Virtual Network Functions Routing and Placement for Edge Cloud Latency Minimization

Racha Gouareb, Vasilis Friderikos and A. Hamid Aghvami
Centre for Telecommunications Research, King's College London, UK
E-mail: {racha.gouareb, vasilis.friderikos, hamid.aghvami}@kcl.ac.uk

***Abstract*—As a new way to design, deploy and manage network services, network functions virtualization (NFV) decouples the network functions, from one or more physical network infrastructures and black boxes so they can run in software. It therefore comes as no surprise that NFV originated from service providers, who were looking to improve the deployment of new network services to support their revenue and growth objectives. Within the NFV ecosystem, high availability and low latency are one of the key QoS benefits that service providers can expect from the 5G Cloud and the NFV networks to make delay-critical services such as remote surgery a reality. Therefore, network services should be placed, chained and routed through the network considering users/tenants stringent quality of service (QoS) and service-level agreement (SLA) requirements. To this end, routing and placement optimization plays a major role in improving network performance and the overall network cost. In this paper, we study the problem of virtual network functions (VNFs) placement and routing across the physical hosts to minimize overall latency defined as the queuing delay within the edge clouds and in network links. In that respect, this work takes a holistic view by considering not only VNFs chaining and placement problem but also considering the flows routing aspect since these two problems are inter-related and have a major impact on network latency.**

***Index terms*— Network function virtualization, Virtual networks, 5G networks, Virtual Network function, Latency.**

I. INTRODUCTION

In recent years, communication networks have been witnessing an exponential growth in user data traffic as well as an increase in the use of virtualization technologies. The deployment of network resources, the maintenance of hardware appliances and the never ending race for marketing new services has resulted for the network operators an excessive operational expenditure (OPEX), the ongoing costs a company pays to run its basic services, and capital expenditure (CAPEX), the cost of expanding, upgrading and maintaining company's physical assets.

In order to process multiple operations simultaneously and balance the load among servers, moving toward data-centric models allow to reduce traffic congestion episodes and move closer to the end users. Therefore, network function virtualization gives users/tenants the ability to place and deploy network functions on the cloud.

To separate control and data, two complementary but independent concepts are introduced: software-defined networking (SDN) and network function virtualization (NFV) and have been studied during this last decade [1], [2]. SDN decouples the control and data planes and enables programming the behavior of the network using well-defined interfaces. As a complementary paradigm, NFV uses virtualization technology to run network functions on software that can be easily moved through different network locations, which reduces CAPEX, OPEX, space and power consumption.

The European telecommunications standards institute (ETSI) describes a high-level NFV framework composed of three principal domains; virtualized network functions (VNFs), NFV infrastructure (NFVI) and NFV management and orchestration (MANO). The group highlighted three key criteria to establish a high-level architecture framework: Decoupling as to complete the separation of hardware and software, flexibility in the automation and scalability of the network functions deployment, and dynamic operations in controlling the operational parameters of the network functions through control and monitoring the state of the network [3].

Cloud service providers such as Cisco, Google [4], Amazon [5] and Oracle own the virtual infrastructure and offer network services, infrastructure or applications from a shared infrastructure. Types of offered services can vary from Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) or host-based applications. This allows users to get the required services and deploy their application by paying only what they used, without any control or ownership. VNFs can be placed in different clouds and deployed on one or more Virtual Machines, different scenarios may require different scaling. For example, an application that needs to run faster in order to support more transactions per unit of time can be scaled vertically. On the other hand, horizontal scaling can be used for applications where the load can be spread across different virtual machines (VMs). In practise, a required service may be composed of more than one functions, and traffic should go through the chain of VNFs in pre-defined order to provide the required service. Therefore, the placement and routing of VNFs is based on the required resources and affect both the quality of service and the overall cost for offering to the end users a specific service.

As the number of requests, the edge clouds and available VMs per edge cloud increases the problem of allocating resources becomes combinatorial in its nature. To this end, and in order to find optimal decision policies we formulate the problem of a batch based network service chaining, routing, and placement. Based on the incoming virtual network requests and their requirements, such as delay toler-

†corresponding author: racha.gouareb@kcl.ac.uk

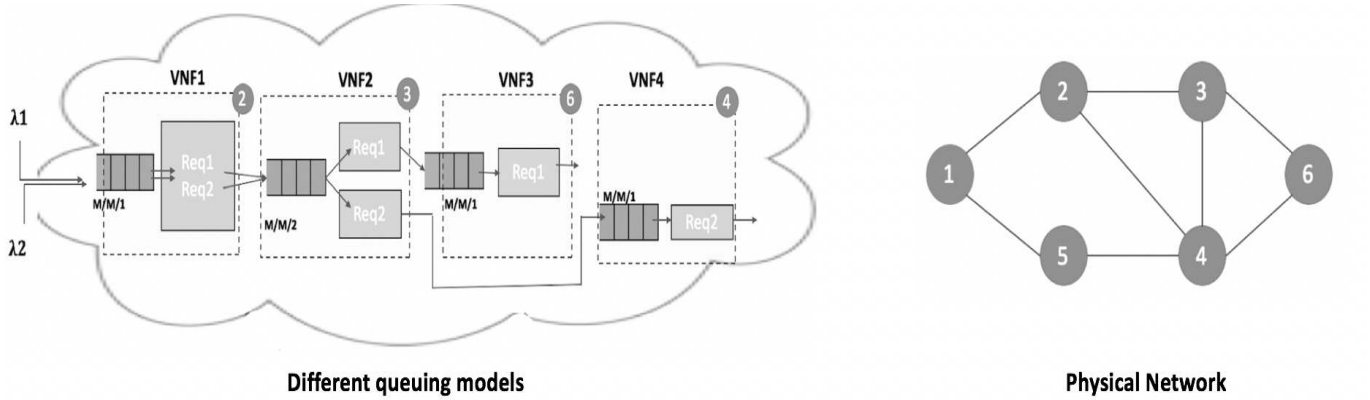

Fig. 1: VNFs routing and placement example

ance, we consider a network physical infrastructure where different virtual networks have to be set up. We develop a mathematical programming formulation using Mixed Integer Linear Programming (MILP) model to achieve an optimal solution, which consider vertical scaling for the purpose of minimizing network latency. For large networks, to accelerate the process of routing and placement, we use a scale-free heuristic algorithm in order to be able to provide a real-time allocation for a large number of requests. The proposed approach also takes into account the horizontal and vertical scaling of VMs. We also provide a performance comparison between the proposed heuristics and a simple greedy approach from the state-of-the-art [6].

