CCVP: Cost-efficient Centrality-based VNF Placement and Chaining Algorithm for Network Service Provisioning

Shohreh Ahvar, Hnin Pann Phyu,
Sachham Man Buddhacharya, Ehsan Ahvar and Noel Crespi
Telecom SudParis, Evry, France
Email: {shohreh.ahvar, hnin.pann_phyu}@telecom-sudparis.eu
{buddhacharya, ehsan.ahvar, noel.crespi}@telecom-sudparis.eu

Roch Glitho
Telecommunications Service Engineering Lab
Concordia University
Montreal, Canada
Email: glitho@ece.concordia.ca

***Abstract*—Network services have been significantly increased in today's enterprise networks. The time and cost of deploying these services are recently considered as critical challenges for enterprise networks. Network Functions Virtualization (NFV) is a promising solution to offer cost-efficient, scalable and more rapid deployment of such services. It allows the implementation of fine-grained services as a chain of Virtual Network Functions (VNFs). These chains need to be placed in the network. The chain placement is critical since it effects on both quality of service (QoS) and the provider cost. This paper formulates the problem of VNF placement and chaining as an Integer Linear Program (ILP) and proposes a Cost-efficient Centrality-based VNF Placement and chaining algorithm (CCVP). The objective is to find the optimal number of VNFs along with their locations in such a manner that the provider cost is minimized. Apart from cost minimization, the support for large-scale environments with a large number of servers and end-users is an important feature of the proposed algorithm. Finally, the algorithm behavior is analyzed through simulations.**

I. Introduction

Computer network services are nowadays considered as a key component in keeping the network running at all times.

To provide various network services, telecommunication service providers network includes a number of middleboxes. They can support various types of functions (e.g., Firewalls, Network Address Translators (NATs), load balancers and Intrusion Detection Systems (IDSs)) and for this reason middleboxes are important for network operators.

By increasing network requirements in both scale and variety, service providers have to add new middleboxes and upgrade already existing middleboxes continuously. A recent research shows that the number of different middleboxes in enterprise network is comparable to the number of physical routers [1]. However, adding and updating middleboxes comes out with high Capital Expenditures (CAPEX) and Operational Expenditures (OPEX).

To address these issues, Network Functions Virtualization (NFV) [2] [3] has been proposed to transform middleboxes from specialized hardware appliances to software running on inexpensive, commodity hardware (e.g., x86 servers with 10Gb NICs) [4].

NFV-based (i.e., software-based) middleboxe is a good solution to reduce CAPEX, as network operators do not longer need to buy specialized hardware. In addition, software-based middleboxes can be deployed and managed dynamically without necessary network administrators to reduce OPEX [3]. In addition to improving OPEX and CAPEX costs, NFV gives an opportunity to network operators to manage their network functions easily. The software-based middleboxe is referred to Virtual Network Functions (VNFs) in NFV terminology that can run on the top of a virtualized infrastructure.

In order to use VNFs, they can be placed on computational nodes (e.g., servers, switches, data centers) that meets their resource demands. The computational nodes must provide NFV Infrastructure (NFVI) functions to support the execution environment. The placed VNFs can be also chained together to provide a required service.

Placement and chaining of the VNFs can affect both quality of service (QoS) and cost. For this reason, placement and chaining of the VNFs has recently attracted both network providers and researchers.

In this paper, the cost includes the license, computing and communication cost. Where the license cost includes instances and sites license and it is computed based on the number of utilized VNF instances and sites (i.e., the servers). The computing cost includes the cost of running VNFs on servers and communication cost is defined as the sum of the bandwidth used by the chains in the network. Although, some studies (e.g., [1]) consider the cost of transferring, booting and attaching a VM image to devices before deploying a VNF, we do not need to consider this cost in our work as we assume that the network provider is owner of NFVI.

The VNF placement and chaining problem in this paper is defined as follow: consider a chain of VNFs which should be placed on a given NFVI for the requested traffic (flows) from different sources and destinations. In order to minimize the overall cost four metrics should be considered: (i) how to find the optimal number of VNFs instances (ii) how to find appropriate locations for the VNF instances (iii) how to chain the placed VNFs instances in appropriate manner and (IV) how to assign the requested traffic to the chains of VNFs. Therefore, this paper defines a four-dimensional problem and the research question is how to consider these four, sometimes conflicting, metrics together.

The rest of the paper is organized as follows: Section II presents a business model, introduces the key requirements and reviews related works. Section III defines and fomulates the problem. Section IV describes the proposed method (CCVP). Section V portrays the simulation results and, finally, Section VI concludes the paper.

II. Background

A. Business Model

We assume a business model with the following entities: (i) telecommunication service provider (TSP), (ii) network operator, (iii) VNF provider and, (v) network consumer(s). TSP is the entity to provide the network services. Network operator owns servers and operates NFVI on the servers. VNF provider is the entity that provides VNFs. Network consumer (i.e., end user or end point) is the entity that consumes the network service. Similar to any business model, the same actor may play several roles at the same time.

Consider a scenario where a network operator needs to deploy a new service in its network. To this end, it asks TSP to provide that service. After receiving the new service request, TSP searches and uses the corresponding VNFs in its local VNF repository to provide the requested service.

However, if the required VNFs are not available, the TSP needs to ask new VNFs from the VNF provider. Finally, the network consumer uses this new service in a way that its received/sent traffic (flows) should pass through the new established service.

B. Requirements

In order to design a VNF placement and chaining algorithm, following issues should be considered.

First, the QoS (i.e., service delay) is important. For provisioning a service, an inefficient VNF placement and chaining can lead to high service latency.

