Effective NFV Orchestration for Wide-Ranging Services Across Heterogeneous Cloud Networks.

Bart Spinnewyn*, Juan Felipe Botero†, Carlos Donato*, and Steven Latré*
*University of Antwerp - imec, IDLab, Department of Mathematics and Computer Science,
Sint-Pietersvliet 7, 2000 Antwerp, Belgium
†Universidad de Antioquia, Department of Telecommunications Engineering, Calle 67 # 53 - 108, Medellín, Colombia

***Abstract*—In the Network Functions Virtualization (NFV) environment, to successfully orchestrate a network service, first a Virtual Network Function-Forwarding Graph (VNF-FG) must be composed that realizes the required functionality. Second, this VNF-FG must be embedded onto the infrastructure, that is increasingly becoming heterogeneous. To avoid wasting precious resources during orchestration, intelligent resource allocation mechanisms and algorithms are needed to effectively tailor the VNF-FG to the cloud network onto which the service will be deployed. In this paper, we introduce an improved service model supporting network services that have bidirectional chaining constraints, comprise optional VNFs, and require traffic aggregation. Building on this service model, we propose placement algorithms that can optimize the order and number of instances of VNFs, to adapt the VNF-FG to the availability of resources in the network. Numerical experiments show that through coordination of composition and embedding tasks, our proposed algorithms can significantly improve the acceptance ratio, compared to algorithms that perform these tasks in two separate stages.**

***Index Terms*—Heuristics, network function virtualization, optimization, orchestration, resource allocation.**

I. INTRODUCTION

Broadband services are increasingly being deployed in Network Functions Virtualization (NFV) environments. For instance, YouTube TV [1], DIRECTV [2] and Twitch [3] deliver real-time video streams to users world-wide. The cloud networks onto which these services are being deployed, are becoming more heterogeneous as services are increasingly composed of Virtual Network Functions (VNFs) that require specialized hardware. Moreover, cloud infrastructure is placed near the edge of the network [4], [5], so, in order to deploy these complex services on-demand, highly automated service orchestration algorithms are needed.

These orchestration algorithms realize two important functions [6], [7]. First, the orchestrator composes the Virtual Network Function-Forwarding Graph (VNF-FG) based on the Service Requirements (SRs). Second, it decides on how to embed these virtualized resources in the Substrate Network (SN). Current orchestration approaches often consider the VNF-FG precomposed. When they do consider the composition, they typically make very restrictive assumptions on the VNF-FG. For instance, they assume the VNF-FG to be a Service Function Chain (SFC) or a tree, which means that VNFs aggregating traffic, e.g., to generate video compositions or fuse sensor data in Wireless Sensor Networks (WSNs), are not supported. Further, state-of-the-art approaches compose the VNF-FG without any consideration for the SN's resources.

In this paper, we propose a more widely applicable service model that can produce VNF-FGs of the Directed Acyclic Graph (DAG) class, while considering more general chaining requirements and optional performance enhancing VNFs. Through our proposed optimal algorithm that performs both composition and embedding at the same time, we demonstrate the importance of considering the SN during composition. This consideration of the SN is especially important when optional VNFs are involved that can either ease or complicate the embedding. Further, we present two heuristics that can be used to orchestrate real-life services. One is an extension of an existing coordinated algorithm to support our generalized service model. The other is based on an existing embedding algorithm, that can produce solutions fast, and find better quality solutions when given more time.

II. RELATED WORK

In a recent survey, the resource allocation problem in NFV (NFV-RA) has been divided in three main challenges [7]. 1) The SFC composition challenge asks the following question: How to concatenate the different VNFs efficiently in order to compose a Network Service (NS) in the most adequate way, with respect to the operator goals?; 2) The embedding challenge that, after the SFC is composed, seeks to find where to efficiently allocate the VNFs in the network infrastructure accomplishing the Quality of Service (QoS) service constraints; and 3) the SFC scheduling that seeks to answer the question: How to execute each function in order to minimize the total execution time without degrading the service performance and respecting all the precedences and dependencies between the VNFs composing the NS?.

Most of the current work has been devoted to solve the second stage of the problem that is a specific variant of the VNE [8]. In [9], authors solve jointly the admission control and the VNF embedding challenges by proposing an Integer Linear Program (ILP) formulation that is solved using successive convex approximation methods. Haeri et al. focus on maximizing the provider revenue, they jointly perform Virtual Network Embedding (VNE) embedding and admission control using Monte Carlo Tree Search (MCTS) [10]. In [11], an ILP is proposed to formulate the SFC embedding problem where the objective function seeks for the minimization of the

978-3-903176-15-7 © 2019 IFIP

bandwidth requirements; the problem is solved in a scalable and exact way via a mathematical model with a decomposition scheme. The authors of [12] propose a fully automated approach to jointly optimizing scaling, placement, and routing VNF Requests (VNFRs) called JASPER; the approach is divided in two, an exact Mixed Integer Linear Program (MILP) approach and a custom heuristic.

While the aforementioned approaches do not consider the composition stage of the problem, there are some recent proposals that face this problem. Authors of [13] were the first to propose an exact ILP-based approach to solve the composition problem, even if this solution is not scalable it can be taken as a first approximation to solve this stage. The first proposal to solve both the composition and embedding stages was proposed in [14], here authors present an uncoordinated solution where the composition is solved via a greedy heuristic and the resulting SFC is embedded using an Mixed Integer Quadratically Constrained Program (MIQCP). The first approach to solve in a coordinated way SFC composition and embedding is presented in [15] using a recursive algorithm that concatenates VNFs and embeds them at the same time; the problem with this approach is the running time when the algorithm backtracks due to the impossibility to embed a VNF.

Compared to previous work [16], the approach proposed in this paper presents three major contributions: 1) a widely applicable service model with support for optional performance enhancing VNFs, bidirectional chaining constraints (e.g., for heterogeneous service delivery) and traffic aggregation (e.g., fusion of sensor data in WSNs); 2) an optimal formulation of the problem as a 1-stage ILP; which is based on a symmetry-removing Augmented Graph (AG); and 3) a heuristic based on MCTS to solve the problem in a scalable and near-optimal way.

III. Problem formulation

This section formally introduces the problem that is studied in this paper. The problem is to find a good quality initial Placement Configuration (PC), given SRs and the SN description. First, the composition and embedding requirements are introduced. Second, an AG is introduced to simplify formulation of the problem as an ILP. Third, the combined problem is formulated as a 1-stage ILP that optimizes the service composition and embedding at the same time. Finally, an ILP formulation is provided that can be used to perform VNF-FG composition and embedding in two separate stages. The problems introduced in this section require the embedding of a VNF-FG, which is an instance of the VNE problem and has been shown to be NP-hard [17]. Consequently, these problems are at least NP-hard. Throughout this paper, a media streaming service that distributes a sports event to multiple TV providers, is used to illustrate the concepts.

