Placement and Routing of VNFs for Horizontal Scaling

R. Gouareb, V. Friderikos and A.H. Aghvami
Department of Informatics, King's College London, London, UK
Corresponding author email: racha.gouareb@kcl.ac.uk

***Abstract*—To deploy a wide range of services, improve network flexibility and facilitate the deployment and maintenance of network services, virtualization of network functions promises to make the networks more agile and efficient. Network functions virtualization (NFV) enables network service providers to deal with the increase of traffic of data and the deployment of new services by offering a more scalable and flexible network. Many existing studies on NFV have dealt with the placement of virtual network functions (VNFs) in the software layer as a problem of deploying different network function instances within the network without considering routing and horizontal scalability. In the proposed model, we aim to minimize latency, considering horizontal scaling of VMs by placing VNFs across physical nodes. Routing and placement of VNFs are linked and have a significant impact on latency especially for delay-critical services. Therefore, we aim at optimizing the distribution and utilization of available resources and meet user/tenant requirements with the objective of minimizing the overall accumulated latency on both the edge clouds, where VNFs are running and the flow routing paths.**

***Index Terms*—Virtual networks, 5G networks, Network function virtualization, Virtual Network function, Horizontal scaling, latency.**

I. Introduction

As a fundamental technology for the cloud computing industry, virtualization has become an important focus of companies in order to deal with the growth of their workloads and create a more flexible and scalable environment.

In addition to virtualization of different resources, as defined by the European telecommunications standards institute (ETSI) [1], virtualization of network functions allows orchestration and management in a single infrastructure.

Service providers, such as Amazon [2] and Google [3] use virtualization technology to expand their revenue opportunities, deliver new services and capabilities and modify virtual capacity as needed for maximum agility and efficiency. Cloud service providers offer virtual infrastructure or applications from a shared physical infrastructure. Owning a virtual infrastructure, they can offer users to pay for only what they use without any control or ownership. A user/tenant request is composed of different ordered virtual functions (VNFs) that have to be assigned and routed through one or different virtual networks and deployed by virtual machines (VMs) considering both demand and system conditions. To keep up with the increase or decrease in customer demand and workload, VMs can be scaled differently: (i) horizontally by increasing or decreasing the number of virtual entities behaving as one entity to adjust to the variation of the load of network which can lead to the waste of resources even in a case where workload slightly increases; and (ii) vertically by decreasing or increasing the number of resources allocated to an existing VM during runtime, such as memory or CPU [4]. By increasing the scale of a VNF, both horizontal and vertical scaling result in increasing the consumption of physical resources the same way.

Few cloud providers support vertical scaling such as Profitbricks [5], however big providers have not implemented vertical scaling yet such as Amazon [2]. Hence, we studied the problem of VNFs routing and placement, this paper evaluates the solution of MILP and the heuristic algorithm considering horizontal scaling in an off-line network. To optimize latency, we present a model considering horizontal scaling as a Mixed Integer Linear Programming (MILP) model. The delay is defined by both: (i) inter cloud delay as the queuing delay occurred within the cloud; and (ii) intra-cloud delay as the queuing delay in the link. As the optimal problem is NP-hard, we propose a heuristic algorithm taking into account horizontal scaling to find an approximate solution to VNFs placement and routing problem, and we compare it with the optimal solution and a greedy approach from previous research work [6].

We organize the rest of the paper in such a way that: Section II presents the state of the art. Section III describes the suggested model and defines the mathematical formulation and heuristic algorithm. In Section I, we are discussing the results obtained from the comparison between the optimal and sub-optimal solutions, this to analyse the network performance with the presented approach. Finally, in the last section, we concluded this paper.

