Virtual Function Placement for Service Chaining with Partial Orders and Anti-Affinity Rules

Zaid Allybokus
Huawei Technologies, France Research Center

Nancy Perrot
Orange Labs, Châtillon, France

Jérémie Leguay and Lorenzo Maggi
Huawei Technologies, France Research Center, Boulogne Billancourt, France

Eric Gourdin
Orange Labs, France

Software-Defined Networking and Network Function Virtualization are two paradigms that offer flexible software-based network management. Service providers are instantiating Virtualized Network Functions, for example, firewalls, DPIs, gateways—to highly facilitate the deployment and reconfiguration of network services with reduced time-to-value. They use Service Function Chaining technologies to dynamically reconfigure network paths traversing physical and virtual network functions. Providing a cost-efficient virtual function deployment over the network for a set of service chains is a key technical challenge for service providers, and this problem has recently caught much attention from both Industry and Academia. In this article, we propose a formulation of this problem as an Integer Linear Program that allows one to find the best feasible paths and virtual function placement for a set of services with respect to a total financial cost, while taking into account the (total or partial) order constraints for Service Function Chains of each service and other constraints such as end-to-end latency, anti-affinity rules between network functions on the same physical node and resource limitations in terms of network and processing capacities. Furthermore, we propose a heuristic algorithm based on a linear relaxation of the problem that performs close to optimum for large scale instances. © 2017 Wiley Periodicals, Inc. NETWORKS, Vol. 000(00), 000–000 2017

1. INTRODUCTION

In addition to packet forwarding equipment such as switches and routers, service provider networks commonly include middleboxes, also called *Service Functions* (SFs), such as firewalls, Deep Packet Inspections (DPIs), web proxies, media gateways, and so forth. These SFs are traditionally provided with a specific hardware. Their deployment and reconfiguration thus suggest manual interventions in order to connect them to the network via wired cables, which is costly and time consuming. Nowadays, SFs are delivered as Virtual Machines (VM) that can run on off-the-shelf servers thanks to Network Function Virtualization (NFV) technologies [9]. They have yet to be improved to provide comparable performances: they still usually represent less effective versions of their hardware peers in terms of packet processing capacity and speed. However, their reduced cost and generic hardware compatibility allow them to be duplicated easily and to adapt quickly to evolving traffic. In addition to this transformation, Software-Defined Networking (SDN) offers the unprecedented possibility to separate the control plane from the data plane, providing high programmability in the management and control of individual flows. Consequently, SDN, along with NFV are two elegant paradigms that offer flexible software-based network management.

One major challenge for service providers is to deploy *Service Function Chains* (SFC), where a customer demand has to traverse SFs in a predefined, flow-specific order.

For instance, a link connecting two enterprise sites may need to pass through a firewall, a DPI, and a web proxy. While SFCs have been initially defined as a set of network functions that must be traversed in a given order, this model can be refined to better meet service providers needs and optimize resources. While order constraints may impose a

given function, say a classifier, to be traversed prior to a WAN optimizer, some functions (e.g., a firewall) may not have ordering constraints. Recent studies [2] have showed that this flexibility optimizes resource usage as it may reduce the number of activated functions and decrease the network footprint.

For this reason, we believe the possibility to set partial order constraints is an interesting feature. In addition, as mentioned in requirements [1] from the ETSI NFV standardization group, the so called *anti-affinity* rules are generally used to reduce the impact of an infrastructure failure. More specifically, anti-affinity rules prevent certain Virtual Network Functions (VNF) within the same SFC from sharing the same physical resources. This increases the robustness of the service against one node failure at the same time.

Motivated by this, in this paper we address the joint VNF Placement and Routing (VNF-PR) problem for Service Function Chaining in the case of partially ordered SFCs satisfying the anti-affinity rules. To the best of our knowledge, the partial orders and anti-affinity rules we consider in this work have not been investigated before.

