influence the embedding decisions. Recall that the *cosine similarity* metric ranks servers according to Eq. 4. As such, a previously unused server may be selected for the placement of a VNF, although this VNF may fit into a previously allocated server within the same rack. This can, in turn, lead to the mapping of subsequent VNFs onto the same server (since it currently hosts only one VNF), although (according to Algorithm 1) there will be no consideration of the degree of alignment between the VNF resource requirements and the available resources of the server.

Consequently, the *heuristic-cos* utilizes the *cosine similarity* less frequently in the computation of VNF-to-server mappings, which eventually affects its efficiency. On the other hand, the *heuristic-L2* tends to map a VNF to the server that minimizes Eq. 1 (*i.e.,* its embedding approach is dictated by the *Best Fit* principle). As such, subsequent VNFs are less likely to be placed onto the same server, due to lack of available resources. Thereby, the *L2 norm-based greedy* metric is used more frequently for the computation of the additional VNF-to-server assignments.

Along these lines, we reach the following observations: (i) the *heuristic-L2* yields a higher degree of server consolidation compared to the *heuristic-cos*, and (ii) the effectiveness of a multi-dimensional mapping efficiency metric is highly dependent on the embedding heuristic. These observations are essentially reflected by our resource allocation and skewness results. The *cosine similarity*, in particular, has been proved a more efficient metric compared to the *L2 norm-based greedy* [14]. However, this gain of the *cosine similarity* is not translated into better embedding efficiency, since the corresponding metric is

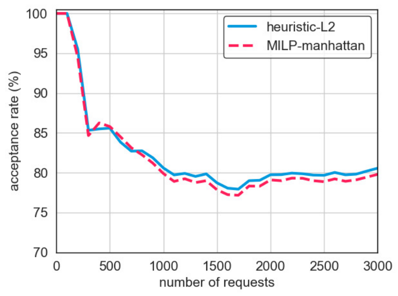

Fig. 12: Request acceptance rate.

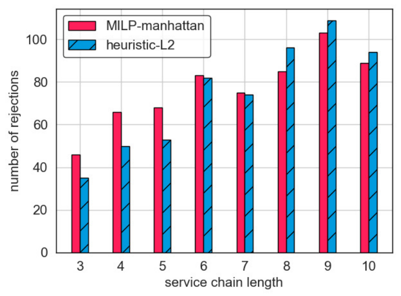

Fig. 13: Number of NSE rejections for the different service chain lengths.

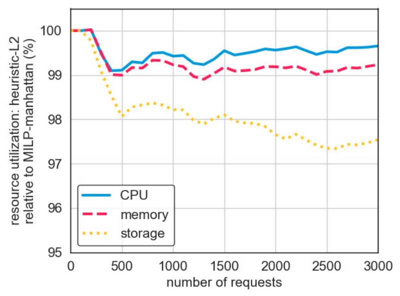

Fig. 14: Resource utilization of the *heuristic-L2*, in relation to the *MILP-manhattan*.

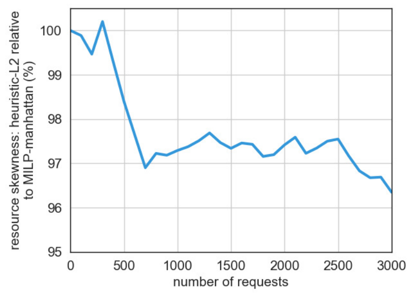

Fig. 15: Resource skewness of the *heuristic-L2*, in relation to the *MILP-manhattan*.

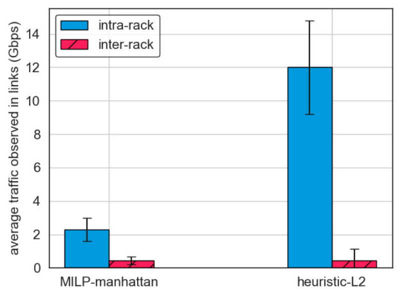

Fig. 16: Intra-rack and inter-rack traffic distribution.