II. RELATED WORK

Various approach are developed to effectively solve the VNFs placement problem by considering different metrics and measurements such as infrastructure resources and different network costs. Reference [7] have modeled traffic steering and service placement as a joint problem.In this paper the main objective is to minimize the core resource and link utilization. [6] proposes an approach to the problem of virtual placement across distributed clouds minimizing inter-cloud traffic and response. Using an ILP optimization they define latency as computational and link delays, modeled as M/M/1 and M/D/1 queuing models respectively. Differently, Bi, Zhu, Tian, and Wang [8] have modeled the first and the rest of the tiers as M/M/1 and M/M/m queuing models respectively. They have

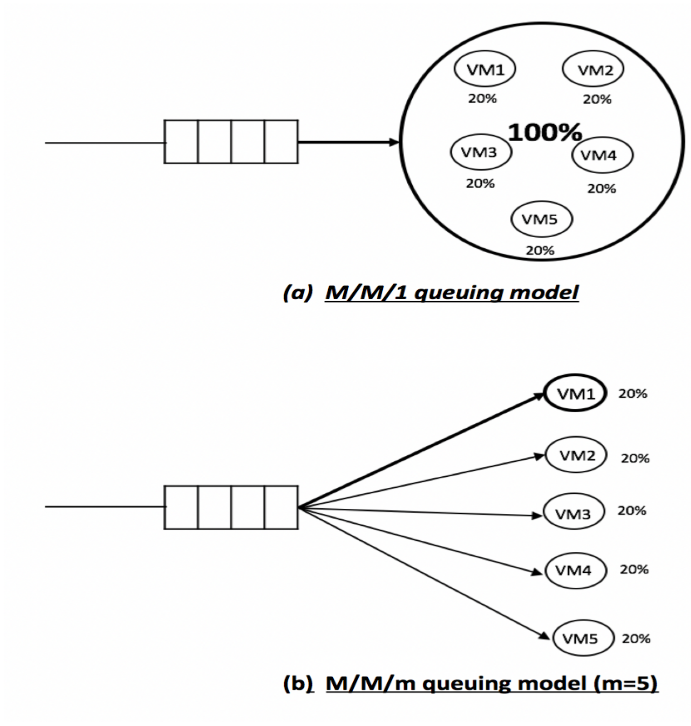

Fig. 1. Horizontal scaling

proposed a flexible optimization in order to minimize the VMs number. They have maintained the required arrival time and the average response time, reaching the optimum resource utilization. In [9], authors have presented a load balancing approach using multi-path routing, where the evaluation shows a decrease in latency by 6% and a growth in user demands.
One of the approaches to VNFs placement problem considering VMs scaling is [10], presenting a Genetic algorithm (GA) for VNFs resource allocation. They have analyzed an algorithm for VNFs initial placement and another genetic algorithm for existing VNFs scaling during the variation of the traffic.
In this work, we propose an optimization model for VNFs placement and routing considering horizontal scaling by instantiating new VNF components with the aim of minimizing the overall delay.

III. System Description

This section presents the problem of delay optimization by defining link and cloud queuing delay and various constraints to satisfy and that are described later in this section. A request or a VNFs chain is defined as an ordered functions chains, similarly to various previous work such as in [11]. We assume that requests are already accepted, each can be composed by different services such as Session Border Controller (SBCs), Firewall and Deep Packet Inspection (DPI) [12]. We can assign different VNFs to one edge cloud if the capacity constraints are satisfied or in different nodes where the flow should go through the required services in the order specified in the request. For each request, we assign two shortest paths, generated based on the link weights that are predefined by service providers, using Dijkstra's algorithm [13]. The first choice shortest path is assigned to the related request as far as the link and nodes capacity constraints are not infringed, the second shortest path will be used otherwise.

