Scheduling Service Function Chains for Ultra-Low Latency Network Services

Hyame Assem Alameddine[1], Long Qu[2], and Chadi Assi [1]
[1]Concordia University, Canada
[2]Ningbo University, China

***Abstract*—The fifth generation (5G) of cellular networks is emerging as the key enabler of killer real-time applications, such as tactile Internet, augmented and virtual reality, tele-driving, autonomous driving, etc., providing them with the much needed ultra-reliable and ultra-low latency services. Such applications are expected to take full advantages of recent developments in the areas of cloud and edge computing, and exploit emerging industrial initiatives such as Software Defined Networks (SDN) and Network Function Virtualization (NFV). Often, these 5G applications require network functions (e.g., IDSs, load balancers, etc.) to cater for their end-to-end services. This paper focuses on chaining network functions and services for these applications, and in particular considers those delay sensitive ones. Here, we account for services with deadlines and formulate the joint problem of network function mapping, routing and scheduling mathematically and highlight its complexity. Then, we present an efficient method for solving these sub-problems sequentially and validate its performance numerically. We also propose and characterize the performance of a Tabu search-based approach that we design to solve the problem. Our numerical evaluation reveals the efficiency of our sequential method and the scalability of our Tabu-based algorithm.**

I. Introduction

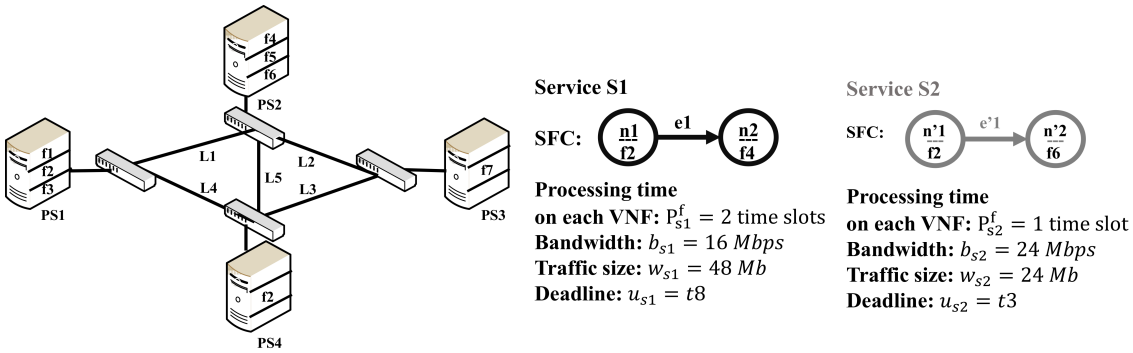
Soon, human history will witness a new era that will make science fiction a reality due to the joint efforts performed by both, industry and academia, to develop and implement the **fifth generation (5G) networks**, perceived to see the light in 2020 [1], [2], [3]. Real-time applications such as remote surgery, self-driving cars, transport services, remote control of robotic operations in the manufacturing industry [4] will become possible due to the 5G technology that is envisioned to guarantee **Ultra-Reliable** and **Ultra-Low Latency** communication services. Nonetheless, 5G systems should be elastic, cost-efficient, agile, programmable and able to support the ever-growing mobile data traffic [2], [5].

In order to achieve these goals, 5G should overcome several challenges to be able to support mission critical machine type communication (MTC) services (i.e., real-time control, process automation and manufacturing, etc.) [1]. Further, it should ensure security and performance of the transfered data. Such functionalities are provided by hardware appliances deployed in the network and known as **Middleboxes (MB)** (i.e, firewall, WAN optimizer, proxies, etc.) [6]. The authors of [7] state that the number of MBs in the network are comparable to the number of routers. In fact, network operators over-provision MBs in order to accommodate for the increasing mobile traffic demands and peak hours. These hardware appliances are expensive, vendor-specific, deployed in fixed locations and require trained personnel for their deployment and maintenance [8], [9]. They lead to high Capital Expenditures (CAPEX) and Operation Expenditures (OPEX) [8], [9], [10].

Hence, **Network Function Virtualization (NFV)** has been proposed as a solution for the aforementioned MBs limitations and is a Key enabler for 5G networks [5], [11]. NFV offers **Virtual Network Functions (VNFs)**, which are software appliances able to replace expensive hardware MBs. By decoupling network functions (NFs) from hardware resources, NFV allows the deployment of VNFs on demand on commodity hardware (i.e, high volume servers, switches and storage) anywhere, anytime in the network [9], [12], [13], [14]. VNFs can thus be relocated to different network locations without requiring the purchase and the installation of new hardware [9]. Hence, NFV guarantees flexibility, manageability and cost-efficiency [9], [14], [15].

To cope with 5G challenges and maintain an ultra-low latency end-to-end communication, NFV should ensure a fast processing of the traffic. For instance, autonomous driving cars within the same area are perceived to have sensors that will exchange information in real-time to help avoid fatal accidents and traffic congestions. Thus, such real-time information requires an ordered processing within a deadline by one or a chain of NFs. This chaining of NFs is known as **Service Function Chaining (SFC)** [9], [13]; whereas a **Network Service (NS)** is identified as "*a composition of NFs defined by its functional and behavioral specification*" [16]. Usually, delay sensitive NSs will be rejected from the network if their deadlines were not met [12]. Thus, to reach the ultra-low latency end-to-end communication objective, NFV should overcome three main challenges: 1) *The NFs mapping sub-problem* which aims at deciding which VNFs deployed in the network has to process the traffic of a given NS [8], [12], [17]; 2) *The traffic routing sub-problem* which entails steering the traffic through the SFC to be processed by the VNFs in the required order; 3) *The deadline-aware service scheduling sub-problem* that asks to determine the execution time slots (on each VNF) of the various services sharing the same VNF subject to their chaining and deadline requirements [12], [17], [18]. That is, the allocation of the VNF computing resources (i.e, CPU schedule allocation) to the flows of the given NS. While much work in the literature tackled the NFs mapping and traffic routing sub-problems [8], [17], [18], [19], only few discussed the deadline-aware service scheduling sub-problem [12], [20]. For instance, Mijumbi et al. [12] presented a list of benchmark algorithms while the authors

This work is supported by NSERC Discovery Grant.



(a) Physical network hosting different types of VNFs (p_f= 24Mbps; c_l = 30 Mbps)

(b) Service S1 network requirements

(c) Service S2 network requirements

Fig. 1: Physical network topology and NSs used to illustrate the motivational examples.

of [20] considered the VNFs transmission and processing delays when solving the scheduling sub-problem. However, only few considered services with strict deadlines while none studied the interplay that exists between NFs mapping, traffic routing and deadline-aware service scheduling sub-problems. Our work builds on top of prior works (e.g. [12], [20]) and introduces through several motivational examples new insights on the impact of the NFs mapping and traffic routing on the NS schedule. Thus, we present the **Deadline-Aware Service Function Chaining Scheduling (DA-SFCS) problem** that tackles the three aforementioned sub-problems jointly. To the best of our knowledge we are the first to solve the DA-SFCS problem while considering deadline constraints and accounting for ultra-sensitive low latency services. Hence, we define and formulate the **DA-SFCS problem as a mixed integer linear program (DA-SFCS-MILP)**. Given its complexity, we propose a sequential approach (**Sequential-DA-SFCS (SDA-SFCS)**) and a tabu search-based heuristic to solve it. Our numerical evaluation shows the efficiency of our SDA-SFCS and tabu search heuristics in comparison to DA-SFCS-MILP.