A. Composition and embedding requirements

The SRs are listed in Table I. The NS's functionality is realized by a VNF-FG. In this paper, the VNF-FG is considered a DAG. This VNF-FG comprises VNF instances, interconnected by VL instances. A path in this DAG, formed by an ordered set of VL instances, is referred to as a SFC. The VNF instances in the VNF-FG are created based on a set of VNFs (V). Each VNF $v \in V$ has a set of egress VLs (L^v_{out}) and ingress VLs (L^v_{in}). The processing requirements of an instance of $v \in V$ are proportional to the total ingress bandwidth to the instance, by a factor $d_{rel}(v)$.

TABLE I
Input parameters to the service composition.

Symbol	Description
V	Set of VNFs in the service.
L	Set of VLs in the service.
$L_{out} \subset L$	Set of egress VLs in the service.
$L_{in} \subset L$	Set of ingress VLs in the service.
$L^v_{out} \subset L_{out}$	Set of egress VLs of $v \in V$.
$L^v_{in} \subset L_{in}$	Set of ingress VLs to $v \in V$.
$V_{init} \subset V$	Initial VNFs where the service originates.
$V_{term} \subset V$	Set of terminating VNFs.
$r_{init}(v) : V_{init} \to \mathbb{R}^+$	Initial data rate arriving at initial VNF $v \in V_{init}$.
$\mathcal{C}(l_{out}, l_{in}) : L_{out} \times L_{in} \to \{0,1\}$	Binary parameter indicating if l_{in} and l_{out} are compatible.
$M(l_{out}) : L_{out} \to \mathbb{Z}^+$	Maximum number of VL instances flowing out of a VNF instance via l_{out}.
$L^{next}_{l_{out}} \subset L_{in}$	Set of ingress VLs that the traffic on $l_{out} \in L_{out}$ should pass through next.
$L^{prev}_{l_{in}} \subset L_{out}$	Set of egress VLs that the traffic on $l_{in} \in L_{in}$ should have passed through previously.
$d_{rel}(v) : V \to \mathbb{R}^+$	The ratio of the processing requirement of VNF v, to the ingress data rate at v.
$r_{rel}(l_{out}) : L_{out} \to \mathbb{R}^+$	The ratio of traffic flowing out $l_{out} \in L_{out}$, to the traffic generation rate of VNF $v \in V$.

An instance of VNF $v_1 \in V$ can communicate to an instance of VNF $v_2 \in V$ through a VL instance, that is formed by connecting an egress VL of the source VNF instance $l_{out} \in L^{v_1}_{out}$ to an ingress VL of the target VNF instance $l_{in} \in L^{v_2}_{in}$. l_{out} can be connected to l_{in} if and only if (iff) both VLs are compatible, i.e. $\mathcal{C}(l_{out}, l_{in}) = 1$.

The VNF-FG must contain exactly one VNF instance of each VNF $v_{init} \in V_{init}$ and VNF $v_{term} \in V_{term}$. An initial VNF $v_{init} \in V_{init}$ has no ingress VLs and generates traffic at a rate $r_{init}(v_{init})$. For an instance of a non-initial VNF $v \in V \setminus V_{init}$, the bandwidth on each VL instance corresponding to egress VL $l^v_{out} \in L^v_{out}$ is proportional, by a factor $r_{gen}(l_{out})$, to this VNF instance's total ingress bandwidth.

In the VNF-FG, the chaining requirements for an instance v^{inst} of VNF $v \in V$ are the following. On the one hand, there are requirements related to the neighborhood of v^{inst}. First, through each of its ingress VLs $l_{in} \in L^v_{in}$, v^{inst} must be connected to one parent. Second, v^{inst} can be connected to at most $M(l_{out})$ children, through each of its egress VLs $l_{out} \in L^v_{out}$. On the other hand, there are bidirectional chaining requirements related to v^{inst}. First, for any terminal VNF

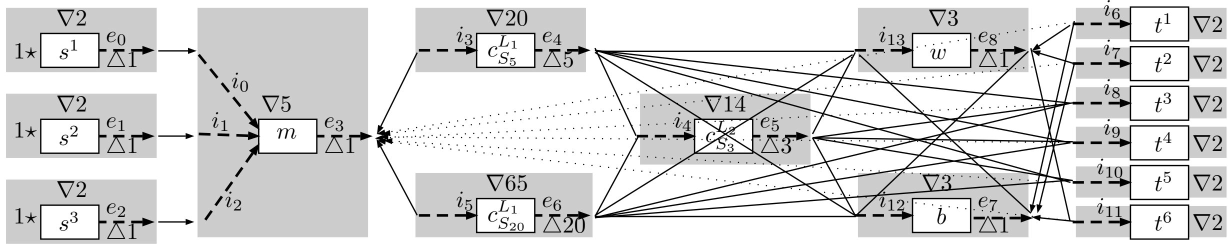


Fig. 1. Service requirements for the working example, grouped per VNF in gray rectangles. Arrows with intermittent lines whose label start with an i and an e, represent ingress and egress VLs respectively. A solid line connecting $l_{out} \in L_{out}$ and $l_{in} \in L_{in}$ indicates $\mathcal{C}_{l_{out},l_{in}} = 1$. An arrowhead on this same line from l_{out} to l_{in} indicates that $l_{in} \in L^{next}_{l_{out}}$. An opposite arrowhead indicates that $l_{out} \in L^{prev}_{l_{in}}$. For all $v \in V$, $r_{init}(v)^{\star}$ and $d_{rel}(v)^{\nabla}$ are indicated. Further, for all $l_{out} \in L_{out}$, $M(l_{out})^{\triangle}$ is indicated and $r_{rel}(l_{out}) = 1$.

instance that can be reached from v^{inst} through its egress VL $l_{out} \in L^{v}_{out}$, any SFC to this descendant must contain all ingress VLs in L^{next}_{v}. The set of required ingress VLs that any SFC from v^{inst} to any terminal VNF instance must contain, will be denoted as $N(v^{inst}) \subset L^{next}$. Then, for any child $\tilde{v}^{inst}$ of v^{inst} that is connected via l_{out} to l_{in}, $N(\tilde{v}^{inst}) = N(\tilde{v}^{inst}) \cup L^{next}_{l_{out}} \setminus \{l_{in}\}$. Since a terminal VNF has no egress VLs, each of its instances ${v_{term}}^{inst}$ must have resolved all dependencies of its ancestors, i.e. $N({v_{term}}^{inst}) = \emptyset$. Egress VLs that are not in L^{prev} are referred to as *optional*. It is assumed that optional VLs do not change the bandwidth requirements along an SFC, otherwise the level of service would depend on the VNF-FG composition. Further, for any initial VNF instance ${v_{init}}^{inst}$ that can reach v^{inst} through an SFC that contains $l_{in} \in L^{v}_{in}$, this SFC must contain all egress VLs in L^{prev}_{v}. The set of required egress VLs that any SFC, from any initial VNF instance towards v^{inst} contains, is denoted as $P(v^{inst}) \subset L^{prev}$. This set determines which children can be connected to v^{inst}.