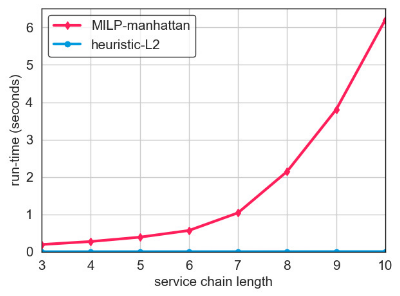

Fig. 17: Average solver run-time for the different service chain lengths.

less frequently utilized.

Lastly, Fig. 11 illustrates the average intra- and inter-rack traffic generated on the DC. According to this plot, all algorithms achieve a negligible degree of inter-rack traffic, with the *heuristic-cpu* exhibiting higher variation from the average value. This is attributed to the more frequent *min-cut* triggering, *i.e.*, the algorithmic element that enables the partitioning of a NS across multiple racks. We also observe a higher volume of intra-rack traffic for the *heuristic-L2*. This stems from the following observations: (i) the *heuristic-L2* admits a larger number of incoming NSes, and (ii) it exhibits a tendency for NS partitioning across multiple servers, as explained in our previous discussion about the *L2 norm-based greedy* metric.

C. Comparison of MILP and heuristic-L2

As explained in Section III-B, we incorporate the *Manhattan distance* into a MILP in order to penalize inefficient assignments, while retaining the exact method's linearity. In the following, we compare the resulting algorithm (*i.e.*, *MILP-manhattan*) with the *heuristic-L2*, which stood out from the previous comparison. To this end, we assess the margin between the heuristic algorithm and the optimal solutions computed by the *MILP-manhattan*.

According to Fig. 12, the two algorithms manage to embed roughly the same number of NSes. Their marginal difference of $\sim 1\%$ (that unexpectedly favors the *heuristic-L2*) is further investigated via additional micro-benchmarks. More precisely, an analysis of the rejections (Fig. 13) uncovers the limited computational capacity of the heuristic for embedding long (≥ 8 VNFs) NSes compared to the MILP; instead, the heuristic utilizes the available resources for embedding more short NSes (≤ 5 VNFs). This computational limitation of the *heuristic-L2* can be mainly attributed to its poor *search space exploration* capability (compared to the *MILP-manhattan*), since the exercised *min-cut* policy reduces the VNF-to-server allocation combinations.

After approximately $3{,}000$ NSE requests, we observe the relative utilization for each resource dimension, illustrated in Fig. 14. It is apparent that storage (which comprises the most abundant resource type) has the most profound impact on the overall resource utilization of the *heuristic-L2*. As such, we conclude that the proposed heuristic exhibits only a minimal (1-2%) margin from the MILP in terms of maximizing resource utilization. This, combined with the improved resource skewness (*i.e.*, Fig. 15), empower the *heuristic-L2* to achieve high efficiency on par with the *MILP-manhattan*.

We observe, though, a substantial difference between the two methods in terms of consolidation, in favor of the *MILP-manhattan*. This is implied by the respective intra-rack traffic level shown in Fig. 16. Nevertheless, MILP exhibits a significant drawback, as its solver run-time grows exponentially with the problem size. For instance, according to Fig. 17, the computation of the embedding for a NS with 10 VNFs requires 6 seconds on average, whereas the heuristic's run-time is in the order of milliseconds for all NS lengths. The scalability limitation of the MILP could be potentially addressed by employing relaxation and rounding techniques, similar to our previous work in [6]. However, the resulting linear program (LP) is expected to yield a degree of sub-optimality (compared

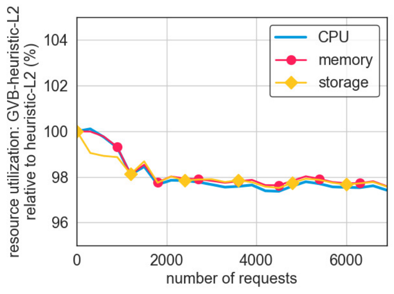

Fig. 18: Resource utilization of the *GVB-heuristic-L2*, in relation to the *heuristic-L2*.