The proposed model enables operators to increase the acceptance rate of strict delay requirement requests and reduce blocking requests due to capacity constraints.
The main logic behind the proposed algorithm can be seen in Figure 1 is a simple example of virtual network functions routing and placement in a small substrate network with 6 edge clouds is presented. We assume that we have two admitted requests, the related flows have different arrival rates. The Figure shows different ways of scaling that we consider in this work. VNF1 and VNF2 are both required by request 1 and 2. We use horizontal scaling and vertical scaling for VNF1 and VNF2 instances, respectively. As illustrated in Table I, we define a set of VNFs chains requests. The destination nodes are different, for request 1 and 2 destination nodes are nodes 6 and 2, respectively. In all these cases, node 1 is considered as the gateway. Each request has a specific computing resource and a bandwidth requirement, based on which we choose suitable physical links from two candidate paths selected in advance by k-shortest path algorithm. Then, we map VNFs following the algorithms presented in Section V, by giving higher priority to VNFs requesting higher levels of computing resources and assign them to the physical node with the highest remain carrying capability. If two requests are sharing the same edge cloud to get the same service, we can either share the same instance with a high processing capacity between the request using vertical scaling as in the case of sharing VNF_1 instance between request 1 and request 2 in node 2. In the case of VNF_2, we use horizontal scaling by creating two instances of VNF_2 in node 3, one instance for request 1 and the second instance for request 2. We also present different queuing models in the same example, M/M/1 queuing model when one processing unit is serving the incoming requests and an M/M/m model to capture the delay when having two or more processing units. The rest of the paper is organized as follows. In Section II, we discuss the related research work and describe the proposed model in Section III. A mathematical optimization model to minimize the overall latency is proposed in Section IV. We then propose a heuristic algorithm to solve the problem for larger instances scenario in Section V. In Section VI, we discuss the experimental set-up and different results comparing between optimal and different suboptimal results for performance evaluation.

II. RELATED WORK

Many previous research works have been focused in the area of VNF placement, chaining and routing by considering different metrics to increase network efficiency.
Different research works solved VM placement problem such as [7] presenting the Advanced Predictive Placement Algorithm where the best locations are defined as the less utilized and the closest to most of the user equipments considering the overload of VMs, the data overload, and the QoS.

In [8], the authors have considered an autonomic resource management framework for virtual networks. They have argued that to ensure reliability, availability, and QoS requirements, advanced features of service offerings have to take place via an automation and elasticity of resource distribution and allocation. They have introduced an autonomic and distributed virtual network management resource management based on a reinforcement learning algorithm in order that the agents can learn progressively to enhance the performance of the resource management in virtual networks.
An interesting approach is proposed in [9] solving the problem of joint service placement and traffic steering incrementally. The authors have formulated the VNF placement and routing problem with the objective of minimizing both link and core resource utilization. For this purpose, they have modeled the

TABLE I: Request requirements

Request Index	Candidate nodes	Requested functions	Carrying nodes	Candidate Paths
1	{2} {3} {4} {6}	1 2 3	{2} {3} {6}	$\{1 \rightarrow 2 \rightarrow 3 \rightarrow 6\}$ $\{1 \rightarrow 2 \rightarrow 4 \rightarrow 6\}$
2	{2} {3} {4}	1 2 4	{2} {3} {4}	$\{1 \rightarrow 2 \rightarrow 3 \rightarrow 4\}$ $\{1 \rightarrow 2 \rightarrow 4\}$

problem using mathematical programming aiming to provide efficient placement of service chains while considering latency as a constraint rather than an objective to minimize.
Similarly, in [10], the authors tackle the problem of VNF placement by considering two factors the paths between users and gateways in addition to features mobility. This paper presents different VNF placement algorithms, such as Avoiding S-GW Relocation (A-SGWR) algorithm, which aim to minimize the Serving Gateway (S-GW) relocation overhead in a delay-constrained network. The evaluation of this approach considered the delay of data packets delivery as one of the metrics. In [11], two efficient algorithms are presented to ensure the QoS and low cost deployment for vEPC/5G; one uses MILP to optimise the number of virtual resource instances of the different VNFs of vEPC/5G core network and the second algorithm is based on coalitional game to place these instances over a federated cloud. Chua et al. propose in [12] a Service Function Chain (SFC) provisioning system referred to Stringer which enables virtual network providers to minimize the infrastructure resources and end-to-end delay. Three methods are used for SFC provisioning system; a scalable round-robin heuristic, an optimization-based method and a queueing-theoretic model. This paper compares the performance of Mixed Integer Programming (MIP) with the heuristic method. The results show that the heuristic method outperforms the MIP significantly for the proposed system. However, this paper does not consider the routing cost.

A number of different approaches to the VNF placement problem consider delay as a requirement or observe the impact of different metrics on overall latency, with different optimization criteria such as reliability and load balancing. The authors in [13] investigate the problem of virtual placement for optimal service function chains (SFC) deployment across distributed clouds. The authors have solved SFCs deployment focusing on VNFs placement through an affinity-based heuristic and minimize inter-cloud traffic and response time in a multi-cloud scenario as an ILP optimization problem. In this work, the latency is described as the link and computational delays and modeled as M/D/1 and M/M/1 respectively.
The authors in [14] have presented an off-line approximation FAST-RACE algorithm for load balancing using multipath routing that decreases the latency and increases user demands. They have shown that using this method, the average delay of flows decreases about 26% and increase user demands around 14% compared with those of the hop-count weight vector method for load balancing.
The authors in [15] address the VNF scheduling problem and its respective resource optimization solutions. The authors have considered both VNF transmission and processing delays in this investigation. They have proposed a generic algorithm for solving the joint problem of VNF scheduling and virtual network resource allocation. They have evaluated the effectiveness of the proposed heuristic algorithm through a numerical method. They have shown that by dynamic allocation of bandwidth to virtual links shorter scheduling can be achieved.
The work in [16] proposes a new resource allocation algorithm to enable energy-aware Service Function Chaining (SFC) in Software Defined Networks (SDN)-based virtual networks. The authors have mathematically formulated the problems of resource allocation of VNFs to traffic flows and flow routing as optimization problems with the aim of minimizing energy consumption and network reconfiguration overhead. They have proposed new heuristic algorithms for the above-mentioned optimization problems. They have shown that the proposed heuristic algorithms can offer sub-optimal solution near to the optimal solution as long as minimization of energy consumption is concerned.
In [17] the authors aim to find the optimal route in Mobile Wireless Networks to minimize the total energy consumption. They model the problem as a joint optimization problem considering both the transmitting and receiving energy. The efficiency of their framework was evaluated on a real-life network dataset and validated by three algorithms considering different delay constraints, which revealed lower energy consumption, optimizing transmitting and receiving cost and showing a trade-off between delay and the receiving energy in mobile wireless networks. Bi, Zhu, Tian and Wang [18] aim to minimize the total number of VMs for a cluster-based three-tier virtualized applications by suggesting a flexible hybrid optimization. To do so, the authors have modeled the queue as a model of M/M/m system for the first tier and multiple M/M/1 for the remaining tiers. They have shown that under fine-grained resource provisioning, the optimum resource utilization can be achieved while maintaining average response time and request arrival time requirements.