For the working example, the chaining requirements are shown in Figure 1. This media distribution service makes a composition (VNF m) of three video sources (VNFs s_1, s_2, s_3). The composed stream can contain a combination of multiple viewing angles of the event and commentary from a TV studio. m aggregates the traffic flowing onto its ingress VLs, i.e., i_0, i_1, i_3. In contrast, aggregation of traffic is not supported in [13], [15], [16]. These ingress VLs each need to be connected to the egress VL of s_1, s_2 and s_3 respectively. The VNF-FG can only comprise a single instance of each initial VNF, and $M(e_0) = 1$. Thus, the service can contain only a single instance of m. The terminal VNFs, i.e., t^1 through t^6, each require a copy of the composed video stream from m. Additionally, these terminal VNFs can have their own specific service delivery requirements. For instance, the ingress VL to t^1 requires the traffic on its ingress VL, i.e. i_6, to have passed through the egress VL of a watermarking VNF (w), billing VNF (b) and composition VNF first. The watermarking VNF can add the TV provider's logo or advertise a popular upcoming program. The sequence in which this stream passes through b and w does not matter for the functionality of the service. Since $M(e_8) = 1$, only a single stream can

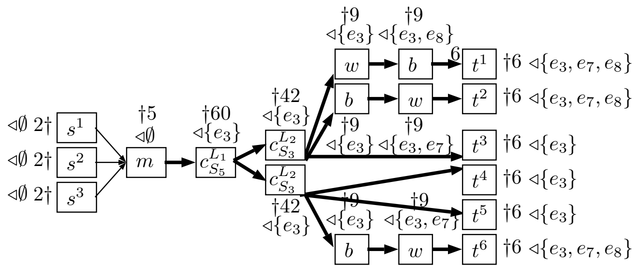


Fig. 2. Illustration of a valid VNF-FG for the working example with annotation of CPU requirements† and $P^{\triangleleft}$. N identically equals $\emptyset$. Thin and thick arrows represent VL instances with a bandwidth requirement of 1 and 3 bandwidth units respectively.

be processed by any instance of w. Therefore, each stream arriving at a terminal VNF has been uniquely watermarked. VL i_6 can be directly connected to e_8 or e_7 (solid line), but cannot be directly connected to e_3 (dotted line). While the ingress VL to t^1 does require watermarking and billing, the VL ingress to t^3, i.e. i_8, does not. Moreover, since i_8 is not compatible with e_7 and e_8, a SFC arriving at i_8 cannot have flown through w and b.

The distribution of the composed video stream from m towards the watermarking, billing and terminal VNFs is done by a hierarchical caching network. The first layer (L_1) contains either $c^{L_1}_{S_5}$ or $c^{L_1}_{S_{20}}$, neither one's inclusion in the VNF-FG is strictly required. The L_1 caches can serve watermarking, billing and terminal VNFs and layer 2 (L_2) caches. $c^{L_2}_{S_3}$ can serve up to 3 VNFs and is optional. While our model supports bidirectional chaining requirements and optional VLs, [13], [15], [16] consider only chaining requirements from the initial VNF towards the terminal VNFs and mandatory VLs.

A VNF-FG satisfying the chaining requirements of the working example is shown in Figure 2. The used caching hierarchy comprises one instance of $c^{L_1}_{S_5}$ and two instances of $c^{L_2}_{S_3}$. The total ingress bandwidth to the composition instance (m) equals $1 + 1 + 1 = 3$. Since $r_{rel}(e_3) = 1$, the egress bandwidth along the VL instance from m to $c^{L_1}_{S_5} = 3 \times 1$. For the initial VNF instances, N is empty. For each of the ingress VLs of m, the ingress VL instance immediately satisfies the connecting egress VL's chaining requirement. Consequently,

TABLE II
INPUT PARAMETERS TO THE SERVICE EMBEDDING.

Symbol	Description
H	Set of PMs.
E	Set of PLs.
$\phi(v) \subset H$	Set of PMs $\in H$ that can host VNF $v \in V$.
$D(h)$	Remaining processing capability of PM $h \in H$.
$B(h_1, h_2)$	Remaining bandwidth capability of $(h_1, h_2) \in E$.

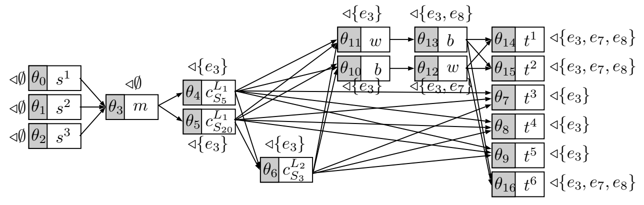
for this m instance, N is empty. Further, e_0, e_1 and e_2 are not L^{prev}, whence for this instance of m, P is empty as well. e_3 however is a required egress VL, therefore it is added to P for all of its descendants. The same goes for e_7 and e_8.

The input parameters to the service embedding are listed in Table II. The SN comprises a set of Physical Machines (PMs) (H), interconnected by directed Physical Links (PLs) (E). The set of PMs on which a particular VNF $v \in V$ can be executed is given by $\Phi(v)$. The remaining processing capability of $h \in H$ is given by $D(h)$. The remaining bandwidth capacity of $(h_1, h_2) \in E$ is given by $B(h_1, h_2)$.

B. Augmented Graph

The VNF-FG in Figure 2 comprises three instances of VNF w. The two w instances that are children of a b instance, have identical chaining, bandwidth and processing requirements. These two instances will be called *equivalent*. Given the SRs in Section III-A, two VNF instances, v_1^{inst} and v_2^{inst} are equivalent iff they (1) correspond to the same VNF $v \in V$; (2) have identical ingress VL dependencies (N) after the VNF instance and the same required egress VLs (P) appear before the VNF instance; and (3) they have the same bandwidth and processing requirements. This equivalence relation is denoted as $v_1^{inst} \sim v_2^{inst}$. Since optional VNFs are assumed to not change the bandwidth requirements and the bandwidth requirements are multiplicative; condition (3) follows from (1) and (2). This property will be used in the AG, which serves as a basis to describe any VNF-FG satisfying the SRs. This AG $G(\Theta, \Psi)$, comprises augmented VNFs Θ, interconnected by augmented VLs Ψ, to represent VNF instances and their possible interconnections. By construction, these augmented nodes and augmented VLs satisfy the chaining requirements.