The remainder of the paper is organized as follows: Section II exploits the interplay that exists between those aforementioned sub-problems and explains their impact on the number of admitted NSs. Section III provides a definition and a formulation of the DA-SFCS problem. Section IV explains our sequential approach for solving the DA-SFCS problem while Section V presents a tabu search-based DA-SFCS algorithm. Our numerical evaluation is exposed in Section VI. We conclude in Section VII.

II. Motivation and Challenges

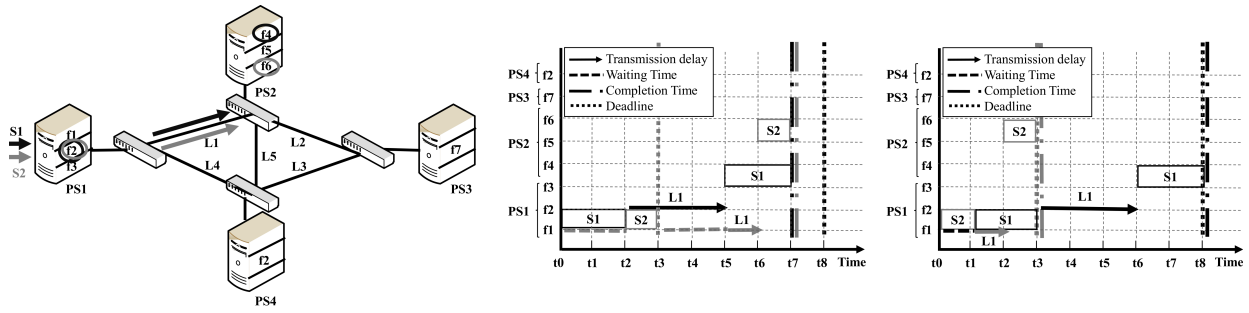
A. Problem Description

VNFs are software modules identified by their processing and buffer capacities and require some physical resources (i.e. CPU, RAM, storage, etc) when placed in the network [10]. They can be deployed on virtual machines (VMs) hosted on commodity hardware [9], [10], [21]. The problem of placing VNFs in the network [22], [23], [8], [24], [25] has been widely studied in the literature and is out of the scope of this paper. Thus, in the remaining of this paper, we consider a physical network (e.g. telecom operator's network) where VNFs are placed and running on top of VMs deployed on physical servers. For the sake of simplicity, we consider that each VM is dedicated to one VNF [12], and we assume that VNFs have unlimited buffer capacity. Given a certain delay-sensitve NS requiring its traffic to be processed by a NF or a chain of NFs (SFC), before reaching a certain deadline, and transmitted with a guaranteed bandwidth[1], our problem consists of mapping the NFs along the chain onto VNFs deployed in the network, steering its traffic between them while respecting their ordering and the demanded bandwidth, and finally scheduling the allocation of VNFs to its traffic with respect to the deadline. Hence, we consider that VNFs can be shared between several NSs, however, a VNF can process the traffic of one NS at a time. In addition accounting for the NFs ordering in the SFC entails that the traffic of a NS should finish its processing on a NF before being transmitted to the next one in the chain. In the remainder of this section, we consider a physical network of 4 physical servers connected by physical links ($l \in L$) of uniform capacity $c_l = 30Mbps$. These servers are hosting VNFs of different types (i.e., $f1, ..., f7$) as shown in Fig.1(a). Here, $f1, ..., f7$ can each represent a firewall, a proxy server, a cache, etc. The VNFs hosted in the network may have different processing capacities. The processing time P^f_s required to process the traffic of a NS s on a VNF f is calculated using Eq.(1) where w_s is the traffic size of the NS s (in bits) and p_f depicts the processing capacity of VNF f (in bps).

$$P^f_s = \frac{w_s}{p_f} \tag{1}$$

For the sake of simplicity, we assume in our following motivational examples that all VNFs have the same processing capacity $p_f = 24Mbps$. We explore the mapping, routing and scheduling of two NSs $S1$ and $S2$ with different computing and network requirements. Here, we consider that each NS requests a set of NFs of defined types in the form of SFC. We represent the required SFC as a forwarding graph [16] of virtual nodes depicting the NFs connected by virtual links. For instance, $S1$ (Fig.1(b)) requests a chain of two NFs; $n1$ of type $f2$ and $n2$ of type $f4$ which are connected by a virtual link denoted by $e1$. The processing time of the traffic of $S1$ on each VNF is calculated based on Eq.(1). Similarly, the network requirements of NS $S2$ are depicted in Fig.(1(c)). Further, we

[1]A guaranteed bandwidth for the communication between VMs (hosting VNFs) implies a predictable performance for the running services [26], [27], [28], [29], [30].



(a) S1 and S2 sharing the same VNF and route

(b) Deadline violation of S2 when S1 starts processing on f2 on PS1 first

(c) S2 meets its deadline when it starts processing on f2 on PS1 before S1

Fig. 2: Interaction between NSs schedules.

assume that time is divided into slots (i.e., $t1, t2, t3$, etc.), each representing a duration of one second. We consider that each NS s has a deadline time depicted by u_s and identified by the time slot number (i.e., deadline of $S1$ is $u_{s1} = t8$). In addition, we calculate the transmission delay D_s of the traffic of a NS s on each physical link on which it is routed by applying Eq.(2) where b_s depicts the bandwidth to be guaranteed for s.