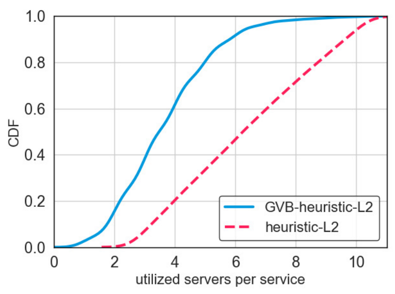

Fig. 19: CDF of the number of servers utilized per network service.

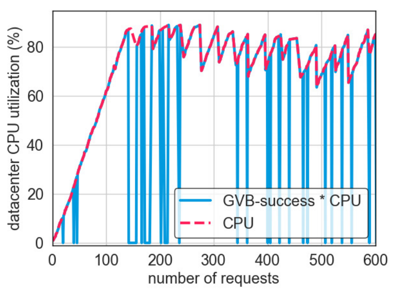

Fig. 20: *GVB* triggering frequency.

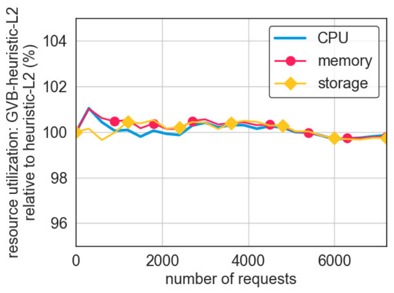

Fig. 21: Resource utilization of the *GVB-heuristic-L2*, in relation to the *heuristic-L2* (threshold-based variant).

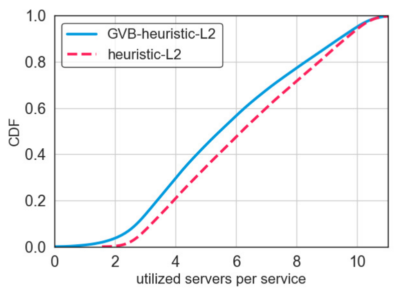

Fig. 22: CDF of the number of servers utilized per network service (threshold-based variant).

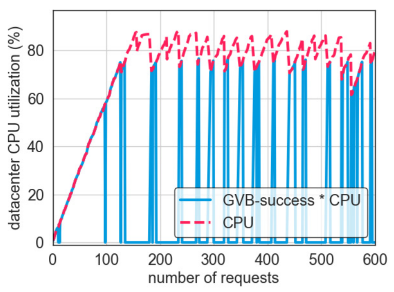

Fig. 23: *GVB* triggering frequency (threshold-based variant).

to the MILP), whereas its solver run-time would still be considerably higher than the heuristic.

D. Comparison of GVB-heuristic-L2 and heuristic-L2

We hereby investigate the potential gains from the *GVB* pre-processing method, which generates VNF bundles in order to augment VNF embedding. To assess the efficiency of *GVB*, we couple it with the *heuristic-L2*, which stands out as the most prominent embedding method (in the following, we term this combination as *GVB-heuristic-L2*). In particular, *GVB* processes the initial service graph generating a potentially shorter graph with bundled VNFs, which is, in turn, conveyed to the *heuristic-L2* for optimized embedding. Since VNF bundles are associated with higher resource requirements (compared to the VNFs of the initial service graph), this reduces the search space for VNF embedding optimization. Consequently, if the embedding of a bundled VNF is not possible, instead of the embedding request rejection, we perform another embedding attempt using the initial (non-bundled) service graph. In such case, we assign *zero* to a *GVB-success* binary variable. We also assign *zero* to the same variable, whenever the number of generated bundles equals the number of initial VNFs (which might occur if, for instance, all VNFs of a service chain are CPU-intensive). Otherwise (*i.e.*, whenever *GVB-heuristic-L2* successfully embeds bundled VNFs), we assign the value of *one* to this variable.