In [6], the authors have considered inter-cloud latency and VNF response times to solve the problem of deploying SFCs as an ILP through an affinity-based heuristic. The latency is described as link delay and computational delay modeled as M/D/1 and M/M/1 respectively.
VMs are the most manageable entities sharing hardware resources [19] providing a number of benefits such as isolation from hardware and other VMs [18]. Those VMs are scalable to meet the requirements of users/tenants in a virtualized environment. Scaling can vary according to the operator requirements such as traffic load, application type and the amount of input

[20].
In [21], the authors propose an analytical model based on G/G/m queuing to estimate the mean response time of a VNF. The model can easily be extended to consider one or more service function chains. The validation of the model has been performed by computer simulation. The special case of the validation has been done for an LTE virtualized Mobility Management Entity (MME) with a three-tiered architecture. It has been shown that the proposed model has a computational complexity comparable to those used for analyzing Jacksons networks and the estimation error of the mean response time is much lower than those of the considered baseline systems. Rankothge et al. [22] have presented a resource allocation algorithm for VNFs based on Generic algorithms (GAs). They have carried out an extensive analysis of two GA algorithms for both initial placement of VNFs and the scaling of existing VNFs for supporting traffic variation. It has been shown that the proposed GA algorithms outperform Integer Linear Programming (ILP) resource allocation for a large number of VNFs in service function chains and number of virtual machines where ILP takes several hours to process while GP takes only a few milliseconds.
As far as we are aware, none of the above papers have considered the joint optimization problem of VNFs routing and placement in a multi-clouds scenario. In this work, we formulate the optimization model considering multiple instances of the virtual functions across different edge clouds to serve flows of packets considering three VNFs models: single-feature single-request, single-feature multi-requests, and multi-features multi-requests [23]. We develop an optimization model to reduce the inter-cloud and link latency. The inter-cloud queuing delay is modeled as M/M/1 or M/M/m and link delay is modeled as M/M/1. Later on, we present the problem as a Bin packing problem solved by a standard heuristic approach following Best-fit Decreasing (BFD) method. Additionally, we provide a performance comparison between heuristic and optimal solution and we compare the results of the proposed heuristic with those of random greedy.

III. PROBLEM DESCRIPTION

As already eluded above, the optimal VNFs chaining and routing problem is an area that has gained significant research attention and the problem itself falls within the NP-hard optimization problems. In this section, we set up the problem of minimizing the delay defined as inter-cloud and link queuing delay satisfying different constraints. We formulated the optimization model to route and assign VNFs to meet the service requests. All VNFs of a service request can be located at the same access/core location (the same edge cloud) or in different edge clouds. Furthermore, we assume that all the service requests are already admitted into the network, in other words, no admission control is considered in this work.
Data flows should visit different network functions depending on the required service, such as video optimizer, Deep Packet Inspection (DPI), Session Border Controller (SBCs) and Firewall in a specific order to be applied to the flow of data [24]. We define each service request as a chain of ordered functions similar to many research work such as in [25]. In our model, each request/service chain is associated with: a source node and a destination node in the network; a set of VNFs that needs to be executed on the flow and the arrival flow rate known in advance [26].
We assign a list of shortest path to each request using Dijkstra's algorithm [27] in a weighted network. The weights assigned to links are positive, pre-defined by a service provider and can be related to link bandwidth, average link delay, or even the required power for transmission. Shortest paths are sorted by cost, the first shortest path is assigned as far as there is enough capacity to host the VNFs and enough bandwidth to transmit the flow through VNFs executed by Virtual Machines (VMs). In other words, we assume a set of pre-defined multiple shortest paths between end users and edge clouds as well as the network gateway. Hence, any multiple shortest path algorithm can be used in the proposed framework.
Every edge cloud is a pool of physical resources that can be shared through different VMs. Different VMs might offer distinct performance and execute the same service, this can be due to the heterogeneity of hardware. Since a VNF instance can adapt its capacity as a function, VM can be scaled up or scaled down.
We consider the fact that edge clouds have different geographical locations linked between them; there is a traffic going between the nodes. For this, the capacity of links or edge clouds will be defined as the remaining capacity that can be used to solve VNFs routing and placement problem. We are going to compare three models [23] in this paper: single-feature single-request (SFSR) where a VNF instance can serve packets related to one request, single-feature multi-requests (SFMR) where a VNF instance, scaled up, can process more than one flow and multi-features multi-requests (MFMR) where different VNF instances, scaled horizontally, can process more than one flow all sharing the same buffer.
The delay in SFSR and SFMR can be modeled as M/M/1 queuing model where arrivals are determined by a Poisson process, in the same way, the delay in MFMR can be modeled as M/M/m for the reason that m VMs are processing the same function on different flows.

A. VMs scaling:

One way to scale virtual machines is vertical scaling (scaling up) that allow us to add more or less physical resources (CPU/Memory) to an existing virtual machine. This way of scaling allows us to resize the virtual machine by changing CPU or memory. Usually, vertical scaling requires downtime to add new resources and has defined limits by hardware.
On the other hand, horizontal scaling (scaling out) allow us to add more or less virtual entities to work as a single logical unit to adapt to network's load changes. Based on resource demands we can dynamically add or reduce the number of VMs.

B. Queuing theory

Traditionally, queuing theory is used to model servers and internet routers, to measure different metrics and improve

TABLE II: Description of queuing system notations

Parameter	Description
μ_e	Service rate (inverse of average service time)
λr	Arrival rate (inverse of average inter arrival time)
W_{ef}	Average customer waiting time in queue
ρ_{ef}	Ratio of arrival rate
P_{ef}	Probability that an arriving customer has to wait in queue
p^0_{ef}	Probability of no customers in the system

network performance [28][29]. In this work we provide an amalgamation of queueing theory with integer programming in order to optimize the overall delay. To this end, we utilize two queuing models methods; the M/M/1 which is used to model link queues, whereas servers queues in the edge clouds are modelled using the M/M/m model. The difference between the two models is that with the M/M/m we assume that there are m available resources to run VNFs (i.e., m VMs) in the system that are independent. Similarly to the M/M/1 model, arrivals and server's service time follows an exponential distribution with λ and μ parameters respectively. As defined in the queuing theory [30], the delay is formulated based on the definitions of the average processing time (2), the arrival rate (3), its ratio (4), the probability of a customer waiting in the queue (5) or none (6), and the average waiting time (7). The variables are summarized in TableII and the delay is formulated in equation (1) as:

$$D_{e,f} = \frac{1}{\mu_{ef}} + W_{ef} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{1}$$

To define the average waiting time W_{ef} of a packet in the queue (7) we define the average processing time and the arrival rate in equations (2) and (3) respectively.

$$\mu_{ef} = \frac{\sum_{i=1}^{m} \mu_{ief}}{\sum_{i=1}^{r} x_{ief}} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{2}$$

$$\lambda ef = \sum_{i=1}^{r} \lambda_i x_{ief} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{3}$$

$$\rho_{ef} = \frac{\lambda ef}{m\mu_{ef}} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4}$$

$$P_{ef} = \frac{p^0_{ef}(m\rho_{ef})^m}{m!(1-\rho_{ef})} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{5}$$

$$p^0_{ef} = \frac{1}{\sum_{k=0}^{m-1} \frac{(m\rho_{ef})^k}{k!} + \frac{(m\rho_{ef})^m}{m!(1-\rho_{ef})}} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{6}$$

$$W_{ef} = \frac{\rho_{ef} P_{ef}}{\lambda_{ef}(1-\rho_{ef})} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{7}$$

C. Virtual Network functions affinity:

Affinity and anti-affinity rules in NFV must be considered and added carefully in order to reduce communication costs between VNFs instances, ensure high availability, resilience, privacy and service performance [31]. In this context, two main aspects should be considered: modeling and describing the affinity rules and adapting the placement algorithm to respect the constraints [32].