We will be using edge cloud and node interchangeably, where an edge cloud is defined by its remain capacity. The available resources of a node can be shared fully or partially, by one or different virtual machines. In the case where VMs are scaled horizontally, the set of one type of VNF instances will be homogeneous, but the set of VMs assigned to the same node can be heterogeneous if they are processing different VNFs. As illustrated in the figure 1, inter-cloud delay model can follow either M/M/1 (a) or M/M/m (b) queuing models where the queuing delay general equations are defined in (1) and (2) by D and $D\prime$ respectively. In the first case (a), the edge cloud is considered as an aggregated processing unit, where the total processing capacity is the total of VM capacities assigned to it. However, in the second case (b), we consider each VM as the processing unit independently, which capacity is the resource portion assigned to each VM. In the case of low utilization, modeling the queuing delay as an M/M/1 model will result in under evaluating the delay compared to the M/M/m model representation. This clearly because one processing unit can process a request much faster than a small processing unit with only 20% capacity of the aggregated set of VMs in the model (a). This can be shown in figure 2 where the delay in the case of M/M/1 queuing model is under evaluated compared to the delay we modeled as M/M/m. However, the model results in a very close to a similar delay when the utilization is medium to high, this because one processing unit is expending its capacity between the requests as the flow of requests increases. Therefore, the behaviour of one high capacity unit get closer to the performance of different small units with an equal total capacity. As we consider in this paper a medium to high arrival load and utilization, we used M/M/1 model [14] to model latency, where arrivals and service time follows a Poisson process an exponential distribution respectively, this model both link and inter-cloud queuing delays.

$$D = \frac{1}{\mu - \lambda} \tag{1}$$

$$D\prime = \frac{1}{\mu} + W \tag{2}$$

Therefore, this work contributes mainly to optimizing the delay when scaling the nodes horizontally and evaluating the VNFs placement and routing model. Therefore, this will result in maximizing the revenue of providers by improving the deployment of their infrastructure and increasing the number of users. Since we scale out VNFs by instantiating new components if needed, we will follow a single-feature multi-requests (SFMR) model. SFMR consider that a service instance is serving flows related to one request. We compare optimal and suboptimal solutions and compare between a greedy

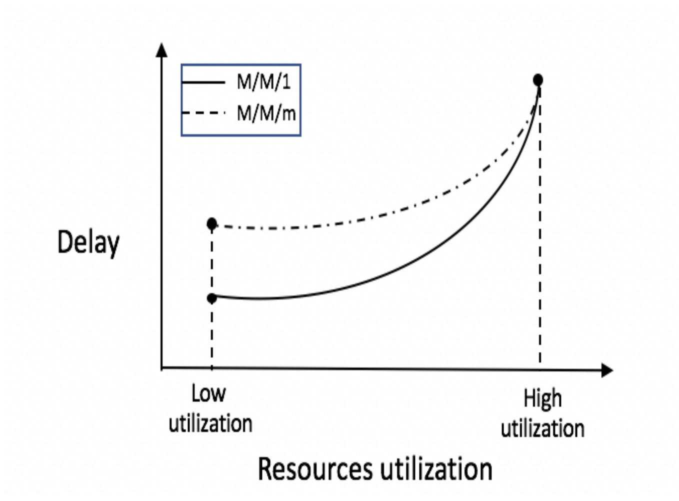

Fig. 2. Delay modeling

approach used in [6] and the proposed heuristics to validate our approach. We describe the mathematical programming formulation that represent our model, we define the objectives and the constraints, then we explained the linearization of the non-linear terms in the constraints and the objective function.

A. Preliminaries

TABLE I
VARIABLES DEFINITIONS

Variable	Domain	Definition
$h_{r,f}$	$\mathbb{R}_{>0}$	Rate of processing f
μ		Service rate
λ		Arrival rate
W		Average customer waiting time
$N_{r,e,f}$	$N_{r,e,f} \in \mathbb{R}_{>0}$	Delay occured by function f in node e
L_l	$L_l \in \mathbb{R}_{>0}$	Link delay
T_l	$T_l \in \mathbb{R}_{>0}$	Link l total traffic
C_l	$C_l \in \mathbb{R}_{>0}$	Link l remaining capacity
S_p	$S_p \in \mathbb{R}_{>0}$	Paths p cost
r	R	Request
l	L	Link
p	P	Path
e	E	Edge cloud
f	F	Function
n	N	Breaking point
η_{rf}	$\eta_{rf} \in \{0,1\}$	If a request r requires f, equal to 1
x_{ref}	$x_{ref} \in \{0,1\}$	If request r use e for f, equal to 1
ψ_{rp}	$\psi_{rp} \in \{0,1\}$	If path p hosts request r, equal to 1
v_{ep}	$v_{ep} \in \{0,1\}$	If node e belong to p,equal to 1
ω_{ref}	$\omega_{ref} \in [0,1]$	Defines the capacity utilization of e capacity to host f for a request
d_r	$d_r \in \mathbb{R}_{>0}$	Required total of functions by r
μ_e	$\mu_e \in \mathbb{R}_{>0}$	Edge cloud e average processing rate
λ_r	$\lambda_r \in \mathbb{R}_{>0}$	Edge cloud e arrival rate
ζ_{pl}	$\zeta_{pl} \in \{0,1\}$	If link l is apart of p, equal to 1 i
A_f	$A_f \in \mathbb{R}_{>0}$	A VNF f instance resource requirements
R_e	$R_e \in \mathbb{R}_{>0}$	Node e resource capacity
T_r	$T_r \in \mathbb{R}_{>0}$	Request r delay tolerance