According to Fig. 18, the assignment of CPU, memory, and storage of the *GVB-heuristic-L2*, in relation to the respective allocations of the *heuristic-L2*, is approximately 2% less across all three resource dimensions. This implies slightly lower profits for an InP; however, the VNF bundling method appears to yield higher embedding efficiency than *heuristic-L2*, as indicated in Fig. 19. More precisely, 90% of the services embedded with the *GVB-heuristic-L2* are assigned to 5 servers at most, whereas the respective number of servers for *heuristic-L2* is 9. Although these servers usually belong to the same rack, the higher degree of VNF co-location achieved by *GVB-heuristic-L2* reduces the volume of generated traffic (the respective results are omitted, due to space limitations).

To shed more light on the behavior of the bundling algorithm, we examine its success rate. In Fig. 20, a continuous line illustrates the product of the *GVB-success* variable with the average CPU utilization of the datacenter, whereas the dashed line merely represents the latter. These curves intersect whenever a bundled VNF placement occurs, *i.e.*, *GVB-success=1*. Unexpectedly, the *GVB-heuristic-L2* yields *successes* even at very high utilization levels (*e.g.*, above 80%), which indicates that *GVB* is triggered very frequently.

The insights gained from the behavior of *GVB-heuristic-L2* lead to a critical observation, which stems from an intuitive principle of bin-packing, *i.e.*, having many and small items, instead of fewer and larger, when closing up the bins (*i.e.*, when bins are almost full), is more effective. Therefore, we evaluate a variant of *GVB-heuristic-L2*, whose main difference from the initial bundling scheme is that it is not executed when the infrastructure exceeds a certain utilization level. In our simulations, we set this threshold to 75% of the mean DC CPU utilization. As such, this *GVB* variant is triggered only for low and medium utilization levels. Instead, it degenerates to the basic *heuristic-L2* for the embedding of the initial VNFs (in analogy to many and smaller items), when the infrastructure utilization is high (analogous to bins ready to close up). With this behavior, *GVB* lays the foundation for a balanced resource consumption within servers.

This adjustment in the behavior of *GVB* yields even higher resource efficiency, since it leads to (i) equivalent resource allocations with the *heuristic-L2* (Fig. 21) and (ii) slightly improved VNF co-location efficiency, according to Fig. 22. Fig. 23 corroborates that *GVB* is triggered (frequently) only at low and medium utilization levels.

Consequently, the *GVB-heuristic-L2* effectively couples the *cosine-similarity* (according to the bundling logic described in Section IV-D) with the *L2 norm-based greedy* (since the *heuristic-L2* is employed to map the service graph with the bundled VNFs). Thereby, the *GVB-heuristic-L2* promotes the synergy of the two metrics, whose gains are illustrated in Figs. 21 and 22. Additional gains could be achieved by coupling the proposed bundling scheme with a NSE MILP, as well. In particular, the smaller number of VNFs in the service graph (as the outcome of VNF bundling) is expected to reduce the MILP solver run-time, which is illustrated in Fig. 17.

VI. Related Work

Throughout this section, we discuss related work on the following two areas: (i) NSE and (ii) multi-dimensional VM assignment.

A. Network Service Embedding

Benson et al. propose CloudNaaS [31], a system that allows the deployment of cloud applications offering features such as virtual network isolation, service differentiation and flexible deployment of middleboxes. The network controller of CloudNaaS is responsible, among others, for the assignment of VMs (that compose the cloud application) to physical servers considering their communication dependencies and requirements. Authors employ a heuristic approach for tackling this problem, with its ultimate goal being to minimize the amount of flow mapped on the network links.