Depending on the use case there might be instances where we need to place a pair of VNFs on the same edge-cloud (e.g., VNFs exchanging a big amount of data). In this case, we should define affinity constraints to place the two or more VNFs in the same host [33]. In other cases, anti-affinity rules are considered to allow critical VNFs to run on different nodes (e.g., in the case of failure, it will be better to have different instances of the same function placed on different edge clouds or different physical servers in the same edge cloud). Anti-affinity rules ensure the minimum cross interaction between VNFs running on the same server.

Based on the above discussion, We pre-initialize an affinity matrix that defines if two VNFs have a high affinity or a non-affinity relation. V_{ij} will be defining the affinity between VNF_i and VNF_j as follows:

$$V_{ij} = \begin{cases} 1 & \text{if } VNF_i \text{ and } VNF_j \text{ have a high affinity.} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

To ensure that affinity between VNFs wouldn't affect the performance of the network, we define the affinity constraint (9).

$$\sum_{i \in \mathbf{F}} \sum_{j \in \mathbf{F}} \sum_{r \in \mathbf{R}} V_{ij} x_{rei} x_{rej} = 1 \quad \forall e \in \mathbf{E} \tag{9}$$

IV. OPTIMIZATION MODEL

In this section, we present our model as a mathematical programming formulation where key notations are described in Table II and Table III. We set up the problem of minimizing inter-cloud and link delays in a multi-cloud scenario as a MILP optimization problem. The goal of our approach is to place VNFs, route workflows and assign client requests to these flows to meet the service demands. We present our objectives combined with constraints, then we elaborate on how we linearized the objective function and nonlinear constraints.

A. Optimization Function

We will be modeling edge-cloud and link delay in the following subsections using the M/M/1 queuing model to optimize the delay. Packets for each flow request are enqueued in every edge cloud waiting to get processed by available VMs, and then in another queue to get transmitted through network links. Each Edge cloud can have none, one or different queues, depending on the number of VNF instances assigned to it.

We consider the M/M/1 queuing model for both link traffic and inter-cloud traffic where one VM is available to serve incoming traffic requiring a specific function.

We define the decision variable x to define the allocation of VNFs instances composing every service request, such that x_{ref}=1 or x_{ref}=0. The value 1 will be assigned if VNF f of a

request r is assigned to an edge cloud e consuming a portion of its resources: memory, network, and computing available resources. Decision variable ψ represents the assignment of one path p to one or different requests, $\psi_{rp} = 1$ means that request r will use path p and go through all nodes and links belonging to path p to get the requested services. Requests might share the same paths or some links belonging to the same path. Both variables v_{ep} and ζ_{pl} define the nodes and the links belonging to path p respectively.

$$x_{ref} = \begin{cases} 1 & \text{if flow related to request } r \text{ go} \\ & \text{through edge cloud } e \text{ for VNF } f. \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

$$\psi_{rp} = \begin{cases} 1 & \text{if request } r \text{ use SP } p. \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

$$v_{ep} = \begin{cases} 1 & \text{if edge cloud } e \in \text{ path } p. \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

$$\zeta_{pl} = \begin{cases} 1 & \text{if link } l \in \text{ path } p. \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

Using Little's Theorem [30], we define the delay in the inter-cloud in Equation 14 where h_{rf} is the processing capacity and λ_r is the arrival rate.

$$N_{ref} = \frac{1}{h_{rf} - \lambda_r} \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \quad (14)$$

Similarly, we define The link delay as follows, where C_l is the capacity of the link and T_l is the total traffic in link l.

$$L_l = \frac{1}{C_l - T_l} \quad \forall l \in \mathbf{L} \quad (15)$$

In order to minimize the overall delay from the gateway to the end-users, we solve the MILP formulation modeling both inter-cloud and link queuing delays. We use mixed integer linear program (MILP) Matlab tool and formulate the objective function in the equation(16) and applicable constraints in equations(17-26) based on the above definitions, the mathematical problem can be formulated as follows:

$$\min_{\psi_{rp}, x_{ref}} \sum_{l \in \mathbf{L}} \sum_{n \in \mathbf{N}} z_{ln} L_l(b_n) + \sum_{r \in \mathbf{R}} \sum_{e \in \mathbf{E}} \sum_{f \in \mathbf{F}} x_{ref} N_{ref} \quad (16)$$

$$\text{s.t. } z_{1,l} b_1 + z_{2,l} b_2 + \cdots + z_{n,l} b_n = x_{1,l} \quad \forall l \in \mathbf{L} \quad (17a)$$

$$z_{1,l} + z_{2,l} + \cdots + z_{n,l} = 1 \quad \forall l \in \mathbf{L} \quad (17b)$$

$$\sum_{r \in \mathbf{R}} \sum_{f \in \mathbf{F}} x_{ref} h_{r,f} \leq \mu_e \quad \forall e \in \mathbf{E} \quad (18)$$

$$x_{ref} \lambda_r \leq \omega_{ref} \mu_e \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \quad (19)$$

$$x_{ref} h_{r,f} = \omega_{ref} \mu_e \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \quad (20)$$

$$\sum_{r \in \mathbf{R}} \sum_{p \in \mathbf{P}} \lambda_r \zeta_{pl} \psi_{rp} \leq C_l \quad \forall l \in \mathbf{L} \quad (21)$$

$$\sum_{p \in \mathbf{P}} \psi_{rp} = 1 \quad \forall r \in \mathbf{R} \quad (22)$$

$$\sum_{e \in \mathbf{E}} \sum_{f \in \mathbf{F}} \sum_{p \in \mathbf{P}} x_{ref} \psi_{rp} v_{ep} \eta_{rf} = d_r \quad \forall r \in \mathbf{R} \quad (23)$$

$$\sum_{f \in \mathbf{F}} \sum_{r \in \mathbf{R}} \omega_{ref} A_f \leq R_e \quad \forall e \in \mathbf{E} \quad (24)$$

$$\sum_{f \in \mathbf{F}} \sum_{e \in \mathbf{E}} \sum_{l \in \mathbf{L}} \sum_{p \in \mathbf{P}} \frac{x_{ref}}{h_{rf} - \lambda_r} + \frac{\zeta_{pl} \psi_{rp}}{C_l - T_l} \leq T_r \quad \forall r \in \mathbf{E} \quad (25)$$

$$v_{ep}, \psi_{rp}, x_{ref}, \eta_{rf}, \zeta_{pl} \in \{0,1\} \quad h_{rf}, \lambda_r, \mu_e \geq 0 \quad (26)$$