We model substrate network and defined $G=(K,L)$ as an undirected graph, where the set of edge clouds and links are defined by E and L respectively. We present our model as MILP optimization problem where key notations are described in Table I. In order to place VNFs, forward flows through the network and assign requests to the appropriate nodes and paths, we define the objective of delay optimization. By considering request requirements as well as network's available resources, we describe the constraints in the following section.

B. Mathematical programming formulation

To model delay optimization problem, M/M/1 queuing model is used for both edge cloud and link delay modelling. For the packets to be processed by VMs, they are placed in a queue waiting for a processing unit (VM) to be available. In addition to the edge-cloud delay, link queuing delay is generated when the packets are waiting to be transmitted through links. By considering the M/M/1 model we will be using a VMs set, and edge cloud, a node, and a processing unit interchangeably. Every node can serve one or more request, subject to its remaining capacity, the instances of VNFs assigned to it is limited.

We define two decision variables: (i) x where x_{ref}=1 if a required VNF f is utilizing a part of an edge cloud e capacity by opening VMs horizontally: network, computing available resources and memory; and a service instance is assigned to i;. and ψ where ψ_{rp} = 1 when request r go through nodes and links of a path p for different function instances. ζ_{pl} and v_{ep} are defining the the links and the nodes respectively belonging to path p.

$$x_{ref} = \begin{cases} 1 & \text{if the related flow to a request } r \text{ uses} \\ & \text{node } e \text{ for a service instance } f. \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$$\psi_{rp} = \begin{cases} 1 & \text{if the request } r \text{ go through path } p. \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

ζ_{pl} and v_{ep} describe the links and the nodes respectively on path p .

$$v_{ep} = \begin{cases} 1 & \text{if a node } e \text{ is part of path } p. \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

$$\zeta_{pl} = \begin{cases} 1 & \text{if a link } l \text{ is part of path } p. \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

We define link and edge cloud queuing delays following Little's Theorem [14] in Equation 8 and 7 respectively, where C_l is the capacity of the link, T_l the total traffic in link l, the arrival rate and the processing capacity are defined by λ_r and h_{rf} respectively.

$$N_{ref} = \frac{1}{h_{rf} - \lambda_r} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \quad \forall r \in \mathbf{R} \tag{7}$$

$$L_l = \frac{1}{C_l - T_l} \quad \forall l \in \mathbf{L} \tag{8}$$

We are solving the mathematical formulation representing the cloud and link delays. In order to linearize the link delay, we use a piecewise linear function [15] with pre-generated breaking points $b_1, b_2, ..., b_n$ and $z_{i,l}$ as piece-wise linear slot weight. The objective function is formulated based on the above notations and solved using Matlab tool for mixed integer linear programming (MILP), we formulate the problem as follows:

$$\min_{\psi_{rp}, x_{ref}} \sum_{l \in \mathbf{L}} \sum_{n \in \mathbf{N}} z_{ln} L_l(b_n) + \sum_{e \in \mathbf{E}} \sum_{f \in \mathbf{F}} \sum_{r \in \mathbf{R}} x_{ref} N_{ref} \quad (9)$$

s.t.