Another research work that considers the problem of NSE is Stratos [20]. In particular, Stratos comprises a framework for a multitude of NFV orchestration functions, such as service chaining and resource provisioning. The resource provisioning aspect of Stratos, which is the most relevant to our work, mainly deals with flow distribution, *i.e.,* the assignment of flows to the various VNF instances. This problem is formulated as a linear program.

Nestor [6] and DistNSE [9] tackle the multi-provider NSE problem, taking into account the privacy restrictions imposed by the different participating providers. Both Nestor and DistNSE decompose this into two sub-problems: (i) service chain partitioning (among providers), and (ii) mapping of each chain segment onto the respective provider's domain. While Nestor relies on a centralized broker for service chain partitioning, DistNSE handles this in a distributed manner, using an embedding protocol (which essentially obviates the need for any centralized orchestrator). The intra-provider mapping of both Nestor and DistNSE (which is the particular sub-problem of relevance to our work) is restricted to a single dimension for all resource types (*i.e.,* CPU demands for VNFs and bandwidth demands for links).

In [7], *Fu et al.* present a deep reinforcement learning approach for the embedding of network services. As implied in the paper, the inherent dynamic nature of Internet of Things should be supported by NSes being embedded by algorithms that can autonomously adjust their decisions based on past knowledge.

The NSE problem has also been studied in the context of cellular network slicing. The aim here is to optimize the placement of VNFs that implement certain elements of a virtualized cellular network, such as the eNodeB (eNB), the Serving Gateway (S-GW), the Packet Data Network Gateway (P-GW), and the Mobility Management Entity (MME). In particular, authors in [10] propose an exact method for the embedding of such VNF-graphs onto a virtualized cellular core. Besides capacity constraints, the proposed method optimizes the placement of VNFs, taking into account delay budgets between VNFs, as mandated by 3GPP. *Papagianni et al.* follow a different approach to this NSE problem, as they promote the sharing of VNFs among multiple service chains in a network slice, as means to reduce the deployed VNF instances and, consequently, the provisioning and management cost of the virtualized cellular network [8]. To this end, the authors propose a MILP formulation for NSE with shared VNFs.

Even though all of the above-mentioned studies shed light on critical aspects of NSE, the fact that they are confined in single-dimensional nodes (*i.e.,* CPU demand) restricts their applicability to NSes with diverse requirements across multiple resources. Hence, we address this limitation by providing an efficient NSE algorithm that accounts for multiple resource dimensions, eventually leading to more efficient utilization and higher revenues for InPs.

B. Virtual Machine Assignment

The multi-dimensional VM allocation problem is examined as a specific application of the more generic VBP problem. Within the context of mapping VMs to physical hosts, several studies (*e.g.*, [15], [32]) primarily focus on server consolidation, *i.e.*, the allocation of VMs into as few servers as possible.

Ghodsi et al. [33] is considered to be one of the most influential studies on the discussed topic. This study is among the first that approaches the fair allocation of multiple resources as pertaining to users with heterogeneous requirements. To this end, they propose Dominant Resource Fairness (DRF), a multi-dimensional allocation scheme that meets a multitude of preferable properties. However, the applicability of DRF can be seen as more suitable for large computer clusters, where all users have equal rights of utilizing their resources, as opposed to commercial cloud infrastructures, at which profit maximization is commonly sought.

An additional research work tackling this problem is [15]. In particular, *Xiao et al.* implement an algorithm that aims at using the least amount of hosts possible. In addition, the utilization of host resources must not exceed respective pre-determined thresholds so as to avoid VM performance degradation. In order to circumvent the difficulty of allocating resources across multiple dimensions, authors embrace the notion of resource skewness. Essentially, skewness acts as means to quantify the efficiency of server resource utilization and also trigger VM migrations that improve the overall server utilization.