Second, for provisioning a service, its VNF placement and chaining method should provide an acceptable run time for both small and large scale networks.

Third, for a service, both deployment cost of its VNFs and its delivering cost to the network consumers must be minimized.

As a result, this paper considers above mentioned issues to design the VNF placement and chaining algorithm.

C. Related Work

Most VNF placement algorithms deal with cost as an optimization objective. A VNF cost is generally made up of a set of individual costs (e.g., instance license, site license, deployment and communication cost). Some of the existing works focus on specific individual costs while others focus on a set of individual costs. They are successively discussed below. The shortcomings are pinpointed last.

1) Algorithms with single costs as objective: Luizelli et al. [5] propose an ILP model to minimize the number of instances for minimizing the license cost. They propose a binary search-based algorithm to improve the ILP run-time. Fang et al. [6] also attempt to minimize the number of deployed instances by proposing an ILP and the longest common sub-sequence (LCS)-based heuristic in inter-datacenter elastic optical networks (inter-DC EONs). They also consider the spectrum utilization cost for fiber links, which is the specialized cost for optical network.

Moens et al. [7] present an ILP to minimize the number of used servers (or compute resources) for the resource allocation of VNFs in NFVI and also for hybrid infrastructures where some NFs are virtualized and others use specific hardware appliances.

Some other studies have mainly focused on the communication cost. Qu et al. [8] propose an ILP and a greedy shortest-path-based heuristic to constructing chains through highly reliable VNFs in the NFV-enabled enterprise datacenter networks with the goal of minimizing the communication bandwidth usage across the network. Xia et al. [9] formulate the problem in binary integer programming (BIP) and propose a heuristic algorithm with the goal of minimizing the overall optical-electrical-optical (O/E/O) conversions (inter-DC traffic) in packet/optical DCs.

2) Algorithms for multiple costs: Unlike the above mentioned works with the simple objective, the following studies consider more complex cost models.

Ghaznavi et al. [10] present a solution called Simple Lazy Facility Location (SLFL) to optimize the placement of the same-type VNF instances in response to the on-demand workload. In this study, the elasticity overhead and the trade-off between bandwidth and host resource consumption are considered together.

In another study, Ghaznavi et al. [11] propose a Mixed Integer Programming (MIP) model and a heuristic called Kariz for multiple VNF instances placement to provide the functionality of a middlebox. Mechtri et al. [12] provide decomposition-based approach for the placement of virtual and physical network functions chains to maximize the provider's revenue based on the number of accepted CPU and bandwidth resources.

Riggio et al. [13] propose a VNF placement scheme to minimize the links and nodes utilization to increase the accepted service chain requests in enterprise WLANs. The authors then have extended their work in [14] where a VNF Placement heuristic called WiNE (Wireless Network Embedding) is proposed.

Sun et al. [15] consider cost as the one IT resources use for deploying the VNFs and Bandwidths cost. They propose an ILP as well as two versions of a heuristic to solve the VNF placement in online and an offline manner. The goal in the online heuristic is to maximize the revenue and the goal in the offline version is to minimize the cost.

Finally, a few studies attempt to consider more comprehensive cost models. Lin et al. [16] present a MILP and Game Theory based VNF placement with the goal of minimizing the cost to deploy NF instances as well as the computing and network cost in optical networks.

Zeng et al. [17] consider the cost of IT resource and spectrum utilization of fiber links as their objective in addition to the cost of VNF deployment (instantiating) in the VNF placement in optical datacenters. They propose a MILP and some heuristic to solve the problem.

Bouet et al. [18] propose an ILP and a centrality-based greedy algorithm to minimize the cost in virtual DPI (vDPI) placement where the cost includes the network cost, the license cost per site and the one per vCPU for VNF instances. Bari et al. [1] propose an ILP and a heuristic to solve the optimal VNF placement by running the Viterbi algorithm. The authors have also considered a penalty cost to be paid to the customer for th service level objective (SLO) violations.

3) Why is the previous work not adequate for the problem at hand?: The ILPs proposed by [5] and [7] for instance are not suitable for large scale environments. References [9]-[12] do not consider QoS (i.e., service delay threshold) and sharing VNFs among the chains. References [5]- [18] do not take a complete cost model into account. In addition, the work done by Bouet at el. [18] and Ghaznavi et al. [10] do not support VNF chain placement. They place instances of a VNF type individually (i.e., without considering the relation between VNFs of a chain). Unlike previous work, our work is appropriate for a large scale environment while trying to consider all the above-mentioned points.

III. Problem Definition

The VNF placement and chaining problem in this paper is defined as follows.

Consider a chain of VNFs which should be placed on a given NFVI for the requested traffic (flows) from different sources and destinations. In order to minimize the overall cost four metrics should be considered, (i) how to find the optimal number of VNFs instances (ii) how to find appropriate locations for the VNF instances (iii) how to chain the placed VNFs instances in appropriate manner and (IV) how to assign the requested traffic to the chains of VNFs.

Therefore, this paper defines a four-dimensional problem and the research question is how to consider these four, sometimes conflicting, metrics together.

A. Problem Formulation

The physical topology of the network is represented by a directed graph G=(V, E).

Let us consider N as a set of servers and U as a set of flows f with source S_f and destination D_f where N, S_f and $D_f \subseteq V$. V is the set of nodes composed of the servers and end-points of flows connected by directional edges E.

Let us also consider K as a set of VNFs, such as firewall, NAT and DPI. Each VNF of type $k \in K$ has a predetermined resource requirement and processing capacity, R_k and P_k, respectively. The network provider also delineates a set of instances for each VNF type $k \in K$, I_k. Such specification may be the result of the license model adopted while acquiring the VNFs from the VNF provider or as a result of management restrictions.