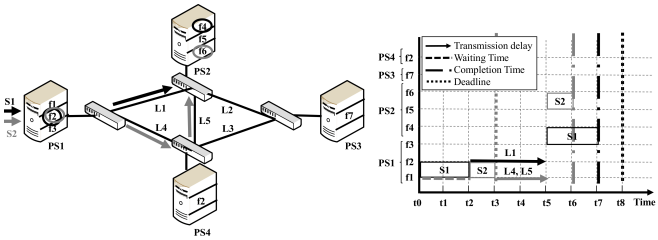
$$D_s = \frac{w_s}{b_s} \tag{2}$$

B. Mapping, routing and scheduling interplay

In order to highlight the interplay that exists between the NFs mapping, traffic routing and deadline-aware scheduling sub-problems, we consider the example in Fig.2(a) where the NFs $n1$ and $n'1$ of NSs $S1$ and $S2$ respectively, are mapped to the same VNF $f2$ hosted on server $PS1$ at the same time slot $t0$. Given that a VNF can process the traffic of only one NS at a time, $f2$ hosted on $PS1$ can start processing the traffic of either one of both NSs $S1$ or $S2$. We assume that it starts by processing the traffic of $S1$ while the processing of the traffic of $S2$ on $f2$ is delayed for $P_{s1}^{f2} = 2\ time\ slots$ (Eq.(1)) (Fig.2(b)). Once $f2$ finishes the processing of the traffic of $S1$, this latter can be transmitted to $f4$ on $PS2$ where the NF $n2$ is mapped. Given that at $t2$, the link $L1$ is not being used and has an available bandwidth $(c_l = 30Mbps) \geq (b_{s1} = 16Mbps)$, the virtual link $e1$ of $S1$ can be mapped/routed through this shortest path. $D_{s1} = 3\ time\ slots$ ((Eq.(2)) are needed for the transmission of all the traffic of $S1$ which arrives to $f4$ hosted on $PS2$ at $t5$ when its processing starts on this latter and lasts for $P_{s1}^{f4} = 2\ time\ slots$. Hence, $S1$ meets its deadline $(u_{s1} = t8)$ by completing its processing at $t7$ while achieving its shortest schedule where no waiting time was incurred. Once $f2$ on $PS1$ finishes the processing of the traffic of $S1$, it starts at $t2$ the processing of the traffic of $S2$ for one time slot ($t2$) ($P_{s2}^{f2} = 1\ time\ slot$) (Fig.2(b)). We choose the shortest path (link $L1$) to transmit the traffic of $S2$ from $PS1$ to $PS2$ so it can get processed on $f6$ where n'_2 is mapped. Even though the traffic of $S2$ is ready to be transmitted at $t3$, it has to wait for 2 time slots ($t3$ and $t4$) to be able to get routed through $L1$ (Fig.2(b)). The reason behind this is that the bandwidth $b_{s2} = 24Mbps$ required by $S2$ can not be guaranteed on $L1$ at time slots $t3$ and $t4$. In fact, the available bandwidth on $L1$ at $t3$ and $t4$ is $bw_{L1} = c_l - b_{s1} = 30 - 16 = 14Mbps$, since the traffic of $S1$ is being routed through $L1$ at $t3$ and $t4$. Hence, $bw_{L1} \leq b_{s2}$, which would delay the transmission of the traffic of $S2$ for 2 time slots (queuing delay) until the bandwidth used by $S1$ on $L1$ is released. Thus, the transmission of the traffic of $S2$ starts at $t5$ and last for one time slot before it gets processed by $f6$ hosted on $PS2$. Hence, the traffic of $S2$ completes its processing at $t7$, thus violating its deadline $u_{s2} = t3$ by 4 time slots which leads to its rejection from the network (Fig.2(b)). Next, we explore several approaches that may allow $S2$ to meet its deadline.

C. Traffic Routing impact the NS schedule

In the previous example, the traffic of $S2$ suffered waiting delays; 1) to be processed on $f2$ on $PS1$ and 2) to be transmitted through $L1$ which prevented it meeting its deadline (Fig.2(b)). Thus, one way to reduce $S2$'s schedule length is to route its traffic through another path that can guarantee its required bandwidth $b_{s2} = 24Mbps$ at $t3$, just when it finishes its processing on $f2$ hosted on $PS1$. Such route can be $L4, L5$ (Fig.3(a)). By using this latter, the traffic of $S2$ is transmitted at $t3$ without any waiting delays (Fig.3(b)). However, the transmission delay increased by one time slot. In fact, the traffic of $S2$ need to traverse 2 links ($L4$ and $L5$), incurring a transmission delay of $2 \times D_{s2} = 2 \times 1 = 2\ time\ slots$. Hence, changing the routing of $S2$ decreased its schedule length by one time slot (in comparison to the shortest path routing used in (Fig.2(a)) to finish it processing at $t6$ (Fig.3(b)) which did not allow it to meet its deadline $u_{s2} = t3$. We conclude that



(a) Change of S2 routing

(b) Reduction of 1 time slot in S2 schedule length

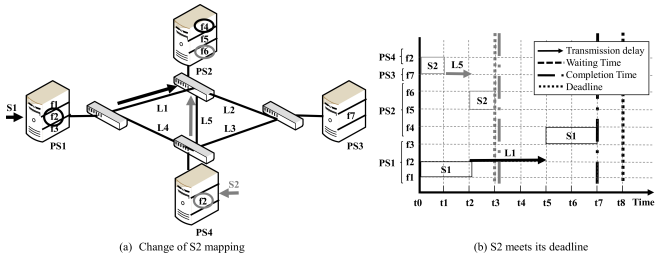
Fig. 3: Impact of traffic routing on NSs schedules.

the path chosen to route the traffic of a NS has a reciprocal impact on its schedule and the network resources utilization.

D. VNFs mapping impacts the NS schedule

Another way to reduce the schedule length of $S2$ is to minimize the waiting time incurred to start processing its traffic on $f2$ hosted on $PS1$ (Fig.2). To do so, we change the mapping of $n'1$ of NS $S2$ from $f2$ hosted on $PS1$ (Fig.2(a)) to $f2$ hosted on $PS4$ (Fig.4(a)). Thus, the route

of the traffic needs to be modified so it can be transmitted from $PS4$ to $PS2$. We choose to route the traffic of $S2$ in this case through $L5$ that represents the shortest path between $PS4$ and $PS2$ and able to guarantee the bandwidth $b_{s2} = 24Mbps$ required by $S2$. Given this mapping and routing, the traffic of $S2$ can start processing on $f2$ hosted on $PS4$ at $t0$ for $P_{s2}^{f2} = 1\ time\ slot$, starts transmission at $t1$ for $D_{s2} = 1\ time\ slot$, then gets processed by $f6$ hosted on $PS2$ at $t2$ (Fig.4(b)). Thus, with this mapping and routing, $S2$ achieves its shortest schedule and meets its deadline $u_{s2} = t3$. Hence, a different mapping of the NFs of $S2$ allowed better



(a) Change of S2 mapping (b) S2 meets its deadline

Fig. 4: Impact of NFs mapping on NSs schedules.

utilization of idle network resources ($f2$ hosted on $PS4$ and $L5$) and enabled its admission in the network. This highlights the interplay that exists between NFs mapping and scheduling.

E. NSs schedule interplay

In the previous examples (Section II-C and II-D), we changed the traffic routing and the NFs mapping of $S2$ to reduce its schedule length and try to meet its deadline. However, one can think of keeping the same routing and mapping as in Fig.2(a) but change the schedule of $S2$. For instance, in Fig.2(a), $S2$ and $S1$ were sharing the same VNF $f2$ (hosted on $PS1$) which delayed the processing of the traffic of $S2$ that had to wait for $f2$ until it finishes the processing of the traffic of $S1$. In fact, the traffic of $S1$ was scheduled to be processed before the traffic of $S2$ on $f2$ hosted on $PS1$. If we change the schedule of $S2$ on $f2$, we can schedule the traffic of $S2$ to be processed before the traffic of $S1$ (Fig.2(c)). In this case, $S2$ is processed and transmitted without any waiting delays. It starts processing at $t0$ on $f2$ and pursue its transmission on $L1$ at $t1$. Here, no waiting delays for the transmission on $L1$ are incurred since this latter is not used by any other service and has enough available bandwidth ($c_l = 30Mbps$) to guarantee the bandwidth required by $S2$ ($b_{s2} = 16Mbps$). After the transmission on $L1$, the traffic of $S2$ is processed by $f6$ hosted on $PS2$ at $t2$ and finishes its processing at $t3$, thus, meeting the deadline. However, this approach delays the schedule of $S1$ by one time slot as it has to wait for $f2$ to finish the processing of the traffic of $S2$ (Fig.2(c)). Hence, in this case, the traffic of $S1$ finishes its processing at $t8$ (Fig.2(c)) instead of $t7$ (Fig.2(b)) but still meets its deadline $u_{s1} = t8$. In conclusion, there exists an interplay between the schedule of NSs sharing the same VNF and/or the same route, whereas the modification of the schedule of one can impact the schedule of the other, and may lead to either meeting or violating the deadline of one or both of them, thus affecting the admission rate in the network. We seek to explore this interplay in the rest of the paper.

III. DA-SFCS - A Mixed Integer Linear Problem (DA-SFCS-MILP)

In the following, we define and present a mathematical formulation for the Delay-Aware Service Function Chaining Scheduling problem (DA-SFCS).