Motivated from VM placement problems, [12] and [13] propose new heuristics for the multi-dimensional VM consolidation and the VBP problem in general. In both studies, authors underline the difficulty of extending the objective of *best fitting* VMs to servers, when considering both of them as multi-dimensional vectors. To this end, they propose two heuristics; namely, the *Dot Product* and the *Norm-based Greedy*. The former is able to evaluate the VM requirements against the residual capacity (across multiple dimensions) of servers, while the latter utilizes a certain norm to quantify the distance between the aforementioned vectors.

Tetris [14], a cluster scheduler that matches tasks with machines according to requirements along multiple resources, is proposed to confront the drawbacks of fairness- and slot-based schedulers. Authors suggest that the former do not optimize job completion time, while the latter result in either excessive resource allocation or resource wastage. To this end, Tetris incorporates the *cosine similarity* (defined in Section III-B) for calculating the alignment of tasks with servers.

The objective of our study is aligned with related work [6], [20], [31], which targets the minimization of the generated inter-rack traffic and also ensures correctness for the embedded service chains. Since we deal with multi-dimensional virtual and substrate nodes, we further employ the efficiency metrics used in [12], [13], [14] and [15], which are discussed in detail in Section III-B. To the best of our knowledge, the coupling of NSE with the multi-dimensional VM allocation problem has not been investigated in the literature. Our work essentially enables the computation of efficient NS embeddings under a pragmatic scope, which leads to a more balanced resource consumption, eliminating resource wastage and potential revenue losses for InPs.

VII. Conclusions

We conducted a comprehensive study on the problem of NSE across multiple dimensions. To this end, we compared the efficiency of various multi-dimensional mapping metrics, which were introduced in the context of multi-dimensional VM assignment. These metrics (*i.e., L2 norm-based greedy*, *cosine similarity*, and *Manhattan distance*) were integrated into heuristic or exact methods, enabling us to quantify the gains of NSE across multiple dimensions.

According to the VNF types and the resource requirements they raise, as well as the initial resource capacities of servers considered during our simulations (see Section V-A), we can infer that (i) there is a positive correlation (pairwise) between the different resource dimensions, and (ii) the CPU resource type is most in demand. These are aspects that affect the efficiency of a mapping metric. In fact, we conducted tests with VNFs that require a uniformly random amount of resources within the range [0.01,0.50], *i.e.*, from 1% to 50% of a server's initial resources, across all resource dimensions. In this setting, the *heuristic-cos* performed the best with respect to all performance indicators. Nevertheless, such parameters are not in line with real-life ones and, hence, we left them out of our evaluation.

The *Manhattan distance* between the vector of residual capacities and the vector of demands, as pertaining to a server and a VNF, respectively, takes values that are tightly related to the number of different resource types, *i.e.*, resource dimensions. For n dimensions, and given that all vector coordinates are normalized, the M_u could theoretically lie within the range [0,n). Therefore, the α value is adjusted to 10, considering (i) that we are dealing with three dimensions, (ii) the domains of the other variables, and (iii) that we wanted to emphasize the MILP's capability of obtaining efficient VNF-to-server assignments. At the same time, as already discussed in Section V, this inherently strengthens the algorithm's capability of consolidating the VNFs of a specific NS. Nevertheless, there will be a pivotal value of α that, if exceeded, the second term of Eq. 8 will be dominated, resulting in inefficient NS embeddings, due to unfavourably large inter-rack traffic volumes within the DC.

Our evaluation results corroborate the improved resource efficiency of multi-dimensional NSE compared to a single-dimensional counterpart. The resource efficiency gains mainly stem from (i) a more balanced resource consumption, and (ii) increased VNF consolidation, which confines both intra- and inter-rack traffic in the DC. Additional insights gained from our evaluations indicate that the way resource consumption balancing is achieved is crucial, since the *heuristic-L2* and the *heuristic-cos* achieve similar resource skewness levels, but they exhibit notable differences in the rest of the efficiency metrics. This brings us to a second conclusion, which is the critical impact of the embedding algorithm on the utilization of the mapping efficiency metric. In our case, this pertains to the frequency that the metric is utilized. Furthermore, we identified an insignificant gap in the optimality between the *heuristic-L2* and the MILP (which is essentially outweighed by the substantial solver run-time of the latter).