A service request $h_f \in H$, requested by flow $f \in U$, is represented by a directed graph $h_f = G(V^f_{\text{NF}}, E^f_{\text{NF}})$, where V^f_{NF} is the set of VNFs that will be installed on nodes in N and E^f_{NF} is the set of virtual edges that dictate the head and tail of the VNF chain. Table I delineates the inputs and variables used in our formulation. Our objective is to minimize the cost of VNF placement and chaining for network services in the network, including the cost of deploying VNF instances, the cost of using servers and the cost of communication as shown in (1).

$$Min. \left\{ \begin{array}{l} \left(\sum\limits_{k\in K, n\in N, j\in I_k} x_{k,n,j} \times \alpha_k \right) + \\ \left(\sum\limits_{\substack{f\in U, \\ k\in V^f_{\text{NF}} | \forall i \in V^f_{\text{NF}} \nexists (k,i)\in E^f_{\text{NF}}, \\ u\in N, j\in I_k, (u,f)\in E}} \beta^f_{(u,D_f)} \cdot \lambda_{f,k,u,j} \right) + \\ \left(\sum\limits_{f\in U, (p,q)\in E^f_{\text{NF}}, (u,v)\in E} \beta^f_{(u,v)} \cdot y^{f,p,q}_{u,v} \right) + \\ \left(\sum\limits_{\substack{f\in U, \\ k\in V^f_{\text{NF}} | \forall i \in V^f_{\text{NF}} \nexists (i,k)\in E^f_{\text{NF}}, \\ u\in N, j\in I_k, (u,f)\in E}} \beta^f_{(S_f,u)} \cdot \lambda_{f,k,u,j} \right) + \\ \left(z_n \times \sum\limits_{n\in N} (\gamma + \sum\limits_{k\in K, j\in I_k} (R_k \cdot \delta_n \cdot x_{k,n,j})) \right) \end{array} \right. \tag{1}$$

The cost of deploying VNFs has a software license cost per instance denoted by α_k. The cost of using server $n \in N$ is the sum of a fixed license cost γ and the operational costs for all VNF type instances. The license cost is the same for every server. The operational cost for a VNF instance of type k is $R_k \cdot \delta_n$. The cost of communication in VNF placement for network services is the sum of the bandwidth costs amongst each pair of servers $u, v \in N$ hosting VNFs of the VNF chain in each flow service request $h_f \in H$, $f \in U$, $\beta^f_{(u,v)}$. Furthermore, it includes the bandwidth costs between the servers $u \in N$ hosting the tail and head VNFs of the VNF chain in each flow service request $h_f \in H$, and flow end-points $f \in U$, $\beta^f_{(S_f,u)}, \beta^f_{(u,D_f)}$. Where, $\beta^f_{(u,v)} = L_f \cdot \sigma_{(u,v)} \cdot B_{(u,v)}$ is the cost of using bandwidth between $u, v \in V$, with hop count $\sigma_{(u,v)}$, for load L_f for flow $f \in U$.

VNF Placement:

$$\sum_{n\in N} \sum_{j\in I_k} \lambda^f_{k,n,j} = 1, \quad \forall f \in U, k \in K \tag{2}$$

$$\sum_{f\in U} L_f \cdot \lambda^f_{k,n,j} \preceq P_k \cdot x_{k,n,j} \quad \forall k \in K, n \in N, j \in I_k \tag{3}$$

TABLE I
INPUT PARAMETERS AND VARIABLES

Network Input	
N	Set of servers in the network, $N \subseteq V$
E	The set of edges (i.e., logical communication links) in the network
$BW_{(u,v)}$	The bandwidth capacity of edge $(u,v) \in E$
$D_{(u,v)}$	Delay of unit load (1 Gbps) for edge $(u,v) \in E$
$\sigma(u,v)$	Hop count of the edge $(u,v) \in E$
$B_{(u,v)}$	Bandwidth cost of unit load per hop between u and v where $u,v \in V$
$\beta^f_{(u,v)}$	Bandwidth cost incurred by sending load of flow f along edge $(u,v) \in E$
C_n	Capacity of server $n \in N$ in terms of resource units i.e., number of vCPU
γ	Site license cost
δ_n	Operational cost for unit resource (vCPU) for server n
Service Inputs	
U	Set of demand flows f
L_f	Load of flow $f \in U$
S_f	Source of flow $f \in U$, $S_f \subseteq V$
D_f	Destination of flow $f \in U$, $D_f \subseteq V$
H	Set of services h_f requested by flow $f \in U$
K	Set of VNF types that constitute all services $h_f \in H$
V^f_{NF}	$V_f \subseteq K$ is a set of VNFs that constitute service $h_f \in H, f \in U$
E^f_{NF}	Set of VNF edges of the VNF chain for service $h_f \in H$, $f \in U$,
I_k	Set of VNF instances of type $k \in K$
α_k	Software license cost of a VNF instance of type $k \in K$
$T_{k,n}$	Processing delay of VNF instance of type $k \in K$ on server $n \in N$ for unit load (1 Gbps)
R_k	Resource requirement (number of vCPU) for VNF type $k \in K$
P_k	Processing capacity (Gbps) of VNF type $k \in K$
D_{h_f}	QoS (i.e., Service Delay),threshold of service $h_f \in H$
Variables	
$x_{k,n,j}$	1, if instance j of VNF type k is assigned to server $n \in N$ and 0, otherwise
$\lambda^f_{k,n,j}$	1, if VNF type k belonging to VNF chain of flow f is mapped to its instance j on server n and 0, otherwise
$y^{f,p,q}_{u,v}$	1, if edge (u,v) hosts VNF edge (p,q) of VNF chain of flow f and 0, otherwise
z_n	1, if server n is used and 0, otherwise

$$\sum_{k\in K}\sum_{j\in I_k} R_k \cdot x_{k,n,j} \preceq C_n \cdot z_n \quad \forall n \in N \tag{4}$$

Variables $x_{k,n,j}$ $\forall k \in K, n \in N, j \in I_k$ are used to identify unique instance $j \in I_k$ of VNF type $k \in K$ installed on server $n \in N$. Variables z_n $\forall n \in N$ are used to record servers hosting VNFs.