A. Problem Definition

Let $G(K, L)$ be a substrate network of K nodes (i.e, servers, switches, etc.) hosting VNFs and L a set of links connecting them. Given a set of services (NSs) where each NS requests its traffic to be processed by one or a sequence of NFs in a defined order and transmitted from one NF to another at a guaranteed bandwidth. Satisfying these services requires: 1) Mapping their NFs to VNFs of the same type hosted in G; 2) Routing their traffic through the VNFs in the defined chain (SFC) order while guaranteeing their required bandwidth; 3) Scheduling the processing time of the host functions such that the NS deadline is met. The most beneficial schedules permit to admit all of them in the network. Hence, we define the DA-SFCS problem as follows:

Definition 1: Given a physical network $G(K, L)$ hosting and running different types of VNFs, a set S of NSs, each demanding a SFC for its traffic, find the optimal mapping, routing and scheduling of the traffic of these NSs in $G(K, L)$ which maximizes the number of admitted ones while respecting their deadlines.

B. Problem formulation

Parameters

$G(K, L)$: Substrate network of K nodes and L physical links connecting them. $K = K_p \cup K_n$ where K_p denotes the set of physical servers in the network and K_n represents the set of physical equipments (i.e., routers/switches).

F : Set of VNF instances running on VMs hosted on the physical servers $k \in K_p$.

T : Set of types of all VNFs $f \in F$ (e.g., firewall, proxy, cache, Intrusion Detection System (IDS), etc.).

t_f : Type of a VNF instance $f \in F$ ($t_f \in T$)

c_{ij} : Capacity of a physical link $(ij) \in L$.

$x_f^k \in \{0, 1\}$: A parameter that indicates that the VNF instance $f \in F$ of type t_f is hosted on the physical server $k \in K_p$ (1) (or not, 0).

Δ : Set of time slots (time line).

S : Set of NSs.

$H_s(N_s, E_s)$: A forwarding graph representing the SFC of a NS $s \in S$ where N_s designates the sequence of NFs required by the NS $s \in S$ and E_s denotes the virtual links connecting them. We refer respectively by $o(e)$ and $d(e)$ to the origin and the destination of a virtual link $e \in E_s$.

b_s : Bandwidth demanded by NS $s \in S$ to be guaranteed for the communication between its NFs $n \in N_s$. b_s is the capacity of each of the virtual links $e \in E_s$.

w_s : Traffic demand of NS $s \in S$.

u_s : Deadline time of NS $s \in S$.

m_{ns} : Type of a NF $n \in N_s$ of service $s \in S$.

p_{ns} : Processing time of the traffic of NS $s \in S$ on the NF $n \in N_s$. We consider that VNFs $f \in F$ of the same type have a uniform processing capacity and we calculate p_{ns} based on

Eq.(1) given that the NF $n \in N_s$ will be mapped to a VNF of the same type.

d_s : Transmission time of the traffic of NS $s \in S$ on the virtual link $e \in E_s$. we calculate d_s based on Eq.(2).

Decision Variables

$a_s \in \{0,1\}$: specifies if service $s \in S$ is admitted (1) or not (0) to the network. A NS is admitted to the network if it can be scheduled within its deadline.

$y_{ns}^{f\delta} \in \{0,1\}$: specifies that the traffic of NS $s \in S$ started processing at time slot $\delta \in \Delta$ on the NF $n \in N_s$ scheduled/mapped to VNF $f \in F$.

$q_{ns}^{k} \in \{0,1\}$: specifies that the NF $n \in N_s$ of NS $s \in S$ is mapped to a VNF instance hosted on a physical server $k \in K_p$ (1) (or not, 0).

$h_{ns}^{k} \in \{0,1\}$: indicates that NFs $n, (n+1) \in N_s$ of NS $s \in S$ are mapped to VNFs hosted on the same physical server $k \in K_p$ (1) (or not, 0).

$\theta_s^{\delta e} \in \{0,1\}$: designates that a NF $o(e) \in N_s$ of NS $s \in S$ begins the transmission of the traffic to its successor NF $d(e) \in N_s$ at time slot $\delta \in \Delta$ on the virtual link $e \in E_s$ (1) (or not, 0).

$\hat{\theta}_s^{\delta e} \in \{0,1\}$: indicates that the virtual link $e \in E_s$ is used for traffic transmission between the NFs $o(e), d(e) \in N_s$ of NS $s \in S$ at time slot $\delta \in \Delta$ (1) (or not, 0).

$l_{ij}^{e} \in \{0,1\}$: denotes that the virtual link $e \in E_s$ of NS $s \in S$ is routed through the physical link $(ij) \in L$ (1) (or not, (0)).

Model

$$\text{Maximize} \quad \sum_{s \in S} a_s \tag{3}$$

$$\sum_{f \in F} \sum_{\delta \in \Delta} y_{ns}^{f\delta} = a_s \quad \substack{\forall n \in N_s \\ \forall s \in S} \tag{4}$$

$$\sum_{f \in F} \sum_{\delta \in \Delta} y_{ns}^{f\delta} t_f = a_s m_{ns} \quad \substack{\forall n \in N_s \\ \forall s \in S} \tag{5}$$

$$\theta_s^{\delta' e} \leq 1 - \sum_{f \in F} y_{o(e)s}^{f\delta} \quad \substack{\forall \delta, \delta' \in \Delta;\ \delta' < \delta + p_{o(e)s} \\ \forall e \in E_s \\ \forall s \in S} \tag{6}$$

$$\sum_{f \in F} y_{d(e)s}^{f\delta'} \leq 1 - \theta_s^{\delta e} \quad \substack{\forall \delta, \delta' \in \Delta;\ \delta' < \delta + d_s \\ \forall e \in E_s \\ \forall s \in S} \tag{7}$$

$$\sum_{f \in F} y_{(n+1)s}^{f\delta'} \leq 1 - \sum_{f \in F} y_{ns}^{f\delta} \quad \substack{\forall \delta, \delta' \in \Delta;\ \delta' < \delta + p_{ns} \\ \forall n, (n+1) \in N_s \\ \forall s \in S} \tag{8}$$

$$\sum_{s\prime \in S : s\prime \neq s} \sum_{n\prime \in N_{s\prime}} y_{n\prime s\prime}^{f\delta} = 1 - y_{ns}^{f\delta} \quad \substack{\forall n \in N_s \\ \forall s \in S \\ \forall f \in F \\ \forall \delta, \delta\prime \in \Delta; \delta \leq \delta\prime < \delta + p_{ns}} \tag{9}$$

$$\sum_{s \in S} \sum_{n \in N_s : m_{ns} = t_f} y_{ns}^{f\delta} \leq 1 \quad \substack{\forall \delta \in \Delta \\ \forall f \in F} \tag{10}$$

$$\sum_{f \in F} \sum_{\delta \in \Delta} y_{|N_s|s}^{f\delta} (\delta + p_{|N_s|s}) \leq u_s \quad \forall s \in S \tag{11}$$

$$q_{ns}^{k} = \sum_{f \in F} \sum_{\delta \in \Delta} y_{ns}^{f\delta} x_f^k \quad \substack{\forall n \in N_s \\ \forall s \in S \\ \forall k \in K_p} \tag{12}$$