Lastly, we evaluated a VNF graph pre-processing method, which strives to group VNFs into resource-balanced bundles. Our results indicate that the proposed bundling scheme, coupled with the *heuristic-L2* (*i.e.*, *GVB-heuristic-L2*), can achieve a high degree of VNF co-location while maintaining fair resource utilization levels. We further evaluated a variant of *GVB-heuristic-L2*, at which the VNF bundling scheme is triggered below a pre-defined resource utilization level. This empowers an InP to properly adjust the trade-off between VNF co-location and resource efficiency in order to increase their revenue by either minimizing resource wastage (*i.e.,* saving capacity to accommodate additional services) or by embedding services at potentially premium prices via a minimal embedding footprint (in order to meet strict latency requirements).

VIII. Acknowledgements

This work is supported by the MESON (Optimized Edge Slice Orchestration) project, co-financed by the European

Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH–CREATE–INNOVATE (project code: T1EDK-02947).

References

[1] J. Sherry *et al.*, "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[2] A. Greenhalgh *et al.*, "Flow processing and the rise of commodity network hardware," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, p. 20–26, Mar. 2009.

[3] "ETSI Network Function Virtualization," http://www.etsi.org/technologies-clusters/technologies/nfv.

[4] "OPNFV," https://www.opnfv.org/.

[5] M.-A. Kourtis *et al.*, "T-nova: An open-source mano stack for nfv infrastructures," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 586–602, 2017.

[6] D. Dietrich *et al.*, "Multi-provider service chain embedding with nestor," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 91–105, 2017.

[7] X. Fu *et al.*, "Service function chain embedding for nfv-enabled iot based on deep reinforcement learning," *IEEE Communications Magazine*, vol. 57, no. 11, pp. 102–108, 2019.

[8] C. Papagianni *et al.*, "Rethinking service chain embedding for cellular network slicing," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 2018, pp. 1–9.

[9] A. Abujoda and P. Papadimitriou, "Distnse: Distributed network service embedding across multiple providers," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2016, pp. 1–8.

[10] D. Dietrich *et al.*, "Network function placement on virtualized cellular cores," in *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2017, pp. 259–266.

[11] D. Spatharakis *et al.*, "A scalable edge computing architecture enabling smart offloading for location based services," *Pervasive and Mobile Computing*, vol. 67, p. 101217, 2020.

[12] R. Panigrahy *et al.*, "Heuristics for vector bin packing," *research. microsoft. com*, 2011.

[13] S. Lee *et al.*, "Validating heuristics for virtual machines consolidation," *Microsoft Research, MSR-TR-2011-9*, pp. 1–14, 2011.

[14] R. Grandl *et al.*, "Multi-resource packing for cluster schedulers," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 455–466.

[15] Z. Xiao *et al.*, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 6, pp. 1107–1117, 2012.

[16] A. Pentelas *et al.*, "Network service embedding with multiple resource dimensions," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[17] A. Haider *et al.*, "Challenges in resource allocation in network virtualization," in *20th ITC Specialist Seminar*, vol. 18, no. 2009. ITC, 2009.

[18] A. Fischer *et al.*, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[19] M. Chowdhury *et al.*, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on networking*, vol. 20, no. 1, pp. 206–219, 2011.

[20] A. Gember *et al.*, "Stratos: Virtual middleboxes as first-class entities," 2012.

[21] Z. A. Qazi *et al.*, "Simple-fying middlebox policy enforcement using sdn," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4. ACM, 2013, pp. 27–38.

[22] M. Rost and S. Schmid, "On the hardness and inapproximability of virtual network embeddings," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 791–803, 2020.

[23] V. Vazirani, "Approximation algorithms springer-verlag," *New York*, 2001.