VNFs chain mapping:

$$\sum_{j\in I_p} \lambda^f_{k,u,j} \cdot \sum_{j'\in I_q} \lambda^f_{k,v,j'} = y^{f,p,q}_{u,v} \qquad \forall f \in U, (p,q) \in E^f_{\text{NF}}, (u,v) \in E \tag{5}$$

$$\sum_{(u,v)\in E} y^{f,p,q}_{u,v} = 1 \quad \forall f \in U, (p,q) \in E^f_{\text{NF}} \tag{6}$$

$$\sum_{f\in U}\sum_{(p,q)\in E^f_{\text{NF}}} L_f \cdot y^{f,p,q}_{u,v} \preceq BW_{(u,v)} \quad \forall (u,v) \in E \tag{7}$$

QoS guarantee:

$$\left\{ \begin{array}{l} \left(\sum\limits_{\substack{f\in U,(p,q)\in E_f,\\ (u,v)\in E}} D_{(u,v)} \cdot L_f \cdot y^{f,p,q}_{u,v} \right) + \\ \left(\sum\limits_{\substack{k\in V_f \not\,\forall i,g\in V_f \exists (k,i)\in E_f\\ \& \exists (g,k)\in E_f,\\ n\in N, f\in U}} T_{k,n} \cdot L_f \cdot y^{f,p,q}_{u,v} \right) + \\ \left(\sum\limits_{\substack{i\in V_f \nexists (i,k)\in E_f,\\ u\in N, j\in I_k, (u,v)\in E,\\ f\in U,}} (T_{k,u} + D_{(u,D_f)}) \cdot L_f \cdot y_{f,k,u,i} \right) + \\ \left(\sum\limits_{\substack{i\in V_f \nexists (k,i)\in E_f,\\ j\in I_k, (u,v)\in E,\\ f\in U, u\in N}} (T_{k,u} + D_{(u,D_f)}) \cdot L_f \cdot y_{f,k,u,i} \right) \preceq D_{h_f} \end{array} \right. \tag{8}$$

We ensure in (2) that, for each flow service request $h_f \in H$, $f \in U$, all its requested VNF type must be assigned to only one of the deployed instance of that VNF type. It is assumed that an instance of a VNF type can cater to multiple flows.

Constraint (3) ensures that the capacity of an instance of a VNF of type $k \in K$ is not exceeded by the total load requested by all the flows assigned to it. Constraint (4) ensures that the total resource required by instances of all VNF types on a server does not exceed the capacity of the host server.

In our model, we map the nodes (i.e., the VNFs) and their edges (i.e., the chain), in each flow service request to the physical network in G. Therefore, the edge (p, q) between two consecutive VNFs in each flow service request must be assigned to a physical edge (u, v) between two servers u and v, in (5). It should be noted that (5) is a non-linear constraint and can be trivially linearized. As is ensured in (6), the VNFs and their respective ordered edges are mapped to only one pair of physical servers and their edge. With respect to the underlying physical network, we guarantee that the total load on an edge in the physical network does not exceed the bandwidth capacity, in (7).

Constraint (8) is the QoS constraint that guarantees that the flow service request is delivered within the predefined delay threshold. The delay in delivering the service consists of two components, the network communication delay and the VNF processing delay on the servers. The network communication delay is the sum of delay between each pair of servers hosting the VNFs of the VNF chain in the flow service requests. It also entails the delay for communicating between the server which hosts the tail and head VNFs of the VNF chain and the end-points of the flows. The processing delay of VNFs hosted on the servers is proportional to the flow load assigned to the VNFs.
The VNF placement and chaining for network services is an NP-Hard problem, calling for an efficient heuristic.

IV. Cost-efficient Centrality-based VNF Placement and chaining algorithm (CCVP)

This section presents our Cost-efficient Centrality-based VNF Placement and chaining algorithm (CCVP). First, a general mechanism of CCVP is described and, then, we present logical architecture and modules of CCVP.

A. CCVP General Mechanism

The main objective of CCVP is to find the optimal places for VNFs of a requested chain in order to get the minimum cost. The proposed algorithm can be used from small to large scale networks.

To present the CCVP mechanism, let consider a scenario where a chain h including various VNF types should be allocated on the given network and a set F of flows ($F \subseteq U$) passing through the chain h. To this end, CCVP mechanism includes two main steps. It first creates a graph G of the network topology (details described in the section IV-B1). In the second step, based on the graph G characteristics and other received information (e.g., available instances and requested flows), CCPV selects appropriate servers, deploy instances on them and indicates each flow f passes through which selected server for all VNF types of the chain h.

It is worth to highlight that CCVP is based on the Betweeness centrality algorithm [19]. The high centrality indicates that a vertex of a graph G can reach other vertices on relatively short paths, or that vertex lies on a considerable fraction of shortest paths connecting pairs of other vertices.