The procedure to generate the AG comprises four steps. First, the augmented VNFs corresponding to the initial VNFs are generated and added to Θ. For these initial augmented VNFs, both N and P are empty. Then, these augmented VNFs are added to a queue (Q_1). In the second step, the egress VLs of the augmented VNFs in Q_1 are explored. When an egress VL l_{out} of θ_{cur} in Q_1 can be connected to a VNF's ingress VL l_{in}, then a new augmented VNF instance is created (θ_{new}). This connection can be made iff l_{out} and l_{in} are compatible; $P(\theta_{cur}) \cup l_{out}$ contains all egress VLs in $L_{l_{in}}^{prev}$; and the target VNF of l_{in} is not an ancestor of θ_{cur}. If θ_{new} is not in Θ, then it is added to both Θ and Q_1, as its egress VLs must be explored. Finally, $(\theta_{cur}, \theta_{new})$ is added to Ψ if it is not already in Ψ. The third step prunes invalid augmented VNFs from the AG. In this step there can be two



$\triangleleft\emptyset$ θ_0 s^1
$\triangleleft\emptyset$ θ_1 s^2
$\triangleleft\emptyset$ θ_2 s^3
$\triangleleft\emptyset$ θ_3 m
$\triangleleft\{e_3\}$ θ_4 $c_{S_5}^{L_1}$
$\triangleleft\{e_3\}$ θ_5 $c_{S_{20}}^{L_1}$
$\triangleleft\{e_3\}$ θ_6 $c_{S_3}^{L_2}$
$\triangleleft\{e_3\}$ θ_{11} w
$\triangleleft\{e_3\}$ θ_{10} b
$\triangleleft\{e_3, e_8\}$ θ_{13} b
$\triangleleft\{e_3, e_7\}$ θ_{12} w
θ_{14} t^1 $\triangleleft\{e_3, e_7, e_8\}$
θ_{15} t^2 $\triangleleft\{e_3, e_7, e_8\}$
θ_7 t^3 $\triangleleft\{e_3\}$
θ_8 t^4 $\triangleleft\{e_3\}$
θ_9 t^5 $\triangleleft\{e_3\}$
θ_{16} t^6 $\triangleleft\{e_3, e_7, e_8\}$

Fig. 3. AG for the working example. Empty values of P and N are not shown.

TABLE III
INPUT PARAMETERS TO THE THE SERVICE COMPOSITION FOR THE ILP.

Symbol	Description
Θ	Set of augmented VNFs in the AG.
Ψ	Set of augmented VLs in the AG.
$v(\theta) : \Theta \to V$	VNF corresponding to $\theta \in \Theta$.
$\Psi^{out}(\theta, l_{out}) : \Theta \times L_{out} \to \Psi$	Set of egress augmented VLs of $\theta \in \Theta$, corresponding to $l_{out} \in L_{out}^{v(\theta_1)}$.
$\Psi^{in}(\theta, l_{in}) : \Theta \times L_{in} \to \Psi$	Set of ingress augmented VLs of $\theta \in \Theta$, corresponding to $l_{in} \in L_{in}^{v(\theta)}$.
$b(\psi) : \Psi \to \mathbb{R}^+$	Bandwidth requirement of augmented VL $\psi \in \Psi$
$d(\theta) : \Theta \to \mathbb{R}^+$	The processing requirement of augmented VNF $\theta \in \Theta$.

types of invalid augmented VNFs in the AG. First, terminal augmented VNFs with remaining dependencies ($N(\theta) \neq \emptyset$) are removed. Second, augmented VNFs with missing parents are removed. When a node is removed from Θ, then the validity of its former neighbours in the AG is verified as well. This step is also implemented with a queue. Forth, the bandwidth and processing requirements of the augmented VNFs are calculated recursively. For the working example, the resulting AG is shown in Figure 3.

C. 1-Stage Integer Linear Program (ILP1STAGE)

In this section, we provide a formal description of our problem as an ILP [18], which can be used to find an optimal solution to the combined problem of service composition and embedding for services with bidirectional chaining requirements and optional VLs. To the best of our knowledge, we are the first to solve this combined problem in an optimal way. Table III lists the input parameters related to the VNF chains and service composition. These parameters result from the AG generation algorithm described in Section III-B. The input parameters related to the service embedding are the same as for the original problem (Table II). The decision variables are listed in Table IV.

Constraints: The augmented VNFs corresponding to the terminal VNFs are selected.

$$X_\theta = 1 : \forall \theta \in \Theta | v(\theta) \in V_{init} \cup V_{term} \tag{1}$$

The number of embedded VNF instances equals the number of VNF instances in the VNF-FG.

$$X_\theta = \sum_{h \in H} x_{\theta,h} : \forall \theta \in \Theta \tag{2}$$

TABLE IV
DECISION VARIABLES OF THE ILP.

Symbol	Description
X_θ	Integer indicating how many instances of augmented VNF $\theta \in \Theta$ are in the VNF-FG.
Y_ψ	Integer indicating how many instances of augmented VL $\psi \in \Psi$ are in the VNF-FG.
$x_{\theta,n}$	Integer indicating how many instances of augmented VNF $\theta \in \Theta$ are embedded onto PM n.
$y_{\psi,(h_1,h_2)}$	Integer indicating the number of VL instances corresponding to augmented link $\psi \in \Psi$ that are routed along $(h_1,h_2) \in E$.
$\sigma_{\psi,h}$	Integer indicating how many instances of augmented VL $\psi \in \Psi$ originate at $h \in H$.
$\tau_{\psi,h}$	Integer indicating how many instances of augmented VL $\psi \in \Psi$ terminate at $h \in H$.