B. Variables and Parameters

TABLE III: Description of variables

Parameter	Domain	Description
$h_{r,f}$	$h_{r,f} \in \mathbb{R}_{>0}$	processing rate of function f for request r
$N_{r,e,f}$	$N_{r,e,f} \in \mathbb{R}_{>0}$	Delay in edge cloud e related to function f in edge cloud e
L_l	$L_l \in \mathbb{R}_{>0}$	Delay in link l
T_l	$T_l \in \mathbb{R}_{>0}$	Total traffic in link l
C_l	$C_l \in \mathbb{R}_{>0}$	Remain Capacity of link l
S_p	$S_p \in \mathbb{R}_{>0}$	Cost of a path p
r	$r \in R$	set of requests
l	$l \in L$	set of links
p	$p \in P$	set of paths
e	$e \in E$	set of Edge clouds
f	$f \in F$	set of functions
n	$n \in N$	set of breaking points
η_{rf}	$\eta_{rf} \in \{0,1\}$	Equal to 1 if request r requests function f
x_{ref}	$x_{ref} \in \{0,1\}$	Equal to 1 if request r goes through edge cloud e for function f
ψ_{rp}	$\psi_{rp} \in \{0,1\}$	Equal to 1 request r is assigned to shortest path p
v_{ep}	$v_{ep} \in \{0,1\}$	Equal to 1 if edge cloud e belong to shortest path p
ω_{ref}	$\omega_{ref} \in [0,1]$	Integer variable defining the utilization of an edge cloud e capacity to host a VNF instance f for a request r
d_r	$d_r \in \mathbb{R}_{>0}$	Number of functions required by request r
μ_e	$\mu_e \in \mathbb{R}_{>0}$	Average processing rate of an edge cloud e
λ_r	$\lambda_r \in \mathbb{R}_{>0}$	Arrival rate in an edge cloud e to go through a function f
ζ_{pl}	$\zeta_{pl} \in \{0,1\}$	Equal to 1 if link l belong to shortest path p
A_f	$A_f \in \mathbb{R}_{>0}$	Resource demand of each service instance of VNF f
R_e	$R_e \in \mathbb{R}_{>0}$	Resource capacity of a computing node e
T_r	$T_r \in \mathbb{R}_{>0}$	Delay tolerance of a request r

C. Explanation of the optimization problem constraints

Constraints (17) ensure the piecewise linear approximation [34] for the delay function using λ-formulation. Constraints (18), (19) and (20) ensure that the available capacity of the edge cloud e is not exceeded. In a similar manner, constraint (21) ensures that the link capacity is not exceeded. Constraint (22) enforces that each request r to be assigned to only one routing path p. Constraint (23) ensures that when a request r is using a path p, all VNFs required for this request are mapped into edge nodes belonging to that chosen path p. Constraint (24) makes sure that a VNF f is placed at an edge cloud e with sufficient resource capacity. Finally, constraint (25) considers each request requirement in terms of delay tolerance.

D. Linearization of the Proposed MILP

In order to linearize the optimization problem, in constraint (23), we replace the product of two binary decision variables $x_{ref}\psi_{rp}$ with a binary variable u_{refm} where u_{refp}=$x_{ref}\psi_{rp}$.

To linearize the constraint (23), we eliminate the non-linear term $x_{ref}\psi_{rp}$ by replacing the product as follow:

$$\sum_{e\in\mathbf{E}}\sum_{f\in\mathbf{F}}\sum_{p\in\mathbf{P}} u_{refp}v_{ep}\eta_{rf} = d_r \quad \forall r\in\mathbf{R} \tag{27}$$

Note that constraints (28) (29) and (30) force the binary variable u_{refp} to take the value of $x_{ref}\psi_{rp}$.

$$\sum_{p\in\mathbf{P}} u_{refp} \leq x_{ref} \quad \forall r\in\mathbf{R} \quad \forall e\in\mathbf{E} \quad \forall f\in\mathbf{F} \tag{28}$$

$$\sum_{p\in\mathbf{P}} u_{refp} \leq \psi_{rp} \quad \forall r\in\mathbf{R} \quad \forall e\in\mathbf{E} \quad \forall f\in\mathbf{F} \tag{29}$$

$$u_{refp} \geq x_{ref} + \psi_{rp} - 1 \quad \forall r\in\mathbf{R} \quad \forall e\in\mathbf{E} \quad \forall f\in\mathbf{F} \quad \forall p\in\mathbf{P} \tag{30}$$

V. Heuristic Based Algorithms

In this part, we present different heuristic approaches that address the same goals as the optimal one. We are using heuristic approaches to generate competitive suboptimal solutions that are able to scale with the size of the problem and reduce the computational complexity. In the first place, we compare the proposed heuristic with a random greedy method, and in the second place with a greedy approach based on FFD (first-fit decreasing method). Algorithm 5 shows the steps for the greedy, before we iterate through all required VNFs instances, we group the VNFs needed to satisfy all requests in one list. Respecting capacity constraints, we allocate VNFs instances at the appropriate node, placing a maximum number of instances in the chosen node, before we move to the next node. The second step of the algorithm consists of assigning every request to a node or multiple nodes hosting the required VNF(s) and satisfying the capacity constraints. We finally define the routing path, following the shortest path approach, to link between the gateway, the chosen nodes, and the destination node.

A. Initialization algorithm

Algorithm 1 describes the steps to initialize all input parameters and completing the routing processing step. We first generate a random connected graph where each edge cloud and every link is defined by its remain capacity. We assume a number of accepted service requests, each defined by an arrival rate a source and destination. VNFs offer different services and process differently, for this we set up the required amount of resources to instantiate every VNF. Since the routing algorithm is the same for horizontal and vertical scaling, we set up a matrix of shortest paths. For each service demand, we calculate the 3 first shortest paths[35] with enough capacity to handle the related traffic flow. This pre-processing is common to heuristic algorithms presented in the following sub-sections.

Algorithm 1 Input Parameter

Require: A connected edge clouds topology G(N x N) and list of requests R(R x V)
AllVnfsPlaced = false
Read λ_r, μ_e and C_l.
Read RC(R x 1) as requests required capacity
Sort R Desc
//N= Number of Edge clouds
Read LRC(N x N) as Links remain capacity
Read NRC(N x 1) as Nodes remain capacity
Sort NRC Desc
//V= Number of VNFs instances
Read VRR(V x R) as VNFs instances required capacity
//P= Number of shortest paths
Construct SP(R x P) as shortest paths list.
Construct SPmin(R x P) as minimum link bandwidth in the path

Algorithm 2 HSFSR & HMFMR (Horizontal scaling)

```
Foreach (Request r in RC)
  While (AllVnfsPlaced = false)
{
    Copy NRC in NRCcopy
    Copy LRC in LRCcopy
    Foreach (VNFs v in VRR(v,r))
    Foreach (Node n in NRC)
    if (n ⊂ SP(r,p) and VRR(v,r) ⩽ NRCcopy(n) and
RC(r) ⩽ SPmin(r,p))
      Assign VNF instance v to Node n
      Update NRCcopy and LRCcopy
      Break;
    EndFor
    EndFor
    if (All VNFs of R are placed)
      AllVnfsPlaced = true
    EndIf
    Go to next Shortest path;
}
  Copy NRCcopy in NRC
  Sort NRC Desc
  Copy LRCcopy in LRC
EndFor
```

For the next step, we are using the initialization algorithm as an input. We will be defining 3 algorithms: a heuristic algorithm using horizontal scaling, a heuristic algorithm for vertical scaling and a Random fit greedy algorithm where the random assignment will help us to measure the impact of our approach on latency.