In CCVP, the server with the highest centrality is the potential node to host the VNF instances. As a result, CCVP try to assign the flows to the VNFs without deviating from their shortest path and, therefore, without using additional network resources which directly impact the overall cost.

B. CCVP Logical Architecture

CCVP consists of three main modules:

- *Graph Creator (GC)*
- *Chain Placement (CP)*
- *VNF Placement (VNFP)*

where the VNFP module itself includes following submodules:

- *Server Selector (SS)*
- *Centrality Computer (CC)*
- *Traffic Mapper (TM)*
- *Fitness value Computer (FVC)*

In this section, the modules are described one by one.

Algorithm 1: CP Algorithm

```
Input: G(V,E), $\mathcal{I}$, $h$, $F(S_f, D_f, L_f)$
 // $F \subseteq U$ indicates the flows requesting
 the chain $h$, $h \subseteq H$
Output: $\mathcal{S}_{sel}^{tot}$, $\mathcal{I}_{dep}$, A[ ]
/* Initialization                              */
1  $\mathcal{S}_{sel}^{k} = \varnothing$ // set of selected servers of
    VNF type $k \subseteq h_f$
2  $\mathcal{S}_{sel}^{tot} = \varnothing$ // set of all selected servers
3  $\mathcal{AS}[\,][\,] = \varnothing$ // for each flow indicates
    the server hosting an instance of VNF
    $K \in K$ where the flow passes through it
4  $\mathcal{I}_{dep}^{k} = \varnothing$ // set of deployed instance of
    VNF type $k \subseteq h_f$
5  $\mathcal{I}_{dep} = \varnothing$ // set of all deployed
    instances
6  for all $k \in h$ do
7      ($\mathcal{A}[k][\,]$ ,$\mathcal{I}_{dep}^{k}$,$\mathcal{S}_{sel}^{k}$)=VNFP(G,$F(S_f, D_f, L_f)$,k)
8      for all $f \in F$ do
9          $S_f$=$\mathcal{AS}[k][f]$
           // AS[k][f] indicates the vertex
           which host an instance of VNF K
           and flow f passes through it
10     $\mathcal{S}_{sel}^{tot} = \mathcal{S}_{sel}^{tot} \cup \mathcal{S}_{sel}^{k}$
11     $\mathcal{I}_{dep} = \mathcal{I}_{dep} \cup \mathcal{I}_{dep}^{k}$
12 return ( $\mathcal{S}_{sel}^{tot}$, $\mathcal{I}_{dep}$, $\mathcal{A}[\,][\,]$)
```

Algorithm 2: VNFP Algorithm

```
Input: G(V,E), $\mathcal{I}_{||}$,$F(S_f, D_f, L_f)$,k
Output: $\mathcal{A}^k_{sel}$, $\mathcal{I}^k_{dep}$, $\mathcal{S}^k_{sel}$
/* Initialization                                  */
1  $\mathcal{S}^k_{sel} \leftarrow \varnothing$ // set of selected servers for
     VNF $k$
2  $\mathcal{I}^k_{dep} \leftarrow \varnothing$ // set of deployed instances of
     VNF type $k$
3  $\mathcal{AS}[\,][\,] = \varnothing$ // for each flow indicates
     the server hosting an instance of VNF
     $K \in K$ where the flow passes through it
4  $fitness_{min} \leftarrow \infty$ , j=0
5  Map F on G and Compute the centrality of all $v \in V$
6  $v_m \leftarrow$ a node with maximum centrality and enough
     capacity
7  while centrality $v_m > 0$ and $F \neq \varnothing$ and $\mathcal{I}^k_{dep} \neq \mathcal{I}^k$ do
8      $\mathcal{I}^k_j \leftarrow j^{th}$ instance from $\mathcal{I}_k$
9      $S^k_j \leftarrow v_m$
10     $\mathcal{S}^k_{sel} = \mathcal{S}^k_{sel} \cup S^k_j$
11     $fitness(G, F, \mathcal{S}^k_{sel}) \leftarrow$ fitness value for (G,F,$\mathcal{S}^k_{sel}$);
12     if $fitness_{min} \neq \infty$ then
13         if $F \neq \varnothing$ or $fitness(G, F, \mathcal{S}^k_{sel}) < fitness_{min}$
             then
14             $fitness_{min} = fitness(G, F, \mathcal{S}^k_{sel})$
15         else
16             break;
17     $F_r$ = the flows whose shortest path is passing
         through $v_m$
18     i=0;
19     while capacity($\mathcal{I}^k_j$) > 0 and $F_r \neq \varnothing$ do
20         Remove a flow f from $F_r$
21         Remove required resources of f from G
22         capacity($\mathcal{I}^k_j$)= capacity($\mathcal{I}^k_j$) - $L_f$;
23         $\mathcal{AS}[k][f] \leftarrow v_m$
            // AS[k][f] indicates the vertex
            which host an instance of VNF K
            and flow f passes through it
24         $\mathcal{A}^k = \mathcal{A}^k \cup \mathcal{A}[f][k]$
25         $\mathcal{I}^k_{dep} = \mathcal{I}^k_{dep} \cup \mathcal{I}^k_j$
26     Map updated F on updated G
27     Compute the centrality of all $v \in V$
28     $v_m$=a node with maximum centrality and enough
         capacity
29     j++
30 return ( $\mathcal{S}^k_{sel}$, $\mathcal{I}^k_{dep}$, $A[k][]$)
```

1) Graph Creator (GC) module : In a real environment, there is a collection of servers and network elements (i.e., routers) which are directly or indirectly connected together. The GC module creates a high abstracted graph G based on the real environment where a server and its one-hope neighboring routers are considered as a vertex. An edge between two vertices x and y is considered as the shortest path between them.