$$h_{ns}^{k} = q_{ns}^{k} q_{(n+1)s}^{k} \quad \substack{\forall k \in K_p \\ \forall n, (n+1) \in N_s \\ \forall s \in S} \tag{13}$$

$$\sum_{\delta \in \Delta} \theta_s^{\delta e} = (1 - \sum_{k \in K_p} h_{o(e)s}^{k}) a_s \quad \substack{\forall e \in E_s \\ \forall s \in S} \tag{14}$$

$$\sum_{\delta \in \Delta} \hat{\theta}_s^{\delta e} = d_s \sum_{\delta \in \Delta} \theta_s^{\delta e} \quad \substack{\forall e \in E_s \\ \forall s \in S} \tag{15}$$

$$\sum_{\delta' \in [\delta,\ \delta + d_s - 1]} \hat{\theta}_s^{\delta' e} \geq d_s \theta_s^{\delta e} \quad \substack{\forall e \in E_s \\ \forall s \in S \\ \forall \delta \in \Delta} \tag{16}$$

$$\sum_{s \in S} \sum_{e \in E_s} l_{ij}^{e} \hat{\theta}_s^{\delta e} b_s \leq c_{ij} \quad \substack{\forall \delta \in \Delta \\ \forall (ij) \in L} \tag{17}$$

$$\sum_{j:(i,j) \in L} l_{ij}^{e} - \sum_{j:(j,i) \in L} l_{ji}^{e} = q_{o(e)s}^{i} - q_{d(e)s}^{i} \quad \substack{\forall e \in E_s \\ \forall s \in S \\ \forall i \in K_p} \tag{18}$$

$$l_{ij}^{e} \leq 1 - \sum_{k \in K_p} h_{o(e)s}^{k} \quad \substack{\forall e \in E_s \\ \forall s \in S \\ \forall (ij) \in L} \tag{19}$$

$$l_{ij}^{e} \leq a_s \quad \substack{\forall e \in E_s \\ \forall s \in S \\ \forall (ij) \in L} \tag{20}$$

The objective function (Eq.3) aims at maximizing the number of admitted NSs. A NS is admitted to the network if it can be served within its deadline time u_s. That is, the NFs of the NS can be mapped, it traffic routed and its schedule meets its deadline. Constraint (4) ensures that a NF n of service s is mapped exactly to one VNF instance f if the NS s is admitted to the network. Constraint (5) guarantees that a NF n of service s is mapped to a VNF instance f of the same type. Constraint (6) prevents the transmission of traffic of a NS s between two consecutive NFs $o(e)$ and d(e) if its processing on $o(e)$ has not been completed. Constraint (7) ensures that traffic of a service s can not be processed by a NF $d(e)$ if it was not transmitted to it by its precedent NF $o(e)$. Constraint (8) prevents a NF $(n+1)$ to start processing the traffic of a NS s before its predecessor NF n finishes its execution. Constraint (9) specifies that a VNF f is processing the traffic of NS s during all the processing period of this latter and prevents it from processing the traffic of another NS $s\prime$ during that period. Constraint (10) ensures that a VNF f can not process the traffic of more than one NS at a certain time slot $\delta \in \Delta$. Constraint (11) ensures that a NS is served and finishes its processing before its deadline u_s. Constraint (12) specifies the physical server k on which a NF n of service s is mapped. Constraint (13) specifies if two consecutive NFs n and $(n+1)$ of the same NS s are mapped to VNFs hosted on the same physical server $k \in K_p$. Constraint (14) prevents the start of the transmission of traffic of a NS s between two consecutive NFs $o(e)$ and $d(e)$ if they are mapped to VNFs hosted on the same physical server or if the NS s is not admitted. Constraint (15) specifies that the virtual link e is used to transmit traffic of a NS s during all the required transmission time (d_s) between NFs o(e) and d(e) only if transmission is possible (i.e, NFs o(e) and d(e) are mapped to VNFs hosted on different physical servers). Constraint (16) ensures the virtual link e is used for transmission of traffic of a NS s during all the transmission delay (d_s) starting at time slot δ (when the transmission begins). Constraint (17) ensures that the physical links capacities are not violated. Constraint (18) represents the flow conservation constraints which determines

the physical links ($(ij) \in L$) of the network through which the virtual links ($e \in E_s$) of a NS s are routed. Constraint (19) prevents routing of a virtual link e of a NS s through a physical link $(ij) \in L$ if $o(e)$ and $d(e)$ are mapped to VNFs hosted on the same physical server. Constraint (20) prevents routing of a virtual link e of a NS s through a physical link $(ij) \in L$ if s is not admitted to the network. Constraints (13), (14), (17) are non linear and can be easily linearized; however, we omit the linearization details due to space limitations.

The Deadline-Aware Service Function Chaining Scheduling model is a Mixed Integer Linear problem ((DA-SFCS-MILP) which is complex to solve. It is NP-Hard given that it is a combination of three NP-Hard sub-problems which are the NFs mapping sub-problem [8], [31], the traffic routing sub-problem [32], [33] and the deadline-aware service scheduling sub-problem [34], [35]. Thus, in the next Sections we present several heuristics to solve it.

IV. The Sequential DA-SFCS problem (SDA-SFCS)

Given the complexity of the DA-SFCS problem, we present in this section two heuristics to solve it by tackling each of its complex sub-problems in a sequential manner:

A. NFs Mapping sub-problem

1- *Load Aware Mapping*: Given a set of NSs, the load aware mapping consists of mapping each of the NFs of every NS into the VNF of the same type that has the least number of mapped NFs (least loaded). Such approach will balance the load between the VNFs of the same type hosted in the network.
2- *Random Mapping*: Given a set of NSs, the random method maps at random the NFs of each NS to VNFs chosen from the pool of VNFs of the same type hosted in the network.

B. Traffic Routing sub-problem

Once the NFs of a NS are mapped to VNFs, we solve the traffic routing sub-problem by finding the shortest path route with enough capacity to guarantee the service its demanded bandwidth (b_s). We use the Dijkstra algorithm to determine the shortest path between each two consecutive NFs in the SFC of each NS.

C. Deadline-Aware Scheduling sub-problem - A mixed Integer Linear Problem (DAS-MILP)

Now, given a set of NSs having their NFs mapped and traffic route defined with respect to their SFC and required bandwidth, we solve the deadline-aware scheduling (DAS) sub-problem through the **DAS-MILP**. The DAS-MILP attempts to maximize the number of NSs that can be scheduled within their deadlines, thus maximizing the admission rate.

Definition 2: Given a physical network $G(K, L)$ hosting and running different types of VNFs, a set S of NSs, each demanding a SFC having its NFs mapped in $G(K, L)$ and its traffic route determined, find the optimal scheduling of the traffic of these NSs which maximizes the number of admitted ones while respecting their deadlines.

Parameters

$r^f_{ns} \in \{0, 1\}$: indicates that a NF $n \in N_s$ of NS $s \in S$ is mapped to a VNF $f \in F$. In other words, the traffic of service $s \in S$ will be processed by the VNF $f \in F$.

$h^k_{ns} \in \{0, 1\}$: indicates that NFs $n, (n + 1) \in N_s$ of NS $s \in S$ are mapped to VNFs hosted on the same physical server $k \in K_p$ (1) (or not, 0).

$l^e_{ij} \in \{0, 1\}$: denotes that the virtual link $e \in E_s$ of NS $s \in S$ is routed through the physical link $(ij) \in L$ (1) (or not, (0)).

The remaining parameters are listed in Section III-B.