B. Horizontal scaling algorithm

Algorithm 2 describes the steps for the heuristic used for two different cases. In the first case of SFSR, each VNF instance may serve only one customer at the same time.

Algorithm 3 HSFMR (Vertical scaling)

```
Foreach (Request r in RC)
  While (AllVnfsPlaced = false)
{
    Copy NRC in NRCcopy
    Copy LRC in LRCcopy
    Foreach (VNFs v in VRR(v,r))
      Foreach (Node n in NRC)
      if (n ⊂ SP(r,p) & VRR(v,r) ⩽ NRCcopy(n) & RC(r)
⩽ SPmin(r,p))
          Assign VNF instance v to Node n
          If (VNF f have been assigned to Node n )
            Assign more resources to VM
          else
            New VM will process VNF f
          endIf
          Update NRCcopy and LRCcopy
          Break;
        EndFor
    EndFor
    if (All VNFs of R are placed)
      AllVnfsPlaced = true
    EndIf
    Go to next Shortest path;
}
  Copy NRCcopy in NRC
  Sort NRC Desc
  Copy LRCcopy in LRC
EndFor
```

We follow an M/M/1 queuing model, where the buffer hosts packets related to one request, in other words, we will have one queue per VNF instance per flow(s) related to one request. In the second case of MFMR, VNFs instances can serve different customers, different service requests sharing the same VNF. We follow an M/M/m queuing model [30] and we group VMs per function type, e.g. packets related to flows requiring the same service and assigned to the same node will share the same buffer. Furthermore, the number of VNFs instances will be the same as the number of different service requests. The heuristic iterates through all requests, for each we iterate through all requested VNFs in order to place the one with the highest resources demands to the edge cloud with the highest remain capacity following the BFD approach. At the end of every iteration, if not all the VNFs of a specific request are placed we start over using the next available path. The main difference between HSFSR and HMFMR algorithms is in the cost measurement (calculation of the inter-cloud delay).

C. Vertical scaling algorithm

In Algorithm 3, each VNF instance can serve different requests at the same time. Therefore one buffer will be hosting different flows queuing to have a similar processing by the same VNF instance. Similarly to the first heuristic, we are following the same approach of BFD. After assigning the VNFs to edge clouds, VMs are scaled vertically in order to serve different flows related to different requests. In the case two or more requests have a function in common and have been assigned to the same edge cloud, they will share the same VM (the same VNF instance). Instead of assigning a VNF to another VM in the same edge cloud, we will be assigning more resources to the same VM to be shared. To cope with a higher number of demands without creating additional VMs. This type of scaling can be used to avoid VMs under-utilization.

Algorithm 4 Randomized heuristic algorithm

```
Foreach (Request r in RC)
    Choose a path from the SR list
  While (AllVnfsPlaced = false)
{
    Copy NRC in NRCcopy
    Copy LRC in LRCcopy
    Foreach (VNFs v in VRR(v,r))
    Scan NRCcopy for a node to accommodate the VNF
instance/VM
    if (such node is found)
      Assign VNF instance v to Node n
      Update NRCcopy and LRCcopy
      Break;
    EndFor
    if (All VNFs of R are placed)
      AllVnfsPlaced = true
    EndIf
    Go to next Shortest path;
}
  Copy NRCcopy in NRC
  Copy LRCcopy in LRC
EndFor
```

Algorithm 5 Greedy heuristic

```
Foreach (VF instance v in V)
Foreach (edge cloud e in NRC)
  While (AllVnfsPlaced = false and NRC(e) ≥ VRR(v) )
  {
    if (constraints are satisfied and node has enough
      capacity)
      Assign VNF instance v to Node e
    EndIf
  }
EndFor
EndFor

Foreach (Request r in RC)
Foreach (edge cloud e in NRC)
Foreach (VF instance v in V)
    if (v instance is installed in edge cloud e)
      Assign request r to Node e
    EndIf
EndFor
EndFor
EndFor
```

TABLE IV: Virtual processing times of virtual network functions used in our evaluation

Network Function	Processing time
Load Balancer	0.647.5 pps
Firewall	7.0771 pps
VPN Function	1.6385 pps

D. Random Placement and routing algorithm

Additionally, with the random routing and placement algorithm, for each service demand, the algorithm selects one path randomly to be assigned to a request. Then randomly choose one of the nodes with sufficient capacity for the placement of VNFs. The output solution will be respecting the constraints defined in the LP approach and compared with the heuristic algorithm outputs.

VI. EXPERIMENTAL SETUP AND RESULTS

In this section, we observe and analyze the behavior of the proposed heuristic based on three different models: SFSR, SFMR, and MFMR. We compare suboptimal approaches and present heuristic based SFSR (HSFSR), heuristic-based SFMR (HSFMR) and heuristic MFMR (HMFMR) results, we also compare their results with the MILP based solution. Furthermore, we show that the proposed heuristics allow us to increase the size of the problem solved compare to the MILP based solution. Thus, they allow a scale-free operation, amenable to run in large network topologies with an increased number of requests.

For instance, we have evaluated our approach on a random 28 nodes topology network, each with total CPU capacity of 100%. To build our topology, we consider one gateway and several destination edge clouds in a random connected graph, having n possible vertices and N edges, chosen randomly with equal probabilities edges [36][37]. To build the graph we are following the theory of random Walk [38], where we select a starting point node, we select a neighbor of it at random and move to this neighbor; then we select a neighbor node of this point at random, and move to it. Each request is defined by a source gateway to a specific destination node in the graph.

For simplicity, we assume that a packet size is 500 bytes, a request arrival rate is assumed to be 1 to 100 packets per second (pps) [39] and VNFs instances have different processing service rates [37] as illustrated in Table IV. We vary link transmission capacities randomly from {2, 20, 200, 510} Kpps or 2 Mpps [13].