$$z_{1,l} b_1 + z_{2,l} b_2 + \cdots + z_{n,l} b_n = x_{1,l} \quad \forall l \in \mathbf{L} \quad (10a)$$

$$z_{1,l} + z_{2,l} + \cdots + z_{n,l} = 1 \quad \forall l \in \mathbf{L} \quad (10b)$$

$$\sum_{f \in \mathbf{F}} \sum_{r \in \mathbf{R}} x_{ref} h_{r,f} \leq \mu_e \quad \forall e \in \mathbf{E} \quad (11)$$

$$x_{ref} \lambda_r \leq \omega_{ref} \mu_e \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \quad \forall r \in \mathbf{R} \quad (12)$$

$$x_{ref} h_{r,f} = \omega_{ref} \mu_e \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \quad \forall r \in \mathbf{R} \quad (13)$$

$$\sum_{r \in \mathbf{R}} \sum_{p \in \mathbf{P}} \lambda_r \zeta_{pl} \psi_{rp} \leq C_l \quad \forall l \in \mathbf{L} \quad (14)$$

$$\sum_{p \in \mathbf{P}} \psi_{rp} = 1 \quad \forall r \in \mathbf{R} \quad (15)$$

$$\sum_{e \in \mathbf{E}} \sum_{f \in \mathbf{F}} \sum_{p \in \mathbf{P}} x_{ref} \psi_{rp} v_{ep} \eta_{rf} = d_r \quad \forall r \in \mathbf{R} \quad (16)$$

$$\sum_{f \in \mathbf{F}} \sum_{r \in \mathbf{R}} \omega_{ref} A_f \leq R_e \quad \forall e \in \mathbf{E} \quad (17)$$

$$\sum_{e \in \mathbf{E}} \sum_{f \in \mathbf{F}} \sum_{p \in \mathbf{P}} \sum_{l \in \mathbf{L}} \frac{x_{ref}}{h_{rf} - \lambda_r} + \frac{\zeta_{pl} \psi_{rp}}{C_l - T_l} \leq T_r \quad \forall r \in \mathbf{E} \quad (18)$$

$$v_{ep}, \psi_{rp}, x_{ref}, \eta_{rf}, \zeta_{pl} \in \{0,1\} \quad h_{rf}, \lambda_r, \mu_e \geq 0 \quad (19)$$

To insure the piecewise linear approximation for the delay function , we define λ-formulation in constraint (10). To insure that the available capacity is not over-used in node e (11), (12) and (13) are added.To respect the capacity of link l (14) is added. Constraint(15) makes sure of the assignment of each request r to a path p. Constraint (16) insures that all required VNFs by a request r are assigned to edge clouds in path p. To insure that a service instance f is assigned to a node e with enough capacity, constraint (17) is defined. Constraint(18) insure the compatibility between the delay tolerance and request requirements.
To insure the linearization, we replace the product of $x_{ref} and \psi_{rp}$ with u_{refm} the remove the non linear term in constraint(16) as follows:

$$\sum_{e \in \mathbf{E}} \sum_{f \in \mathbf{F}} \sum_{p \in \mathbf{P}} u_{refp} v_{ep} \eta_{rf} = d_r \quad \forall r \in \mathbf{R} \quad (20)$$

Constraints (21) (22) and (23) force the binary variable u_{refp} to take the value of $x_{ref} \psi_{rp}$.

$$\sum_{p \in \mathbf{P}} u_{refp} \leq x_{ref} \quad \forall f \in \mathbf{F} \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad (21)$$

$$\sum_{p \in \mathbf{P}} u_{refp} \leq \psi_{rp} \quad \forall f \in \mathbf{F} \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad (22)$$

$$u_{refp} \geq x_{ref} + \psi_{rp} - 1 \quad \forall p \in \mathbf{P} \quad \forall f \in \mathbf{F} \quad \forall e \in \mathbf{E} \quad \forall r \in \mathbf{R} \quad (23)$$

C. Heuristics

In this section, we present a heuristic algorithm to address the computational complexity issue that increases with the requests number, the edge clouds, and the VMs that are available. For a larger number of instances, we solve the problem of VNFs placement and routing falling into NP-hard problems category [16].