Decision Variables

$a_s \in \{0, 1\}$: specifies if NS $s \in S$ is admitted (1) or not (0) to the network. A NS is admitted to the network if it can be scheduled within its deadline.

$y^{f\delta}_s \in \{0, 1\}$: specifies that the traffic of the NS $s \in S$ started processing at time slot $\delta \in \Delta$ on VNF $f \in F$.

$\theta^{\delta e}_s \in \{0, 1\}$: designates that a NF $o(e) \in N_s$ of NS $s \in S$ begins the transmission of the traffic to its successor NF $d(e) \in N_s$ at time slot $\delta \in \Delta$ on the virtual link $e \in E_s$ (1) (or not, 0).

$\hat{\theta}^{\delta e}_s \in \{0, 1\}$: indicates that the virtual link $e \in E_s$ is used for traffic transmission between the NFs $o(e), d(e) \in N_s$ of NS $s \in S$ at time slot $\delta \in \Delta$ (1) (or not, 0).

Model

$$\text{Maximize} \quad \sum_{s \in S} a_s \tag{21}$$

$$\theta^{\delta' e}_s \leq 1 - \sum_{f \in F} y^{f\delta}_s r^f_{o(e)s} \quad \begin{smallmatrix}\forall \delta, \delta' \in \Delta;\ \delta' < \delta + p_{o(e)s} \\ \forall e \in E_s \\ \forall s \in S\end{smallmatrix} \tag{22}$$

$$\sum_{f \in F} y^{f\delta'}_s r^f_{d(e)s} \leq 1 - \theta^{\delta e}_s \quad \begin{smallmatrix}\forall \delta, \delta' \in \Delta;\ \delta' < \delta + d_s \\ \forall e \in E_s \\ \forall s \in S\end{smallmatrix} \tag{23}$$

$$\sum_{f \in F} y^{f\delta'}_s r^f_{(n+1)s} \leq 1 - \sum_{f \in F} y^{f\delta}_s r^f_{ns} \quad \begin{smallmatrix}\forall \delta, \delta' \in \Delta;\ \delta' < \delta + p_{ns} \\ \forall n, (n+1) \in N_s \\ \forall s \in S\end{smallmatrix} \tag{24}$$

$$\sum_{f \in F} \sum_{\delta \in \Delta} y^{f\delta}_s r^f_{ns} = a_s \quad \begin{smallmatrix}\forall n \in N_s \\ \forall s \in S\end{smallmatrix} \tag{25}$$

$$\sum_{s\prime \in S; s\prime \neq s} y^{f\delta'}_{s\prime} \leq 1 - y^{f\delta}_s r^f_{ns} \quad \begin{smallmatrix}\forall \delta, \delta' \in \Delta;\ \delta \leq \delta' < \delta + p_{ns} \\ \forall s \in S \\ \forall n \in N_s \\ \forall f \in F\end{smallmatrix} \tag{26}$$

$$\sum_{s \in S} y^{f\delta}_s \leq 1 \quad \begin{smallmatrix}\forall \delta \in \Delta \\ \forall f \in F\end{smallmatrix} \tag{27}$$

$$\sum_{f \in F} \sum_{\delta \in \Delta} y^{f\delta}_s r^f_{|N_s|s} (\delta + p_{|N_s|s}) \leq u_s \quad \forall s \in S \tag{28}$$

$$\sum_{\delta \in \Delta} \theta^{\delta e}_s = (1 - \sum_{k \in K_p} h^k_{o(e)s}) a_s \quad \begin{smallmatrix}\forall e \in E_s \\ \forall s \in S\end{smallmatrix} \tag{29}$$

$$\sum_{\delta \in \Delta} \hat{\theta}^{\delta e}_s = d_s \sum_{\delta \in \Delta} \theta^{\delta e}_s \quad \begin{smallmatrix}\forall e \in E_s \\ \forall s \in S\end{smallmatrix} \tag{30}$$

$$\sum_{\delta' \in [\delta,\ \delta + d_s - 1]} \hat{\theta}^{\delta' e}_s \geq d_s \theta^{\delta e}_s \quad \begin{smallmatrix}\forall e \in E_s \\ \forall s \in S \\ \forall \delta \in \Delta\end{smallmatrix} \tag{31}$$

$$\sum_{s \in S} \sum_{e \in E_s} l^e_{ij} \hat{\theta}^{\delta e}_s b_s \leq c_{ij} \quad \begin{smallmatrix}\forall \delta \in \Delta \\ \forall (ij) \in L\end{smallmatrix} \tag{32}$$

The DAS-MILP problem aims at scheduling the NSs while maximizing the number of admitted ones Eq.(21). Constraint (22) ensures that the traffic of a NS s can start its transmission from o(e) to d(e), if and only if, its processing on $o(e)$ is completed. Constraint (23) allows the processing of the

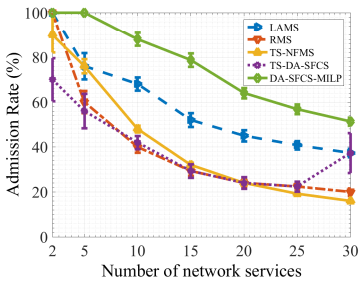


Fig. 5: Admission rate over the number of NSs

Number of NSs	DA-SFCS-MILP	LAMS	RMS	TS-NFMS	TS-DA-SFCS
2	293	108	102.6	1.4	5
5	1 031.8	330.8	224.4	0.8	12.6
10	4 167	628.8	643.4	0.8	9.8
15	18 246.8	1 473	1 332.8	1.4	9.6
20	1 821 094.6	2 083.8	2 818	1.4	11.6
25	2 092 851.4	3 253.6	2 825	1.2	14.4
30	718 6610.4	3 536.2	5 064.2	7.8	12.8

TABLE I: Execution Time (ms) over the number of NSs

traffic of service s on $d(e)$, if it was fully transmitted to it by its predecessor NF $o(e)$. Constraint (24) prevents a NF $(n+1)$ to start processing the traffic of a NS s before its predecessor NF n finishes its execution. Constraint (25) ensures that the traffic of a NS s starts processing at each VNF f to which the NF n is mapped if s is admitted. Constraint (26) specifies that a VNF f is processing the traffic of NS s during all the processing period of this latter and prevents it from processing the traffic of another NS $s\prime$ during the same period. Constraint (27) ensures that a VNF f can not process the traffic of more than one NS at a certain time slot $\delta \in \Delta$. Constraint (28) specifies that a NS is served and finishes its processing before its deadline u_s. Constraint (29) prevents the transmission of the traffic of a NS s if it is not admitted or if the NFs $o(e)$ and $d(e)$ are mapped to VNFs hosted on the same physical server. Constraint (30) specifies that the virtual link e is used to transmit traffic of a NS s during all the required transmission time (d_s) between NFs o(e) and d(e). Constraint (31) ensures that the transmission of the traffic of NS s is done consequently during all the transmission delay (d_s) on the virtual link e starting at time slot δ (when the transmission begins). Constraint (32) ensures that the physical links capacities are not violated.

We solve the DA-SFCS problem sequentially by developing the **Load Aware Mapping-Scheduling (LAMS)** heuristic and the **Random Mapping Scheduling (RMS)** heuristic where the first performs a load aware mapping and the second uses a random mapping (Section IV-A). LAMS and RMS heuristics use a shortest path routing (Section IV-B) and the DAS-MILP scheduling (Section IV-C).