2) Chain Placement (CP) module: CP module (see Algorithm 1) receives a network graph G, a chain (h), a set of available instances (I), a set of requested flows (F, $F \subseteq U$) as inputs. It then (getting help from VNFP module) returns the selected servers (i.e, S^{tot}_{sel}), instances on them (i.e, I_{dep}) and, finally, indicates each flow f passes through which selected server (i.e., AS) for all VNF types of the chain h.

Looking at line 6 of Algorithm 1, CCVP starts from the first VNF type (e.g., type k) to select appropriate servers for allocating its VNF instances. As line 7 of Algorithm 1 shows, for a VNF type k, it returns the selected servers (i.e, S^k_{sel}), instances on them (i.e, I^k_{dep}) and, finally, indicates each flow f passes through which selected server (i.e., AS).

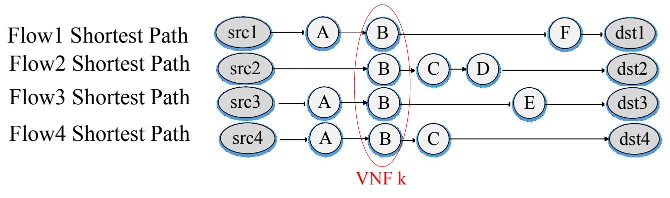
CP then modifies each source of flows (using a loop For in line 8) by replacing a server which hosts already allocated VNF instance with previous source of the flow. The reason of flow source modification is described in the section IV-B2.

Flow Source Modification technique (FSM)– Consider a simple scenario where there is a chain with two VNF types (i.e., k and k') and also four shortest paths have been found for flows 1-4 (Fig. 1). The goal is to allocate a VNF instance of types k on one of the available servers A-E. After that, the second instance (type k') should be allocated. Looking at Fig. 1(a), the first instance (VNF type k) is allocated on server B. This is because server B has the highest centrality value (i.e., the shortest paths of all flows have been passed from B and traffic passing from it is the highest among other servers). For selecting the second server (assume server B is full now), a common way is to again consider an available server with highest centrality value. As Fig. 1(b) shows, server A is selected to allocate instance of type K'. However, in this case, all flows have to return from server B to A to pass from the instance of VNF k'.

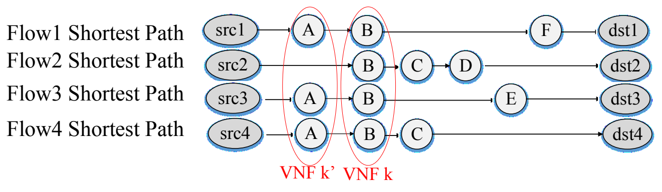
In order to avoid above mentioned situation and offer a more appropriate communication cost between already allocated instances of type k and upcoming (next allocated) instances of VNF type k', we propose a Flow Source Modification technique (FSM).

For allocating instances of the upcoming VNF type k', FSM first shifts sources of flows from their current locations to the related servers which host allocated instances of type k (from sources src1-scr4 to server B in Fig. 1(c)). Modifying (updating) the flows sources can prevent increasing paths length (prevent returning path in direction of the sources) and also can improve communication cost (by selecting servers which are located between already allocated instances and destinations). In other words, FSM can help CCVP to assign the flows to the VNFs without deviating from their shortest path and, therefore, without using additional network resources which directly impacts the overall cost.

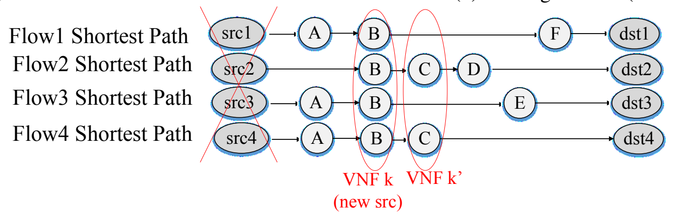
3) VNF Placement (VNFP) module–: This module (see Algorithm 2) is called by CP module in order to select appropriate servers (i.e, S^{tot}_{sel}), instances on them (i.e, I_{dep}) and also to indicate each flow f passes through which selected



(a) Placing VNF k



(b) Placing VNF k'(without source modification)



(c) Placing VNF k'(using FSM technique)

Fig. 1. Flow Source Modification technique(FSM)

server (i.e., AS) for a VNF type k of the chain h. In brief, VNFP follows 5 steps: (1) to select a vertex for a VNF instance type k, it calls SS module and sends a list of unassigned flows to it. (2) After receiving a vertex from SS module, it calls FVC to compute the fitness value for estimating the overall cost of the new placement (after adding the new instance). (3) if the previous placement could assigned all requested flows, VNFP compares fitness value of the new placement to the previous placement (in line 13), (4) if fitness value of the new placement is lower than previous placement, as long as the VNF instance has capacity, it then assign the flows which passing through this selected vertex (lines 19-25), (5) It updates the network resources based on requirement of the placed instance and adds the placed instance to the list of deployed instances. Finally, the centrality of the vertices are updated based on the new set of flows.

The procedure of placing new instances of VNF type k (Steps 1-5) is continued until the fitness value becomes larger than the former ones (and the former one could assigned all requested flows successfully). In this case, NFVP returns and chooses the previous placement. That means we are deploying the new instance of VNF type k as long as the fitness value of the current placement becomes less than previous one and that condition may not lead to overloaded of the VNFs.

VNFP module includes following sub-modules:

Traffic Mapper (TM) module– It receives a graph G and a set of flows. It first finds the shortest path for each flow and, then, maps the computed shortest paths to the graph G.