Algorithm 1 HSFMR (Horizontal scaling)

```
Input Connected network graph G(N x N) - function chains set R (R x V)
Get $\lambda_r$, $\mu_e$ and $C_l$.
Get RC (R x 1) as the resources needed by the requests set
Get LRC (NxN) as the available resources of links
Get NRC (Nx1) as the available resources of nodes
Sort Desc R, NRC
Get VRR(VxR) as service instances required resources
Make SP(Rx2) as shortest paths list.
For each (r in RC)
  While ( All VNfs are not placed )
  {
    Update NRCcopy by NRC
    Update LRCcopy by LRC
    For each (v in VRR)
    For each (n in NRC)
    if (n $\subset$ SP and VRR $\leqslant$ NRCcopy (n) and RC $\leqslant$ SPmin)
      A function instance is assigned to an edge cloud
      change NRCcopy and LRCcopy
      Break;
    End For
    End For
    if (All functions in R are assigned)
      VNFsPlaced = true
    End If
    Next Shortest path;
  }
  Update NRC by NRCcopy
  Sort Desc NRC
  Update LRC by LRCcopy
EndFor
```

A service instance can be used by only one request. Therefore, one request packets can be hosts in one buffer. In Algorithm 1, we start by initializing the input parameters, also

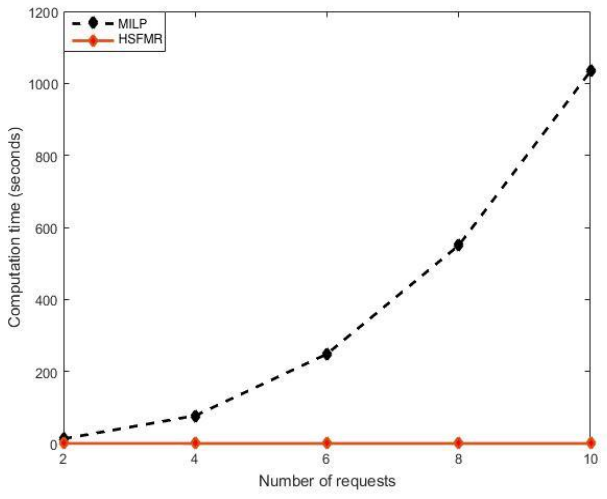

Fig. 3. Computational time (s)

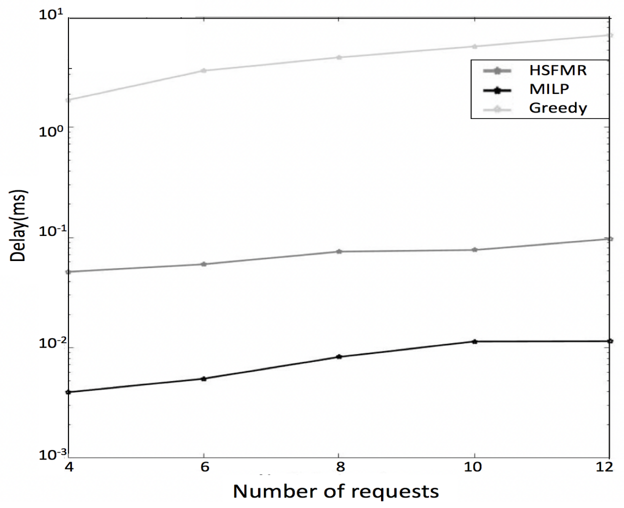

Fig. 4. Optimal vs. Heuristic

we define two shortest paths per request in a matrix. We follow the approach of Best Fit Decreasing (BFD), and we scale VMs horizontally when VNFs are assigned to edge clouds. If a minimum of two requests are using the same service hosted by an edge cloud, we assign as much VNF instances as needed to that specific node.

IV. Numerical Investigations

To evaluate the efficiency and effectiveness of our approach, we first evaluate the proposed algorithm for a 28 nodes topology as a connected graph [17] generated following random walk method [18].