For each augmented VNF instance on a node, the corresponding ingress augmented VL instances are provisioned.

$$\sum_{\psi_{in} \in \Psi^{in}(\theta,l_{in})} \tau_{\psi_{in},h} = x_{\theta,h} : \forall h \in H, \theta \in \Theta, l_{in} \in L_{in}^{v(\theta)} \tag{3}$$

For each egress VL $l_{out} \in L_{out}^v$ of an instance of VNF $v \in V$, the maximum number of corresponding egress VL instances is $M(l_{out})$. $\forall \theta \in \Theta, l_{out} \in L_{out}^{v(\theta)}, h \in H$:

$$\sum_{\psi_{out} \in \Psi^{out}(\theta,l_{out})} \sigma_{\psi_{out},h} \leq M(l_{out}) x_{\theta,h} \tag{4}$$

The processing requirements for the VNFs are assumed additive. If an augmented VNF θ is embedded onto $h \in H$, then its processing requirements $d(\theta)$ are allocated. For each PM $h \in H$, the total processing requirements cannot exceed the node's remaining capabilities.

$$\sum_{\theta \in \Theta} d(\theta) x_{\theta,h} \leq D(h) : \forall h \in H \tag{5}$$

VNF $v \in V$ cannot be instantiated on PMs that are not in its candidate set $\phi(v) \subset H$.

$$x_{\theta,h} = 0 : \forall \theta \in \Theta, h \in H \setminus \phi(v(\theta)) \tag{6}$$

All VL instances in the VNF-FG are routed through the SN. Multi-commodity flow constraints dictate that for any augmented VL $(\theta_1,\theta_2) \in \Psi$ the net number of VL instances leaving a certain PM h_1 equals the difference between the number of instances of θ_1 and θ_2 hosted on h_1. $\forall h_1 \in H, (\theta_1,\theta_2) \in \Psi$:

$$\sum_{(h_1,h_2) \in E} y_{(\theta_1,\theta_2),(h_1,h_2)} - \sum_{(h_2,h_1) \in E} y_{(\theta_1,\theta_2),(h_2,h_1)} = \sigma_{(\theta_1,\theta_2),h_1} - \tau_{(\theta_1,\theta_2),h_1}. \tag{7}$$

If augmented VL $\psi \in \Psi$ is routed along PL $(h_1,h_2) \in E$, then its required bandwidth $b(\psi)$ is allocated. The total bandwidth consumption on a PL cannot exceed the available bandwidth capability $B(h_1,h_2)$.

$$\sum_{\psi \in \Psi} b(\psi) y_{\psi,(h_1,h_2)} \leq B(h_1,h_2) : \forall (h_1,h_2) \in E \tag{8}$$

Objective function: The objective of the resource allocation is to minimize resources consumption, weighted by the scarcity of said resource.

$$\mathbb{L} = \sum_{\theta \in \Theta} \sum_{h \in H} \frac{d(\theta) x_{\theta,h}}{D(h)} + \sum_{\psi \in \Psi} \sum_{(h_1,h_2) \in E} \frac{b(\psi) y_{\psi,(h_1,h_2)}}{B(h_1,h_2)} \tag{9}$$

The objective is to minimize $\mathbb{L}$ subject to Equations 1 - 8.

D. 2-Stage Integer Linear Program (ILP2STAGE)

In this formulation, the optimization is performed in two stages. First, the VNF-FG is composed based on the service requirements, without any knowledge about the SN capabilities. During composition the number of instances of each augmented VNF and augmented VL is determined. Where in the SN, these virtualized resources are embedded is not yet considered. In a second stage, this VNF-FG is embedded onto the SN, taking into account the remaining capabilities. This algorithm is an improvement on related work that performs composition and embedding in two separate stages. For instance, Ocampo et al. solely consider the composition problem for VNF-FGs with a tree topology and without optional VLs. The authors try to find the minimal bandwidth VNF-FG [13]. The authors in [19], [20], [10], [21] assume a precomposed VNF-FG and solely focus on the VNE. The authors in [14] propose a 2-stage approach to the combined problem. In their first stage, a greedy algorithm composes the minimal bandwidth VNF-FG. In their second stage, the VNF-FG is embedded using a MIQCP.

Composition: The terminal and initial VNFs are instantiated exactly once (Equation 1). Further, each augmented VNF instance requires exactly one ingress VL instances corresponding to each of its ingress VLs in $L_{in}^{v(\theta)}$.

$$\sum_{\psi_{in} \in \Psi^{in}(\theta,l_{in})} I_{\psi_{in}} = X_\theta : \theta \in \Theta, l_{in} \in L_{in}^{v(\theta)}, \tag{10}$$

where I_ψ is the number of instances of augmented VL $\psi \in \Psi$. For each instance of $v \in V$, the number of egress VL instances for $l_{out} \in L_{out}^v$ is at most $M(l_{out})$. $\forall \theta \in \Theta, l_{out} \in L_{out}^{v(\theta)}$:

$$\sum_{\psi_{out} \in \Psi^{out}(\theta,l_{out})} I_{\psi_{out}} \leq M(l_{out}) X_\theta \tag{11}$$

ILP2STAGE-$\mathbb{B}$ minimizes the VNF-FG VL bandwidth in this stage. The same objective is used in [14] and [13].

$$\mathbb{B} = \sum_{\psi \in \Psi} b(\psi) I_\psi \tag{12}$$

ILP2STAGE-$\mathbb{C}$ minimizes the processing requirements.

$$\mathbb{C} = \sum_{\theta \in \Theta} d(\theta) X_\theta \tag{13}$$

Embedding: The resulting values of I_ψ, X_θ from the first stage are added as constraints. Further, for each augmented VL its number of instances equals the times that the augmented VL originates and terminates on any PM in H.

$$I_\psi = \sum_{h \in H} \sigma_{\psi,h} : \forall \psi \in \Psi \tag{14}$$

and

$$I_\psi = \sum_{h \in H} \tau_{\psi,h} : \forall \psi \in \Psi \tag{15}$$

Finally, the embedding is subject to the constraints in ILP1STAGE and minimizes $\mathbb{L}$ (Equation 9).

IV. Solution strategy

Given the computational intractability of the exact algorithms presented in the previous section, two heuristics are proposed. REC is a recursive procedure that terminates as soon as a valid mapping is found, while MCTS can find a better quality solution when given more computation time. Both procedures use the GENNEIGH and FINISHED functions.

GENNEIGH($\mathcal{M}, \kappa$): generates the (VL, Physical Path (PP)) mappings that can be added to the current mapping $\mathcal{M}$, i.e. a list of augmented VL to PP mappings. This function performs the following steps. First, the VNF instances in $\mathcal{M}$ are traversed. As soon as an instance of $v \in V$ with missing parents via $l_{in} \in L^v_{in}$ is encountered, the search for a missing parent stops. Now, the candidate VL instances to resolve this missing parent are selected from the AG. A look-ahead mechanism that verifies Equations 1, 10 and 11, is used to filter these candidates based on the current number of instances of each augmented VNF and VL in $\mathcal{M}$. Then, GENNEIGH generates candidate PPs targeted to the PM on which this VNF instance with missing parents is hosted, for each of these candidate ingress VLs. The candidate PPs are generated using a Breadth First Search (BFS) procedure that searches the κ closest PMs that can host the source VNF of this VL instance, while considering remaining bandwidth and processing capabilities. When no VNF instance in $\mathcal{M}$ is missing a parent, then the next missing terminal VNF $v_{term} \in V_{term}$ must be added. In this case, the candidate PMs in $\psi(v_{term})$ that have sufficient processing capability remaining to host this VNF, are selected.