The proposed MILP framework takes 144 seconds to find optimal solutions for 4 service chains composed of 3 services chosen randomly from a list of 5 different VNFs (Load Balancer, Firewall, Intrusion Detection System (IDS), Deep packet inspection (DPI), virtual private network (VPN) function) where 4 GB RAM used by the optimization solver, while the heuristic methods take just 3 seconds to run for 220 requests. To compare the optimal solutions coming from the MILP framework with the heuristic results, we focus on small scale scenarios. This is because, as expected, integer programming suffers from the curse of dimensionality, hence the calculation time increases exponentially with a linear

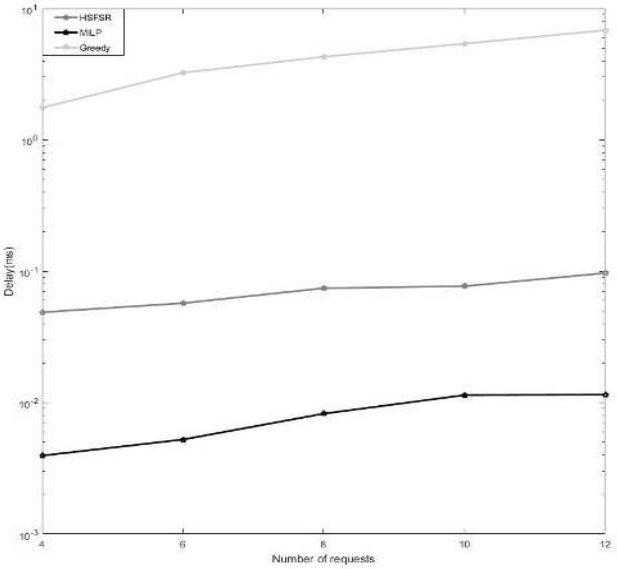

Fig. 2: Heuristic vs. Optimal (varying number of requests)

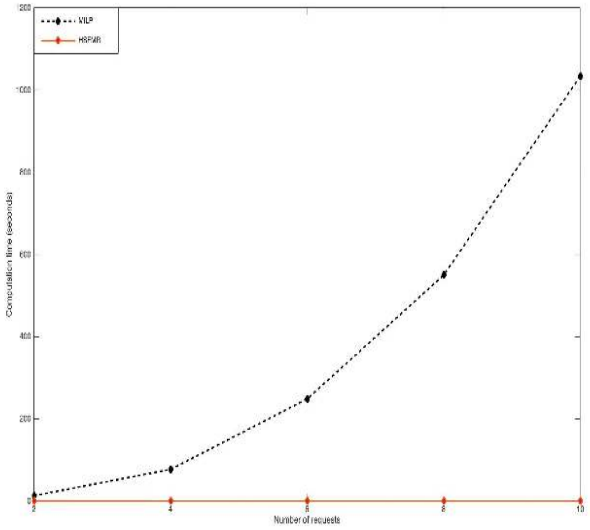

Fig. 3: Computational time (s)

increase of the size of the problem [9]. To investigate the impact of a number of requests, considered as accepted, on the execution time in the proposed model, we plot the average execution time for a different number of requests varying from 2 to 10. We measured the computational time for both heuristic and MILP as illustrated in Figure 3. The results show that the heuristic is 700 to 1000 times faster than MILP and this is the case for the considered small scale scenario. Therefore using MILP on larger network instances can be deemed as prohibited and this explains the scalability challenge we are facing. Figure 2 shows that MILP results are better but close to the heuristic for a number of requests varying from 2 to 12 service chains. We note that the optimality gap is kept low and therefore from these results we can assume that the heuristic

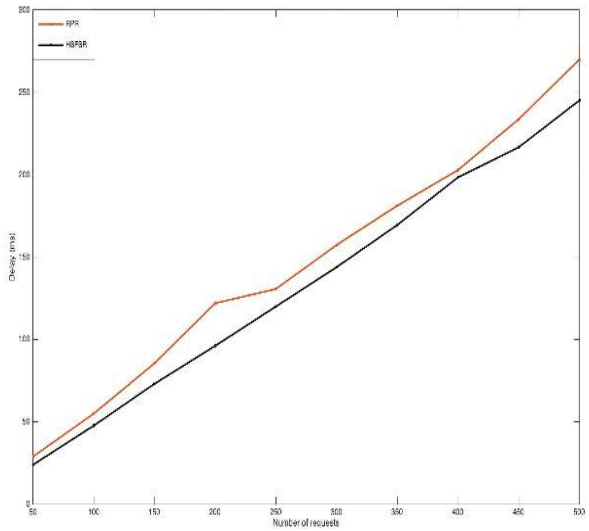

Fig. 4: Heuristic vs. Random greedy

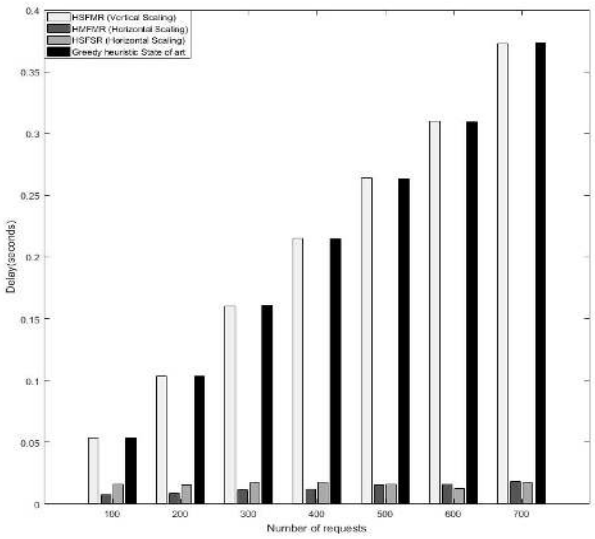

Fig. 6: Comparison with the state of art

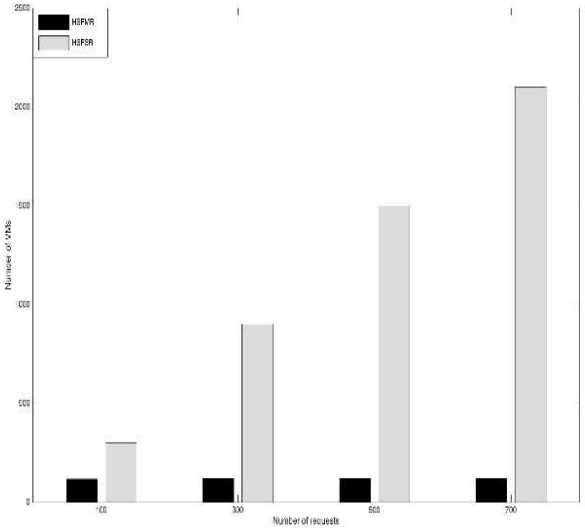

Fig. 5: HSFSR vs. HSFMR (varying number of requests)

based algorithms can find competitive solutions.

Figure 4 shows that our approach can decrease the average delay significantly compared with the randomized solution.

To observe the behavior of the proposed approach, we have evaluated the heuristics on a large scale network scenario. The results of HSFMR show that with vertical scaling, we can decrease the delay by approximately 70% and 35% compared to horizontal scaling in HSFSR and HMFMR approaches, respectively. Sharing a VM instance between different chains with the same processing requirements decreases the delay and the number of VMs from 300 to 78 VMs in a random 28 edge clouds network while serving 100 requests as shown in Figure 5.

Figure 7 shows the average delay of data flows when demand varies between 100 and 500. We observe the expected growth of the delay measured in HSFR, HSFMR, and HMFMR as the number of requests increases. As in the case of vertical scaling, sharing a VM instance between different request helps to increase the utilization of the processing unit which reduces the edge cloud delay but increases the link delay. The results from the greedy algorithm in the state of the art are very close to the results of those of vertical scaling and horizontal scaling but still the results of our heuristic give a better performance. Compared to the proposed greedy in the state of art, the delay is decreased by approximately 65% in the case of horizontal scaling and a little 1% using vertical scaling since the inter-cloud delay is the same but the link delay is different. In our approach, we choose first the best shortest path before assigning the requests to the edge clouds, but in the simple greedy we first assign the VNFs instances of the requests before routing. The first available node is loaded before no capacity is remained before going to the next one which might cause a link bottleneck and increase the delay in that specific link. Our approach helps us to balance the load over the available resources and avoid a bottleneck.