We compare the optimal solution with the results of the proposed heuristic on a small scale scenario, then we compare the performance of the proposed HSFMR with a well known greedy algorithm based on First Fit Decreasing (FFD) presented in [6] for a larger number of request instances.

We assume that the remaining capacity of each node is between 70% and 90% and the transmission capacities of links vary from 2, 20, 200, 510 K or 2 M packet per second (pps) [6]. We define every request by a source and a destination node, and 3 functions from a list of VNFs: Intrusion Detection System (IDS), Load Balancer, Firewall, Virtual Private Network (VPN) function, Deep Packet Inspection (DPI). We fix two candidate paths per request and we define request arrival rates in a range varying from 1 to 100 pps and we fix a packet size to 500 bytes [19]. The VNFs have different processing rates for example a firewall function is 7.0771 pps and a VPN function processing time is 1.6385 pps [20].
While MILP takes 2.5 minutes to run for 4 requests composed of 3 functions, and approximately 6 hours compared to the heuristic algorithm that runs for 220 requests in 3 seconds. As illustrated in figure 3, MILP is limited by its scalability caused by the computation time [7], for a small number of instances where the number of requests varies from 4 to 12, the heuristic algorithm is approximately 100 times faster than the MILP which explain the scalability challenge we are facing for larger instances. We compare the performance the optimal solution with the heuristic as illustrated in figure 4 and we show that the gap between the results decreases while request number is increased. Therefore, we evaluate our suboptimal approach by increasing requests number and we compare it with a greedy algorithm taken from the state of art. Figure 5 shows that the average delay decreases by 65% compared to the greedy method. In contrast to the greedy algorithm, we first choose the route before assigning the VNFs to the appropriate node. As a result, we do not choose always the node with the highest available capacity which helps us to avoid bottleneck and balance the load through the network. For a larger topology scale, we fix the total of requests and increase the total of nodes to measure the effect of the network size on the overall delay. As illustrated in figure 6, having more available nodes, the inter-cloud delay decreases with the increase of the topology size. However, we note that the overall latency increases because of assigning requests to a longer path when the number of nodes goes from 24 to 100 nodes, which results in increasing the number of links as expected. Therefore, the link delay and the overall latency go up.

V. Conclusions

We have evaluated in the paper the performance of horizontal scaling. We have formulated VNFs placement and routing problem comparing the MILP with the proposed HSFMR heuristic results to validate our algorithm and we have shown that the proposed algorithm achieved a better delay compared with the greedy algorithm. When demand changes gradually take place, it is more appropriate to adopt horizontal scaling, but each VM required an operating system, therefore provisioning an additional VM when needed requires more time to set up. Additionally, while using horizontal scaling, a VM is considered as a scaling unit, this can cause unnecessary over-provisioning and high resource wastage [4]. As an extension to

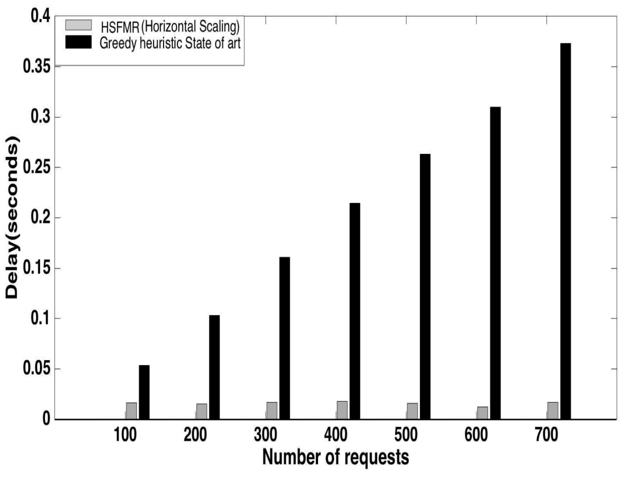

Fig. 5. HSFMR vs. greedy(state of art)