V. Tabu Search-Based DA-SFCS (TS-DA-SFCS)

Given that LAMS and RMS use the DA-MILP which is a scheduling problem, known to be NP-Hard [34], [35], we propose a tabu search-based algorithm **(TS-DA-SFCS)** to efficiently solve the DA-SFCS problem. Our tabu-based approach consists of building an initial solution for the provided NSs first, through traversing them in a pre-defined order and attempting to map their NFs using the random mapping (Section IV-A), route their traffic by running a shortest path algorithm (Section IV-B) and scheduling their traffic on a First Come First Served (FCFS) based scheduling. Should the initial solution yields impossible for any of the NSs, we conclude that this latter can not be admitted to the network, and hence, will not be considered during the tabu search.

Let $initialAdmssion$ be an array holding the subset of NSs for which we obtained an initial solution. We determine a tabu move as the swap of the order of two NSs from 2 randomly chosen indexes in the $initialAdmission$ array. Such swap will change the scheduling order of the NSs sharing the same VNFs of the swapped ones. Such a move is therefore, accompanied by re-mapping, re-routing and re-scheduling of all the NSs in the updated $initialAdmission$ array using the same algorithms applied to obtain the initial solution. The move with the highest number of admitted requests is deemed as the best candidate move and the current solution is updated to hold the DA-SFCS solution for each of the NSs.

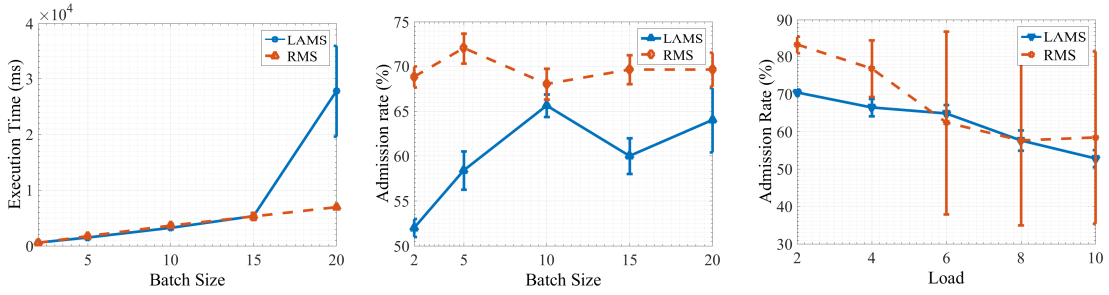
We maintain a tabu list holding the index of the NSs that were swapped. This prevents cycling back to an old solution obtained by swapping the same NSs placed at the same index specified in the tabu list. The stopping condition is set to a predetermined number of iterations where at each we evaluate a defined number of swaps of two NSs of the $initialAdmission$.

VI. Performance Evaluation

We carry out extensive study to evaluate the performance of our DA-SFCS-MILP (Section III) against our LAMS, RMS heuristics (Section IV) in addition to the TS-DA-SFCS (Section V). Further, we compare our proposed methods against the state of the art tabu search-based for network function mapping and scheduling (TS-NFMS) developed in [12]. To have a valid comparison, we add a shortest path routing to the TS-NFMS. Hence, we perform our evaluation using an offline approach where NSs are known a priori and we study the impact of scheduling different number of NSs jointly in providing better resources utilization through exploring an online batch scheduling approach. During our numerical study, we use sets of services randomly generated of varying traffic ([500-1500] Mbits) and bandwidth requirements ([300-500] Mbps) and demanding varying NFs ([3-5] NFs). Their deadlines are set as the 4/3 of the sum of their processing and transmission delays without considering any waiting delays. In case of the online approach we add their arrival times to those latter. All our numerical evaluations, averaged over 5 sets, are conducted using Cplex version 12.4 to solve the optimization problems on an Intel core i7-4790 CPU at 3.60 GHZ with 16 GB RAM.

A. DA-SFCS-MILP Evaluation

We consider a small test mesh network consisting of 4 physical servers hosting 7 VNFs placed at random. Each server is connected to a switch and the switches are interconnected by 5 links of 500Mbps each. We use this network to run the DA-SFCS-MILP model and obtain reference results to compare



(a) Execution time over batch size (b) Admission rate over batch size (c) Admission rate over load

Fig. 6: Online batch scheduling evaluation.

with those acquired using the LAMS, the RMS, TS-DA-SFCS and the TS-NFMS algorithms. We consider that all services are known a priori (offline mode) and we run our tests over sets of services of different sizes.

1-Optimality Gap: Fig.5 shows the admission rate of the DA-SFCS-MILP, the LAMS, the RMS, the TS-DA-SFCS and the TS-NFMS algorithms as we increase the number of NSs. It is clear from Fig.(5), that the admission rate decreases with the increase of the number of NSs as the amount of computing and network (i.e., VNFs, bandwidth) resources decreases, thus reducing the number of NSs that can meet their deadlines. Further, for a small number of NSs (2 NSs), the LAMS and the RMS are able to provide the optimal solution obtained by the DA-SFCS-MILP. However, with the increase of this number, the optimality gap between the DA-SFCS-MILP and the LAMS, the RMS, the TS-DA-SFCS and the TS-NFMS methods increases. In addition, the LAMS outperforms the RMS and the tabu search algorithms in terms of admission rate as it attempts to balance the load between the VNFs of the same type. Moreover, even though the TS-DA-SFCS, the TS-NFMS and the RMS performs a random mapping, RMS outperforms the tabu search algorithms in terms of admission rates as it can provide an optimal solution for the scheduling problem while the others uses a FCFS approach. It is also clear that the TS-NFMS outperforms the TS-DA-SFCS as it considers changing the mapping of NFs with highest flow time (as defined in [12]).

2-Execution Time: In order to study the scalability of the DA-SFCS-MILP, we compare its execution time against the four mentioned algorithms. Table I shows that when the number of services is small (2, 5, 10 NSs), all the compared methods were able to find a solution in less than 1 second. However, as the number of services increases, the DA-SFCS-MILP becomes much harder to solve and its runtime increases exponentially. For instance, when the number of NSs is 30, the execution time of the DA-SFCS-MILP passes the 2 hours whereas the RMS, LAMS, the TS-DA-SFCS and the TS-NFMS returned a solution in less than 5 seconds. Further, it is important to note that even though the execution times of the LAMS and the RMS increase when the number of NSs becomes larger due to the complexity of the DA-MILP, such increase is at slower pace than the DA-SFCS-MILP. Finally, the TS-DA-SFCS and the TS-NFMS are the most scalable as they do not invoke any complex formulation to solve.

B. Online Batch Scheduling Evaluation

We consider a mesh network of 8 physical servers hosting 15 VNFs placed at random. Each server is connected to a switch and the switches are interconnected by 10 links of 3Gbps each. We consider 25 services randomly generated as described earlier, following a Poisson traffic arrival. We alter the load by varying the arrival rate (λ) while fixing the average service time (μ) of the NSs ($load = \frac{\lambda}{\mu}$). We consider batches of different sizes and we evaluate the variation of the admission rate and the batch execution time.