VII. Conclusions

In this paper, we first formulated the VNF placement and routing problem as a non-linear integer mathematical program and subsequently we have linearized the mathematical formulation in order to utilize powerful mixed integer mathematical solvers. The proposed approach is shown to be useful in founding optimal solutions for small to a medium number of instances. This allowed comparing the performance of a number of scale-free heuristic algorithms and validating proposed schemes whilst evaluating the incurred delay for different models.

The results show that the MFMR approach allows us to meet a stringent latency requirement for horizontal scaling, this

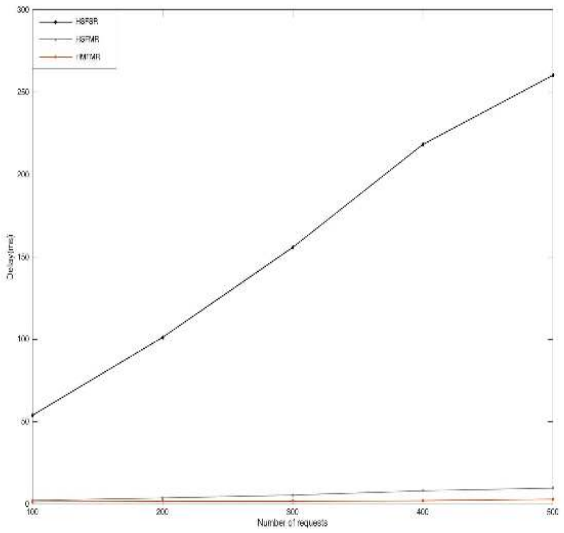

Fig. 7: Comparing heuristics

reduces the delay by 18% when serving 100 requests compared to HSFMR results. Scaling out VMs can provide in general a better performance in terms of delay, it also avoids us to put the machine in an off-line state to upgrade it for peak demand. Likewise, vertical scaling allows us to optimize latency, but increasing virtual resources online and dynamically might be a problem for different tenants since an interruption of the ongoing process is necessary and such a change in the system should be planned in advance. An interesting future avenue of research is to investigate the proposed approach by considering also request admission control and VNFs affinity rules to increase the network performance and take into account potentially other network metrics.

ACKNOWLEDGEMENT

This work has been partially funded by the EC H2020-ICT-2014-2 project 5G NORMA. For details regarding the scope of this project please visit: `www.5gnorma.5g-ppp.eu`

REFERENCES

[1] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2014, pp. 2402–2407.

[2] T. Taleb and A. Ksentini, "Gateway relocation avoidance-aware network function placement in carrier cloud," in *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, ser. MSWiM '13. New York, NY, USA: ACM, 2013, pp. 341–346. [Online]. Available: http://doi.acm.org/10.1145/2507924.2508000

[3] ETSI, "Gs nfv 002-v1. 1.1-network function virtualisation (nfv)-architectural framework," *October*, 2013.

[4] "Google virtual machine instances." [Online]. Available: https://cloud.google.com/compute/docs/instances/

[5] "Amazon virtual machine instances." [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[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," *Computer Communications*, vol. 102, pp. 1–16, 2017.

[7] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: Vnf placement algorithms for a dynamic amp;amp; realistic edge cloud environment," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[8] R. Mijumbi, J. Serrat, and J.-L. Gorricho, "Autonomic resource management in virtual networks," *arXiv preprint arXiv:1503.04576*, 2015.

[9] 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.

[10] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5g network infrastructure," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 3879–3884.

[11] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, and H. Flinck, "Coalitional game for the creation of efficient virtual core network slices in 5g mobile systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 469–484, March 2018.

[12] F. C. Chua, J. Ward, Y. Zhang, P. Sharma, and B. A. Huberman, "Stringer: Balancing latency and resource usage in service function chain provisioning," *arXiv preprint arXiv:1604.08618*, 2016.

[13] 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

[14] 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.

[15] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3746–3758, Sept 2016.

[16] M. M. Tajiki, S. Salsano, M. Shojafar, L. Chiaraviglio, and B. Akbari, "Joint energy efficient and qos-aware path allocation and VNF placement for service function chaining," *CoRR*, vol. abs/1710.02611, 2017. [Online]. Available: http://arxiv.org/abs/1710.02611

[17] L. Fu, X. Fu, Z. Zhang, Z. Xu, X. Wu, X. Wang, and S. Lu, "Joint optimization of multicast energy in delay-constrained mobile wireless networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 633–646, Feb 2018.

[18] 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.

[19] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "Ease: Epc as a service to ease mobile core network deployment over cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, March 2015.

[20] Z. A. Mann, "Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 11:1–11:34, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2797211

[21] J. Prados-Garzon, P. Ameigeiras, J. J. Ramos-Munoz, P. Andres-Maldonado, and J. M. Lopez-Soler, "Analytical modeling for virtualized network functions," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 979–985.

[22] 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.

[23] A. Leivadeas, M. Falkner, I. Lambadaris, and G. Kesidis, "Optimal virtualized network function allocation for an sdn enabled cloud," *Computer Standards & Interfaces*, vol. 54, pp. 266–278, 2017.

[24] 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.

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

[26] M. M. Tajiki, S. Salsano, M. Shojafar, L. Chiaraviglio, and B. Akbari, "Joint energy efficient and qos-aware path allocation and VNF placement for service function chaining," *CoRR*, vol. abs/1710.02611, 2017. [Online]. Available: http://arxiv.org/abs/1710.02611

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

[28] G. Huang, S. Wang, M. Zhang, Y. Li, Z. Qian, Y. Chen, and S. Zhang, "Auto scaling virtual machines for web applications with queueing theory," in *2016 3rd International Conference on Systems and Informatics (ICSAI)*, Nov 2016, pp. 433–438.

[29] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach (6th Edition)*, 6th ed. Pearson, 2012.

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

[31] H. Zhu and C. Huang, "Cost-efficient vnf placement strategy for iot networks with availability assurance," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sept 2017, pp. 1–5.

[32] N. Bouten, R. Mijumbi, J. Serrat, J. Famaey, S. Latr, and F. D. Turck, "Semantically enhanced mapping algorithm for affinity-constrained service function chain requests," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 317–331, June 2017.

[33] M. C. Luizelli, D. Raz, and Y. Saar, "Optimizing nfv chain deployment through minimizing the cost of virtual switching."

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

[35] P. Key, L. Massoulie, and D. Towsley, "Combining multipath routing and congestion control for robustness," in *2006 40th Annual Conference on Information Sciences and Systems*, March 2006, pp. 345–350.

[36] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.

[37] 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.

[38] L. Lovász *et al.*, "Random walks on graphs: A survey," *Combinatorics, Paul erdos is eight 2.1*, 1993.

[39] 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.

[40] 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.