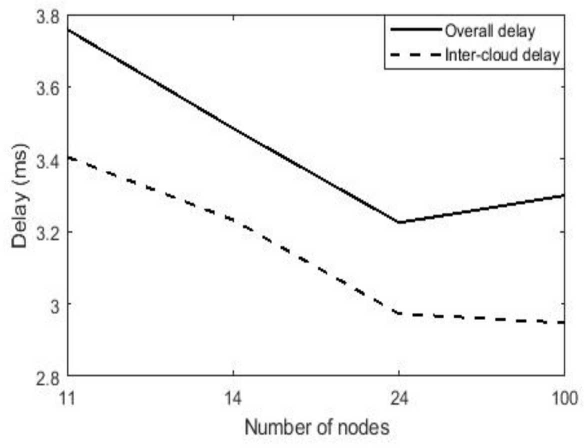

Fig. 6. Delay (Varying the number of edge clouds)

this work, we intend to investigate our approach by considering VNFs affinity with horizontal scaling and measure the effect on the overall delay in addition to minimizing VNFs placement financial cost.

References

[1] N. ETSI, “Gs nfv-eve 005 v1. 1.1 network function virtualisation (nfv); ecosystem; report on sdn usage in nfv architectural framework,” 2015.

[2] “Amazon virtual machine instances.” [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[3] “Google virtual machine instances.” [Online]. Available: https://cloud.google.com/compute/docs/instances/

[4] M. Turowski and A. Lenk, “Vertical scaling capability of openstack - survey of guest operating systems, hypervisors, and the cloud management platform,” in *ICSOC Workshops*, 2014.

[5] A. Gandhi, P. Dube, A. Karve, A. Kochut, and L. Zhang, “Modeling the impact of workload on cloud resource scaling,” in *2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing*, Oct 2014, pp. 310–317.

[6] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, “Optimal virtual network function placement in multi-cloud service function chaining architecture,” *Comput. Commun.*, vol. 102, no. C, pp. 1–16, Apr. 2017. [Online]. Available: https://doi.org/10.1016/j.comcom.2017.02.011

[7] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Wood, “Virtual function placement and traffic steering in flexible and dynamic software defined networks,” in *The 21st IEEE International Workshop on LANMAN*, April 2015, pp. 1–6.

[8] J. Bi, Z. Zhu, R. Tian, and Q. Wang, “Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center,” in *2010 IEEE 3rd International Conference on Cloud Computing*, July 2010, pp. 370–377.

[9] T. M. Pham and L. M. Pham, “Load balancing using multipath routing in network functions virtualization,” in *2016 IEEE RIVF*, Nov 2016, pp. 85–90.

[10] W. Rankothge, F. Le, A. Russo, and J. Lobo, “Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms,” *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 343–356, June 2017.

[11] “Network service orchestration standardization: A technology survey,” *Computer Standards and Interfaces*, vol. 54, pp. 203 – 215, 2017.

[12] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *2014 IEEE 3rd CloudNet*, Oct 2014, pp. 7–13.

[13] N. Makariye, “Towards shortest path computation using dijkstra algorithm,” in *2017 International Conference on IoT and Application (ICIOT)*, May 2017, pp. 1–3.

[14] D. Bertsekas and R. Gallager, *Data Networks (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[15] J. Bisschop, *AIMMS optimization modeling*. Lulu.com, 2006.

[16] F. Carpio, S. Dhahri, and A. Jukan, “Vnf placement with replication for loac balancing in nfv networks,” in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[17] K. A. Zweig, “Random graphs and network models,” in *Network Analysis Literacy*. Springer, 2016, pp. 149–181.

[18] W. Woess, *Random walks on infinite graphs and groups*. Cambridge university press, 2000, vol. 138.

[19] Q. Zhang, Y. Xiao, F. Liu, J. C. S. Lui, J. Guo, and T. Wang, “Joint optimization of chain placement and request scheduling for network function virtualization,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 731–741.

[20] 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.

[21] A. Gandhi, P. Dube, A. Karve, A. Kochut, and L. Zhang, “Modeling the impact of workload on cloud resource scaling,” in *Computer Architecture and High Performance Computing (SBAC-PAD), 2014 IEEE 26th International Symposium on*. IEEE, 2014, pp. 310–317.