1-Admission rate: Fig.6(b) shows that as the number of NSs per batch increases, the admission rate increases. This is due to the fact that the DAS-MILP used in LAMS and RMS heuristics can provide better resources utilization when supplied with more information about the upcoming NSs. Unlike the offline results (Fig.5), the LAMS provids worst admission rate than the RMS in the online scheduling given that it considers balancing the load of a single batch between VNFs without accounting for the actual load performed from the previous batches on these VNFs. To further explore the impact of the load on the admission rate, we consider batches of 5 NSs and we show in Fig.6(c) that as we increase the load, the admission rate decreases given that less network and computing resources are available, preventing NSs to meet their deadline.

2-Execution Time: Fig.6(a) depicts that the execution times of the LAMS and RMS algorithms increase with the size of the batch as the DAS-MILP becomes more complex to solve.

VII. Conclusion

In this paper, we have formally defined and formulated the DA-SFCS problem that studies and solves the three inter-related sub-problems of NFs mapping, traffic routing and deadline-aware scheduling of NSs with critical ultra-low latency requirements. To the best of our knowledge, we are the first to expose the importance of the DA-SFCS problem in NFV in serving the requirement of 5G technology. Given the complexity of the DA-SFCS-MILP, we have proposed a tabu search-based algorithm in addition to two heuristics to solve it. These latter address each of the DA-SFCS sub-problems heuristically but solves the deadline-aware scheduling sub-problem to optimality through the DAS-MILP. Our numerical experiments show that our LAMS and RMS algorithms can provide good solutions and are much more scalable than the DA-SFCS-MILP. However, our TS-DA-SFCS heuristic has the best performance in terms of runtime.

REFERENCES

[1] Osman NC Yilmaz, Y-P Eric Wang, Niklas A Johansson, Nadia Brahmi, Shehzad A Ashraf, and Joachim Sachs. Analysis of ultra-reliable and low-latency 5g communication for a factory automation use case. In *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pages 1190–1195. IEEE, 2015.

[2] Sunny Dutta, Tarik Taleb, and Adlen Ksentini. Qoe-aware elasticity support in cloud-native 5g systems. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[3] Ali Imran, Ahmed Zoha, and Adnan Abu-Dayya. Challenges in 5g: how to empower son with big data for enabling 5g. *IEEE Network*, 28(6):27–33, 2014.

[4] Maria A Lema, Andres Laya, Toktam Mahmoodi, Maria Cuevas, Joachim Sachs, Jan Markendahl, and Mischa Dohler. Business case and technology analysis for 5g low latency applications. *IEEE Access*, 5:5917–5935, 2017.

[5] Tarik Taleb, Adlen Ksentini, and Riku Jantti. " anything as a service" for 5g mobile systems. *IEEE Network*, 30(6):84–91, 2016.

[6] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. Making middleboxes someone else's problem: network processing as a cloud service. *ACM SIGCOMM Computer Communication Review*, 42(4):13–24, 2012.

[7] Justine Sherry, Sylvia Ratnasamy, and Justine Sherry At. A survey of enterprise middlebox deployments. 2012.

[8] Md Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, and Raouf Boutaba. On orchestrating virtual network functions in nfv. *CoRR abs/1503.06377*, 2015.

[9] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. 2015.

[10] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Steven Latré, Marinos Charalambides, and Diego Lopez. Management and orchestration challenges in network functions virtualization. *Communications Magazine, IEEE*, 54(1):98–105, 2016.

[11] Xuan Zhou, Rongpeng Li, Tao Chen, and Honggang Zhang. Network slicing as a service: enabling enterprises' own software-defined cellular networks. *IEEE Communications Magazine*, 54(7):146–153, 2016.

[12] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Steven Davy. Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pages 1–9. IEEE, 2015.

[13] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *Communications Magazine, IEEE*, 53(2):90–97, 2015.

[14] Ruozhou Yu, Guoliang Xue, Vishnu Teja Kilari, and Xiang Zhang. Network function virtualization in the multi-tenant cloud. *Network, IEEE*, 29(3):42–47, 2015.

[15] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati. Network function virtualization in 5g. *IEEE Communications Magazine*, 54(4):84–91, 2016.

[16] Sophia-Antipolis Cedex. Etsi gs nfv 003 v1.2.1: Network functions virtualisation (nfv); terminology for main concepts in nfv, etsi ind. spec. group (isg) netw. functions virtualisation (nfv), 2014.

[17] Long Qu, Chadi Assi, and Khaled Shaban. Network function virtualization scheduling with transmission delay optimization. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 638–644. IEEE, 2016.

[18] Jordi Ferrer Riera, Eduard Escalona, Josep Batalle, Eduard Grasa, and Joan A Garcia-Espin. Virtual network function scheduling: Concept and challenges. In *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*, pages 1–5. IEEE, 2014.

[19] Bernardetta Addis, Dallal Belabed, Mathieu Bouet, and Stefano Secci. Virtual network functions placement and routing optimization. In *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pages 171–177. IEEE, 2015.

[20] Long Qu, Chadi Assi, and Khaled Shaban. Delay-aware scheduling and resource optimization with network function virtualization. *IEEE Transactions on Communications*, 64(9):3746–3758, 2016.

[21] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[22] Md Golam Rabbani, Rafael Pereira Esteves, Maxim Podlesny, Gwendal Simon, Lisandro Zambenedetti Granville, and Raouf Boutaba. On tackling virtual data center embedding problem. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 177–184. IEEE, 2013.

[23] Xin Li and Chen Qian. A survey of network function placement. In *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 948–953. IEEE, 2016.

[24] Xin Li and Chen Qian. The virtual network function placement problem. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 69–70. IEEE, 2015.

[25] Li Erran Li, Vahid Liaghat, Hongze Zhao, MohammadTaghi Hajiaghayi, Dan Li, Gordon Wilfong, Y Richard Yang, and Chuanxiong Guo. Pace: Policy-aware application cloud embedding. In *INFOCOM, 2013 Proceedings IEEE*, pages 638–646. IEEE, 2013.

[26] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 242–253. ACM, 2011.

[27] Jeffrey C Mogul and Lucian Popa. What we talk about when we talk about cloud network performance. *ACM SIGCOMM Computer Communication Review*, 42(5):44–48, 2012.

[28] Jeongkeun Lee, Yoshio Turner, Myungjin Lee, Lucian Popa, Sujata Banerjee, Joon-Myung Kang, and Puneet Sharma. Application-driven bandwidth guarantees in datacenters. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 467–478. ACM, 2014.

[29] Hyame Assem Alameddine, Sara Ayoubi, and Chadi Assi. Offering resilient and bandwidth guaranteed services in multi-tenant cloud networks: Harnessing the sharing opportunities. In *Teletraffic Congress (ITC 28), 2016 28th International*, volume 1, pages 1–9. IEEE, 2016.

[30] Hyame Assem Alameddine, Sara Ayoubi, and Chadi Assi. Protection plan design for cloud tenants with bandwidth guarantees. In *Design of Reliable Communication Networks (DRCN), 2016 12th International Conference on the*, pages 115–122. IEEE, 2016.

[31] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. Near optimal placement of virtual network functions. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 1346–1354. IEEE, 2015.

[32] Tachun Lin, Zhili Zhou, Massimo Tornatore, and Biswanath Mukherjee. Demand-aware network function placement. *Journal of Lightwave Technology*, 34(11):2590–2600, 2016.

[33] Alon Itai. Two-commodity flow. *Journal of the ACM (JACM)*, 25(4):596–611, 1978.

[34] Christian Artigues, Sophie Demassey, and Emmanuel Nron. *Resource-Constrained Project Scheduling*. Wiley, 2008.

[35] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, 207(1):1–14, 2010.