[24] F. Parreño *et al.*, "A hybrid grasp/vnd algorithm for two-and three-dimensional bin packing," *Annals of Operations Research*, vol. 179, no. 1, pp. 203–220, 2010.

[25] M. H. Ferdaus *et al.*, "Virtual machine consolidation in cloud data centers using aco metaheuristic," in *European conference on parallel processing*. Springer, 2014, pp. 306–317.

[26] "Microsoft Azure," https://azure.microsoft.com/.

[27] "Amazon Web Services," https://aws.amazon.com/.

[28] "Google Cloud," https://cloud.google.com/.

[29] "IBM Cloud," https://www.ibm.com/cloud.

[30] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2019. [Online]. Available: http://www.gurobi.com

[31] T. Benson *et al.*, "Cloudnaas: a cloud networking platform for enterprise applications," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 8.

[32] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[33] A. Ghodsi *et al.*, "Dominant resource fairness: Fair allocation of multiple resource types." in *Nsdi*, vol. 11, no. 2011, 2011, pp. 24–24.

Angelos Pentelas obtained a B.Sc. in Mathematics (2015) from the Aristotle University of Thessaloniki, Greece, and a M.Sc. in Applied Informatics (2019) from the University of Macedonia, Greece. Currently, he is pursuing his Ph.D. degree at the same department. His research interests include, among others, the application of optimization and machine-learning methods on NFV orchestration.

George Papathanail is a Ph.D. candidate at the Department of Applied Informatics in the University of Macedonia, Greece. He obtained a B.Sc. in Computer Engineer from the Technological Institute of Central Macedonia (Serres) in 2011, and a M.Sc. in Management and Information Systems from the University of Macedonia in 2017. His research interests include next-generation Networking, NFV, SDN, and Network Slicing.

Ioakeim Fotoglou is a Ph.D. candidate at the department of Applied Informatics in the University of Macedonia, Greece. He received a B.Sc. in Mathematics from the University of Crete in 2015, and a M.Sc. in Applied Informatics from the University of Macedonia in 2019. His research interests include SDN, NFV, and Network & Cloud Security.

Panagiotis Papadimitriou is an Assistant Professor at the department of Applied Informatics in the University of Macedonia, Greece. Before that, he was an Assistant Professor at the Communications Technology Institute of Leibniz Universität Hannover, Germany, and a member of L3S research center in Hanover. Panagiotis received a Ph.D. in Electrical and Computer Engineering from Democritus University of Thrace, Greece, in 2008, a M.Sc Information Technology from University of Nottingham, UK, in 2001, and a B.Sc. in Computer Science from University of Crete, Greece, in 2000. He has been (co-)PI in several EU-funded (*e.g.,* T-NOVA, CONFINE, NECOS) and nationally-funded projects (*e.g.,* G-Lab VirtuRAMA, MESON). Panagiotis was a recipient of Best Paper Awards at IFIP WWIC 2012, IFIP WWIC 2016, and the runner-up Poster Award at ACM SIGCOMM 2009. He has co-chaired several international conferences and workshops, such as INFOCOM SWFAN 2016, IFIP WWIC 2016, IEEE CNSM SR+SFC 2018–19, and IEEE NetSoft S4SI 2020. His research activities include (next-generation) Internet architectures, network processing, SDN, datacenter networking, and edge computing. Panagiotis is a Senior Member of IEEE.

European Union
European Regional Development Fund

HELLENIC REPUBLIC
MINISTRY OF ECONOMY & DEVELOPMENT
SPECIAL SECRETARY FOR ERDF & CF
MANAGING AUTHORITY OF EPAnEK

EPAnEK 2014-2020
OPERATIONAL PROGRAMME
COMPETITIVENESS
ENTREPRENEURSHIP
INNOVATION

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη
Partnership Agreement 2014 - 2020

Co-financed by Greece and the European Union