Centrality Computer module (CC) module– This module receives a graph G and a set of flows. It calculates and returns a centrality value for each vertex of the graph G. The centrality value for a vertex is computed based on the total load of flows which their shortest paths passing through that vertex. To this end, CC module first calls TM module in order to find which flows are passing through what vertices. CC module then computes total load passing from each vertex.

Server Selector (SS) module– This module selects an appropriate vertex (server) for allocating a VNF instance of a requested chain. To select a vertex for a VNF instance type k, it calls CC module in order to get centrality value of vertices. It then sorts centrality values of the vertices and selects a vertex with the highest centrality value and enough capacity to place the instance. In the case of existing several vertices with the highest centrality, SS module selects the vertex which causes the lowest shortest path from the sources of remained flows to that vertex (to select the vertex for the last VNF instances, SS compares the lowest shortest path from the sources of remained flows to the flows destinations passing from that vertex). Even if the shortest paths deduced by the vertices are also the same, this module selects the vertex with more available resources.

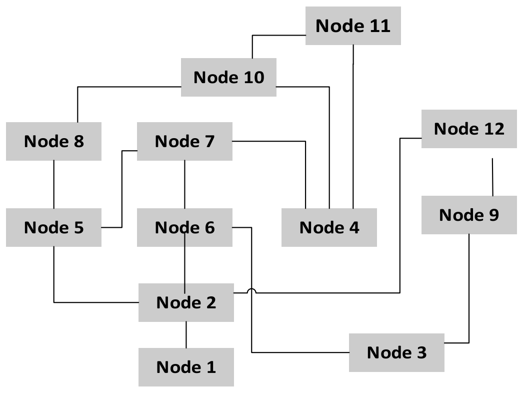


Fig. 2. Internet 2 Topology

Fitness value Computer (FVC) module– Briefly, there are two parts to calculate the fitness value: first part is related to the license cost of VNF, site license cost and server running cost (operational cost) and second part is related to the cost of additional network resources cost, named network footprint, which is calculated based on the shortest path, bandwidth unit cost and load of the flows.

If a flow has been already assigned to a vertex, the shortest path is considered from the source of that flow to the vertex.

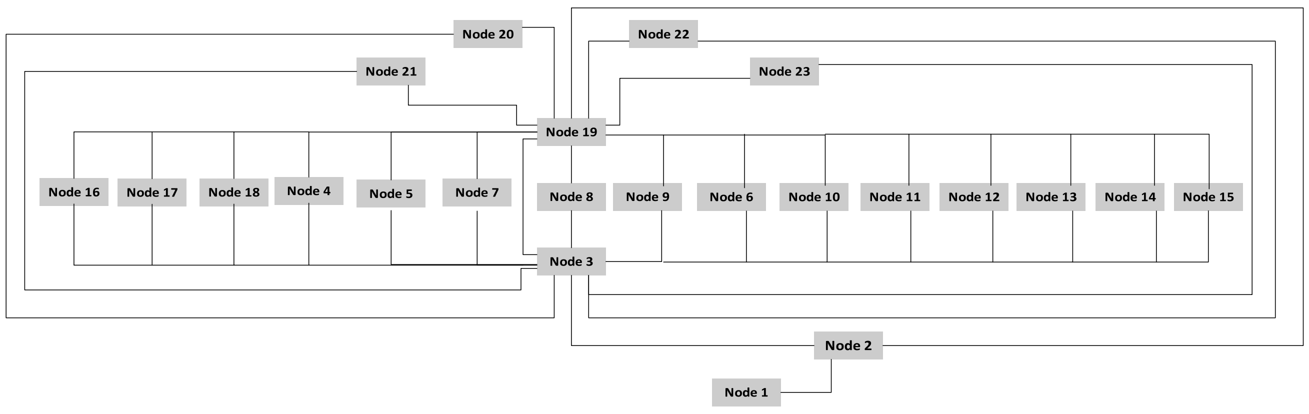


Fig. 3. Data Center Network Topology

If a flow has not been assigned yet, FVC temporary selects a vertex from already selected vertices with minimum shortest path from the source of that flow to that vertex (the vertex should host at least a VNF with enough free capacity).

It should be noted that for the last VNF, the shortest path is calculated from the source to the destination of the flows passing from the vertex. If still some flows remain unassigned, new VNF instances should be deployed (their shortest path considered as infinite).

V. Performance Evaluation

The goal of this section is to compare the performance of CCVP with Proposed ILP VNF placement algorithm using different network topologies and multiple service chain requests. The ILP is considered as optimal cost to show how efficient CCVP is in terms of overall cost. It should be noted that, since processing delay and link bandwidths are not considered in designing the current version of proposed algorithm, the proposed algorithm is compared with the modified version of ILP. In particular, the constraints 7 and 8 are relaxed.

CCVP have been implemented in MATLAB and AMPL-Gurobi 6.5.1 is used to implement ILP. Two real-world network topologies used in order to evaluate the algorithms. The simulation setups and the metrics, which are used for evaluating, are described in next section.

A. Simulation Setup

Network Topology: Both algorithms run on two different network topologies: (i) Internet2 research network [20] which is composed of 12 nodes and 15 links (see Fig. 2), (ii) A university data center research network [21] which is composed of 23 nodes and 42 links (see Fig. 3).