FINISHED($\mathcal{M}$): If the mapping ($\mathcal{M}$) is not missing any ingress VL or terminal VNF instance, then this mapping satisfies the SRs and the function returns TRUE. Otherwise FINISHED returns FALSE.

A. Recursive Heuristic (REC)

The recursive heuristic is described in Algorithm 1, it is an improvement on the recursive heuristic by Beck et al. [15], which does not consider optional VLs, traffic aggregation and bidirectional chaining requirements. This heuristic decides at the same time which VL instance to add to the composition and how to embed it. Parameter α limits the maximum number of backtracks that the algorithm can make. Upon the algorithm's initiation, the number of backtracks that have been performed (a), is set to 0 (Line 2) and the mapping ($\mathcal{M}$) is empty (Line 3).

REC($\mathcal{M}, \alpha$) tries to find a valid mapping by extending $\mathcal{M}$. If REC($\mathcal{M}, \alpha$) finds a valid mapping, then it returns this mapping (Line 6). An empty set is returned if the backtracking limit

Algorithm 1 Recursive heuristic.

```
1: var $G(\theta,\Psi), G(H,E), B, D, \kappa, a$
2: $a \leftarrow 0$
3: $\mathcal{M} \leftarrow []$
4: procedure REC($\mathcal{M}, \alpha$)
5:     if FINISHED($\mathcal{M}$) then
6:         return $\mathcal{M}$                          ▷ Success
7:     else if $a > \alpha$ then
8:         return $\emptyset$                ▷ Backtrack limit exceeded
9:     end if
10:    $C \leftarrow$ GENNEIGH($\mathcal{M}, \kappa$)          ▷ Generate candidates
11:    for each $c \in C$ do
12:        $\mathcal{M}' \leftarrow [\mathcal{M}, c]$            ▷ Add $c$ to the mapping
13:        $\mathcal{M}'' \leftarrow$ REC($\mathcal{M}', \alpha$)                ▷ Recurse
14:        if $\mathcal{M}'' \neq \emptyset$ then
15:            return $\mathcal{M}''$            ▷ Sucessful recursion
16:        end if
17:        $a \leftarrow a + 1$                          ▷ Backtrack
18:    end for
19:    return $\emptyset$                              ▷ Failure
20: end procedure
```

Algorithm 2 Monte Carlo Tree Search.

```
1: var $G(\theta,\Psi), G(H,E), B, D, \kappa$
2: procedure MCTS($\beta$)
3:     $\mathcal{M} \leftarrow []$
4:     while $\neg$ FINISHED($\mathcal{M}$) do
5:         $C \leftarrow$ GENNEIGH($\mathcal{M}, \kappa$)
6:         $c \leftarrow$ SELECT($\mathcal{M}, C, \beta$)
7:         if $c = \emptyset$ then
8:             return $\emptyset$                        ▷ Failure
9:         end if
10:        $\mathcal{M} \leftarrow [\mathcal{M}, c]$
11:    end while
12:    return $\mathcal{M}$                              ▷ Success
13: end procedure
```

has been exceeded (Line 8), or if a valid mapping cannot be found by extending the current mapping ($\mathcal{M}$) (Line 19).

If the current mapping ($\mathcal{M}$) does not satisfy all SRs and the backtrack limit has not been exceeded, then the algorithm iterates over all possible candidates $c \in C$. In each iteration, c is added to $\mathcal{M}'$, i.e. a copy of $\mathcal{M}$ (Line 12). Next, the algorithm recursively searches for a valid mapping by extending $\mathcal{M}'$ (Line 13). If the resulting mapping ($\mathcal{M}''$) is valid, then this mapping is returned. Otherwise, the number of backtracks is incremented and the next candidate in C is tried. If all candidates have been exhausted, then the algorithm returns $\emptyset$.

B. Monte Carlo Tree Search (MCTS)

The MCTS algorithm is described in Algorithm 2. The search procedure is based on the VNE algorithms presented by Haeri et al. [10]. The key difference is that these algorithms do not solve the composition problem. The MCTS is parallelized using root parallelization; multiple independent searches are performed in separate threads and in the end the mapping with the highest reward is selected. For each of these independent searches, initially the mapping ($\mathcal{M}$) is empty (Line 3). Then, while this mapping is not complete, the procedure performs the following steps. The possible candidates (C) to add to the mapping ($\mathcal{M}$) are generated (Line 5). The SELECT procedure selects the next candidate ($c \in C$) to be added to the current mapping ($\mathcal{M}$) as follows. It estimates the reward of the candidates in C, by calling SIMULATE($\mathcal{M}, C$) up to β times. SIMULATE keeps track of the total reward and visit count of each candidate in C. In each call of SIMULATE, a candidate in C is selected using Upper Confidence Bounds for Trees (UCT) selection, which considers the total reward and visit count of each candidate. If $[\mathcal{M}, c]$ cannot be extended, then the reward is calculated and the reward and visit count of c is updated. Otherwise, a rollout is performed on $[\mathcal{M}, c]$ using the default rollout policy. ROLLOUT(M_{tmp}) iteratively generates a random next candidate by calling GENNEIGH($M_{tmp}, \mathcal{N}, \kappa$) and selecting a random candidate, and adding it to $\mathcal{M}_{tmp}$, until either a feasible mapping is found, or one that cannot be extended any further. When SIMULATE returns, the estimated reward of c is updated using the reward of this simulation. The reward equals $-\Gamma$ if the resulting mapping ($\mathcal{M}_{tmp}$) was invalid, where Γ is the penalty for a rejected request, with default value 10. If $\mathcal{M}_{tmp}$ was valid, then the reward equals minus the embedding cost. When the computational budget β is exhausted, SELECT returns the candidate in C with the highest average reward, in case its average reward exceeds $-\Gamma$. This candidate (c) is then added to $\mathcal{M}$ and MCTS proceeds to the next missing VL or VNF, when required (Line 4). In case all simulations were failures, then the request is rejected (Line 8). Finally, when no additional VNFs are required, then the request is accepted (Line 12).

V. Performance Evaluation

A. Simulation Setup

The SRs are the same as in the working example. Requests arrival and service-rates are modeled as Poisson processes. The arrival-rate (λ) equals 1 arrival per unit of time. The average duration of a request is the inverse of the service rate (μ).

Transit-stub SNs are generated using the GT-ITM topology generator [22]. Any two nodes in the transit network are connected by a PL with a probability of 80%. Within a stub-network cluster this probability is 40%. Each PM in the transit network is connected to 2 clusters, each comprising 9 PMs. In the experiments, the number of transit nodes is 4, corresponding to 76 PMs. There are 5 types of PM nodes, modeling the heterogeneity in nodal capabilities. First, 12 of these PMs can host *source* VNFs, i.e., s^1, s^2, or s^3. Further, 24 PMs can host *target* VNFs, i.e., t^1 through t^6. Next, 12 general purpose PMs can host *cpu* VNFs, i.e., caching VNFs and *b*. Further, 16 PMs are essentially *switches*, i.e. these PMs cannot host any VNFs. Finally, 12 PMs can host VNFs that require a dedicated *gpu*, i.e., m and w.

For each parameter configuration, 10 random SNs are generated using GT-ITM, each with a random assignment of PM types. For a given SN, the location of the source and target VNFs of different requests are independent of one other. The PL bandwidth capabilities all equal B_{def}. Processing capabilities are chosen that yield practically useful acceptance ratios: for cpu and gpu PM they equal 1500. The capabilities of source, target and switch PMs equal 3000, 2000 and 1000 respectively. Following configuration is used for the heuristics, balancing computation time and placement quality: $\kappa = 3$ and $\alpha = 10000$. The ILPs were solved using Gurobi 7.5.2. [23].

Following metrics are evaluated: 1) the *acceptance ratio*, i.e. the fraction of requests that is accepted; 2) the SN *bandwidth* and CPU *processing consumption* per accepted request; and 3) the *computation time* to process a request.

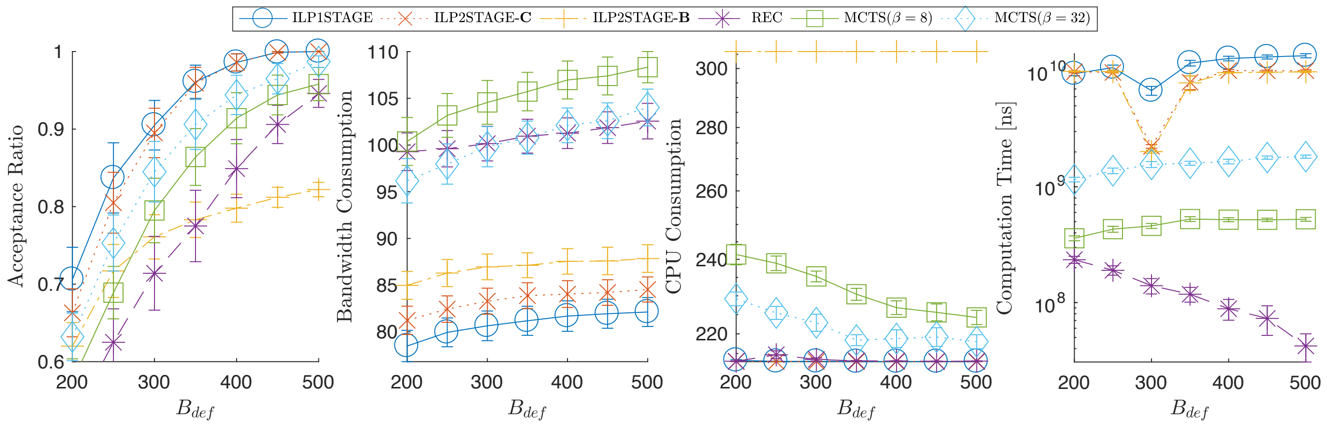
B. Results

Offline: In the offline scenario, the request duration approaches ∞, or $\mu \to 0$. Hence, all requests are active at the same time. In total 100 requests are processed per SN. The impact of the PL bandwidth capability B_{def} is shown in Figure 4. Overall, the acceptance ratio goes up as B_{def} increases and it is the highest for ILP1STAGE. It is the lowest for ILP2STAGE-$\mathbf{B}$ when B_{def} exceeds 350. For lower B_{def} values, ILP2STAGE-$\mathbf{B}$ outperforms REC and MCTS($\beta = 8$), due to better coordination between PM mapping and PL routing. This algorithm always composes the VNF-FG with the highest processing requirements, since it comprises a $c^{L_1}_{S_{20}}$ instance. The acceptance ratio for ILP2STAGE-$\mathbf{B}$ is up to 18% lower than for ILP1STAGE. In this scenario, minimizing the VNF-FG bandwidth is a poor composition goal. First, its performance is limited by its high processing requirements. Second, the minimal bandwidth VNF-FG does not lead to minimal SN bandwidth consumption. ILP2STAGE-$\mathbf{B}$ requires up to 8% more SN bandwidth than ILP1STAGE.

Compared to ILP1STAGE, ILP2STAGE-$\mathbf{C}$ consumes up to 3% more bandwidth. ILP1STAGE can place up to 7% more requests, by sporadically placing additional $c^{L_2}_{S_3}$ instances and by optimizing the order in which the traffic flows through the watermarking and billing VNFs for each target VNF. On average, these additional $c^{L_2}_{S_3}$ instances increase the processing consumption of accepted requests by only 0.14%. The required computation time for ILP1STAGE is up to 4 times higher than for ILP2STAGE-$\mathbf{B}$ and ILP2STAGE-$\mathbf{C}$.

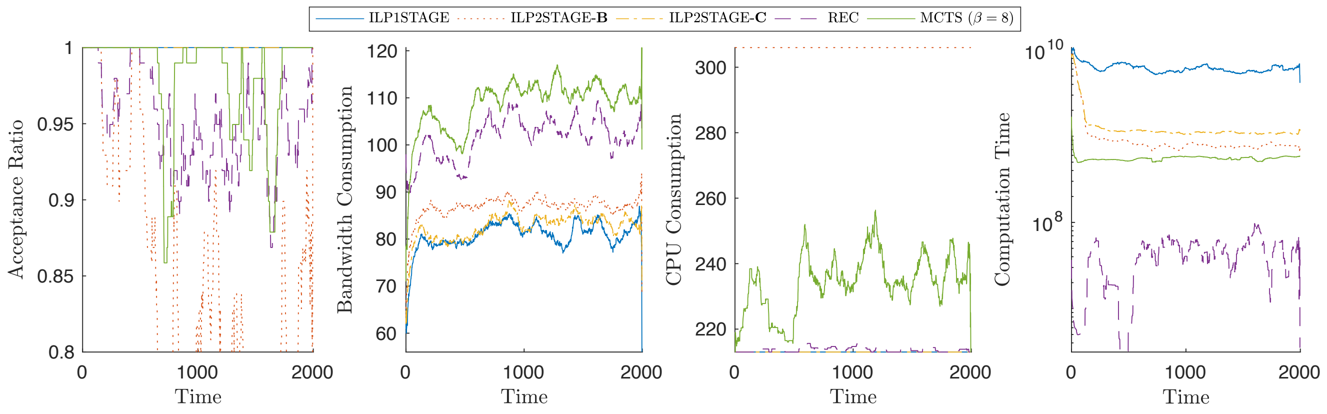
For higher bandwidth values, the acceptance ratio of MCTS approaches that of ILP1STAGE. As B_{def} increases from 200 to 500, the computation time for MCTS increases slightly, while the computation time for REC decreases by a factor 6. The reason for this computation time decrease as the B_{def} increases, is that REC requires significantly fewer backtracks to find a feasible mapping and terminates as soon as a feasible solution is found. When the computational budget β is increased, the acceptance ratio of MCTS approaches that of ILP1STAGE, and the placement cost decreases. As β increases from 8 to 32, the SN bandwidth consumption per requests decreases by up to 5%. For instance for $B_{def} = 450$ and $\beta = 8$, the acceptance ratio of MCTS is 4% lower than for ILP1STAGE, while being 26 times faster. For $\beta = 32$, the acceptance ratio of MCTS is 4% lower than for ILP1STAGE, while being 8 times faster. The acceptance ratio of REC is 9% lower than for ILP1STAGE, while requiring 333 times less computation time.