Traffic data set: the number of flows changes from 4 to 20 flows. The sources and destinations were generated randomly by using MATLAB. It is worth highlighting that, even though the flows configuration are set randomly, once set, it remains fixed across the runs of all tested algorithms ensure comparability of the results. We run simulations for five different random set of sources and destinations and report the average of the five simulations in the figures. A chain with 2 VNFs have been applied in the simulations. The capacity of the physical servers, physical links, VNF instances and server running costs, which we have used in this section, have been chosen the ones which have commonly applied in the most research papers [1] [18]. Table II shows the server and VNF data assumption which is used in evaluation. Note that, CCVP is applicable for various kind of cost assumptions.

TABLE II
Simulation Parameters

Simulation Parameters	
Bandwidth cost (Dollar/Mbps)	10
License cost per VNF instance(Dollar)	1000
Operational cost (Dollar/vCPU)	5
Required CPU cores per VNFs	4
Flow size per source and destination pair(Gbps)	1
VNF capacity (Gbps)	5
Number of flows	4,8,12 16,20

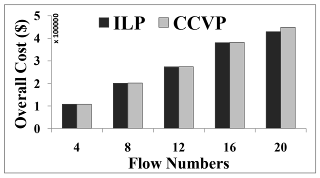
B. Cost Evaluation

Since one of the major benefits of NFV is the significant reduction of OPEX, we verify this claim by showing the quantifiable outcomes.

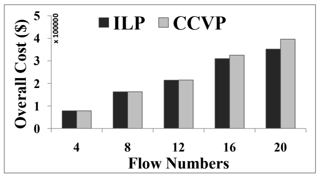
figures 4 (a) and (b) depict the results of the overall cost of the data center and Internet 2 network topologies for the multiple flow requests. In general, by increasing the number of flow requests, the overall cost of service provider also increased gradually since more number of VFNs have to be deployed in all cases. We can also see that the overall cost, as the objective of CCVP, is always close to ILP.

To better evaluation of the overall cost, let looking at figures 4 (c)-(f). The behavior of the overall cost graphs are based on operational and communication cost graphs.

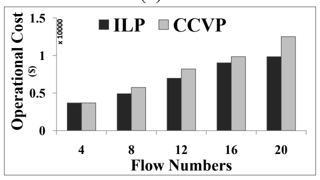
As figures 4 (c)-(f) show, when the number of flow requests is increased, the communication, operational and overall cost of service provider also increased since more number of VFNs



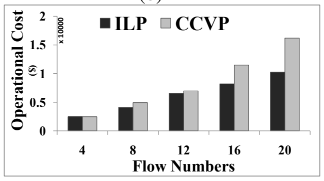
(a)



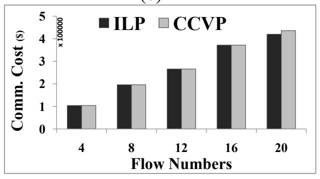
(b)



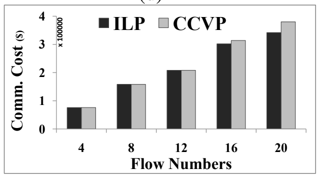
(c)



(d)



(e)



(f)

Fig. 4. Simulation results: (a) Internet2 Overall Cost, (b) DC Overall Cost, (c) Internet2 Operational Cost, (d) DC Operational Cost, (e) Internet2 Communication Cost, and (f) DC Communication Cost.

have to be deployed and communicate in all cases. However, based on the value of the flows and unit of operational and communication cost, the effect of communication is more that than the effect of operational cost in the overall cost.

VI. Conclusion

This paper formulated the problem of VNF placement and chaining as an Integer Linear Program (ILP) and proposed a Cost-efficient Centrality-based VNF Placement and chaining algorithm (CCVP) algorithm for network service provisioning. The objective is finding the optimal number of VNFs along with their locations in such a manner that the overall cost is minimized. Through simulations, the algorithms behavior for two real network topologies was analyzed and compared with ILP. The simulation results showed that overall cost of the proposed algorithm is close to ILP and hence could be used successfully in various topologies.

References

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

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, Feb 2015.

[3] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[4] ETSI, "Etsi gs nfv 001, network function virtualization (nfv) use cases, v1.1.1." ETSI, , 2013.

[5] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 98–106.

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

[7] H. Moens and F. D. Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Nov 2014, pp. 418–423.

[8] L. Qu, C. Assi, K. Shaban, and M. Khabbaz, "Reliability-aware service provisioning in nfv-enabled enterprise datacenter networks," in *IFIP/IEEE 12th International Conference on Network and Service Management*, Oct 2016.

[9] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, April 2015.

[10] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct 2015, pp. 255–260.

[11] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, and R. B. R. Ahmed, "Service function chaining simplified," http://arxiv.org/abs/1601.00751, coRR, vol. abs/1601.00751, 2016.

[12] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 533–546, Sept 2016.

[13] R. Riggio, T. Rasheed, and R. Narayanan, "Virtual network functions orchestration in enterprise wlans," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 1220–1225.

[14] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, June 2016.

[15] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted nfv service chain deployment based on affiliation-aware vnf placement," in *IEEE Globecom*, December 2016.

[16] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Demand-aware network function placement," *Journal of Lightwave Technology*, vol. 34, no. 11, pp. 2590–2600, June 2016.

[17] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type vnf forwarding graphs in inter-dc elastic optical networks," *Journal of Lightwave Technology*, vol. 34, no. 14, pp. 3330–3341, July 2016.

[18] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–9.

[19] M. E. Newman, "A measure of betweenness centrality based on random walks. social networks," *Social Networks*, vol. 27, no. 1, pp. 29–54, 2005.

[20] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures," *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.

[21] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," *ACM SIGCOMM conference on Internet measurement*, pp. 267–280, 2010.