Whether a VNF-FG with minimal VL bandwidth or processing requirements is easiest to embed, or which default composition strategy performs best depends strongly on both the



(a) Average acceptance. (b) Bandwidth consumption. (c) CPU consumption. (d) Average computation time.

Fig. 4. Influence of the PL bandwidth capability B_{def}, offline, 4 threads.



(a) Acceptance trace. (b) Bandwidth consumption trace. (c) CPU consumption trace. (d) Computation time trace.

Fig. 5. Online traces for $B_{def} = 450$, $\mu = 0.01$ processed requests per unit of time.

application and cloud environment. Hence, the performance of a 2-stage approach with a given objective in the composition stage is strongly dependent on the SN conditions. Further, REC explores multiple compositions only when it backtracks. Hence, compared to ILP1STAGE and MCTS, its computation time and placement quality is strongly dependent on the order in which the candidates in GENNEIGH are ordered. In comparison, ILP1STAGE always composes the VNF-FG that is easiest to embed and MCTS converges to this composition when provided more computation budget.

Online: The online traces are shown in Figure 5. In the online scenario, the traces are smoothed by a moving average with a window of size 100 requests. Since $\mu = 0.01$ processed requests per unit of time, when all requests can be accepted, in steady state the expected number of requests active at the same is 100. Initially, all algorithms can place the incoming requests. Then, the acceptance ratio decreases first for ILP2STAGE-**B** and REC. The bandwidth consumption is the highest for MCTS, followed by REC, ILP2STAGE-**B**, ILP2STAGE-**C** and ILP1STAGE. The CPU consumption is the highest for ILP2STAGE-**B**. The computation time for REC shows a lot of variation. When finding a feasible placement is easy, then the computation time is very low. As finding a valid placement gets harder, its computation time increases and is ultimately limited by the backtracking limit. Both ILP1STAGE and ILP2STAGE-**C** can place all requests. Compared to the other algorithms, the computation time for MCTS is rather insensitive to the SN loading.

VI. CONCLUSIONS

In this paper, we focused on the resource allocation challenges related to the orchestration of NSs in NFV environments. We introduced a service model with improved applicability, that can describe a richer class of VNF-FGs. We proposed an optimal placement algorithm that can adapt the VNF-FG to the availability of resources in the SN, while considering the SRs. We demonstrated that our optimal algorithm can improve the acceptance ratio by up to 18% while requiring only 4 times more computation time, compared to algorithms that do not coordinate composition and embedding. Our proposed approach can find the best PC for any cloud environment. Further, we presented two heuristics that can find near-optimal solutions up to 26 and 330 times faster than the optimal algorithm.

Acknowledgment

This work was supported by the 5Guards and FUSE projects, both are realized in collaboration with imec, with project support from VLAIO (Flanders Innovation & Entrepreneurship). Project partners in FUSE are Barco, Axians and e-BO Enterprises. Project partners in 5Guards are Acceleran, Orange and Rombit.

References

[1] "Youtube tv - watch & dvr live sports, shows & news," https://tv.youtube.com/welcome/, (Accessed on 09/10/2018).

[2] "Directv official site," https://www.directv.com/, (Accessed on 09/10/2018).

[3] "Twitch," https://www.twitch.tv/, (Accessed on 09/10/2018).

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[6] "5g architecture white paper," 5GPPP Architecture Working Group, Tech. Rep., 12 2017.

[7] 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, Sept 2016.

[8] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[9] M. A. T. Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "vspace: Vnf simultaneous placement, admission control and embedding," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 542–557, March 2018.

[10] S. Haeri and L. Trajkovi, "Virtual network embedding via monte carlo tree search," *IEEE Transactions on Cybernetics*, vol. 48, no. 2, pp. 510–521, Feb 2018.

[11] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1320–1333, June 2018.

[12] S. Dräxler, H. Karl, and Z. Á. Mann, "JASPER: joint optimization of scaling, placement, and routing of virtual network services," *CoRR*, vol. abs/1711.10839, 2017. [Online]. Available: http://arxiv.org/abs/1711.10839

[13] A. F. Ocampo, J. Gil-Herrera, P. H. Isolani, M. C. Neves, J. F. Botero, S. Latré, L. Zambenedetti, M. P. Barcellos, and L. P. Gaspary, "Optimal service function chain composition in network functions virtualization," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2017, pp. 62–76.

[14] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 7–13.

[15] M. T. Beck and J. F. Botero, "Scalable and coordinated allocation of service function chains," *Computer Communications*, vol. 102, no. Supplement C, pp. 78 – 88, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366416303577

[16] B. Spinnewyn, P. H. Isolani, C. Donato, J. F. Botero, and S. Latré, "Coordinated service composition and embedding of 5g location-constrained network functions," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2018.

[17] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213 – 220, 2016, iNOC 2015 7th International Network Optimization Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571065316300336

[18] A. Schrijver, *"Theory of linear and integer programming"*, ser. Interscience in discrete mathematics and optimization. Wiley, New York, 1986.

[19] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*. ACM, 2009, pp. 81–88.

[20] M. Rost and S. Schmid, "Service chain and virtual network embeddings: Approximations using randomized rounding," *arXiv preprint arXiv:1604.02180*, 2016.

[21] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Network and Service Management (CNSM), 2015 11th International Conference on*. IEEE, 2015, pp. 50–56.

[22] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of IEEE INFOCOM '96. Conference on Computer Communications*, ser. INFOCOM'96, vol. 2. Washington, DC, USA: IEEE Computer Society, 1996, pp. 594–602. [Online]. Available: http://dl.acm.org/citation.cfm?id=1895868.1895900

[23] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com