We first provide an Integer Linear Program (ILP) formulation that jointly searches for an optimal placement and an individual path for each service chain, while minimizing the overall cost of jointly traffic engineering and VNF instantiation. Our formulation is naturally flexible and permits one to relax orders easily. The unconstrained VNF-PR problem is known to be NP-hard [4] as it combines integer versions of the facility location and multi-commodity flow problems. Therefore, heuristics that well approximate the optimum in a reasonable amount of time are a must for service providers and carriers. We thus design a heuristic based on the linear relaxation of our ILP formulation and we show numerically its effectiveness. We also evaluate its performance on the realistic GEANT network topology, using the ILP as performance benchmark and by comparing it to a greedy algorithm that treats commodities in sequence.

The article is organized as follows. We first present related works in Section 2. Section 3 states formally the problem while in Section 4 we present the ILP formulation, providing the optimal solution to our problem. As a warm up, Section 5 introduces the naive, greedy algorithm to tackle the problem. In turn, we propose in Section 6 a heuristic approach. Our numerical evaluation is then commented upon in Section 7, and finally, Section 8 concludes the article.

2. RELATED WORK

NFV leverages IT virtualization technology to consolidate many network services (network address translation, ciphering, firewalling, intrusion detection, caching, and so forth) onto standard high-volume servers which could be located at Points of Presence, data centers, or end user premises [14].

The VNF-PR problem is to allocate server and network resources such that a given set of service chains can be deployed onto the network at minimum cost. A few papers

[2, 13, 15] have recently proposed ILP formulations considering constraints on end-to-end latency, network capacity, and computing resources (e.g., maximum number of chains and functions, maximum amount of traffic). Moens et al. [15] consider hybrid chains that can include both physical and virtual resources. Mehraghdam et al. [13] define a grammar to characterize network chains in which one of the primitives is *optional order* to relax order constraints between functions.

Nevertheless, this grammar is not part of their ILP formulation. The authors of [13] propose an exhaustive search to find all possible permutations and then, they solve the unconstrained ILP for each of them. Addis et al. [2] consider a formulation without any order constraints and that leverages traffic compression properties from multimedia gateways (e.g., video adaptation) or firewalls to reduce the network footprint. They propose a linearization of the model. Our main contribution to the problem formulation is to consider partial orders and anti-affinity rules.

A few algorithms have already been proposed to solve the VNF-PR problem with latency and resource constraints. Ghaznavi et al. [8] propose a local search heuristic to tradeoff between accuracy and speed. The minimum improvement at each step is used as a tunable parameter. Luizelli et al. [10] use a binary search to find the fewest possible number of functions and solve a reduced ILP. Both papers consider service chains with a fully determined order of functions. Cohen et al. [4] study a bi-criteria approximation algorithm with a factor of 8 for the cost minimization objective function and a factor of 16 for the size constraints at the nodes. Beyond offline problems, deterministic online algorithms have been proposed for Service Function Chaining by Lukovszki et al. [12] or for the incremental deployment of a single type of function [11] by the same authors.

Several research works [6] have proposed solutions to the *Virtual Network Embedding* (VNE) problem, where several virtual networks have to be mapped onto a physical substrate. However, the problems of embedding virtual networks differ from the VNF-PR problem as virtual nodes of the same type *but from the different requests* must share the same substrate node if it is cost efficient. Some authors relate the VNF-PR problem with full order constraints to the VNE literature [3, 8], where SFCs are modeled as Virtual Network requests (VNr). Fuerst et al. [7] propose a preclustering algorithm that enables collocation of virtual nodes (or SFs) of *the same* VNr (or SFC) onto the same substrate node. However, we argue that the VNE literature neither responds to the mutualization of virtual nodes (SFs) of the same type between different VNrs (SFCs) nor to the partial orders within the same SFC. As such, to the best of our knowledge, these two problems are separate.

Our contribution is twofold. We first propose an ILP formulation of the VNF-PR problem with partial order and anti-affinity constraints which we reinforce by adding redundant cuts. Second, we propose and evaluate a heuristic approach based on a linear relaxation to solve the problem.
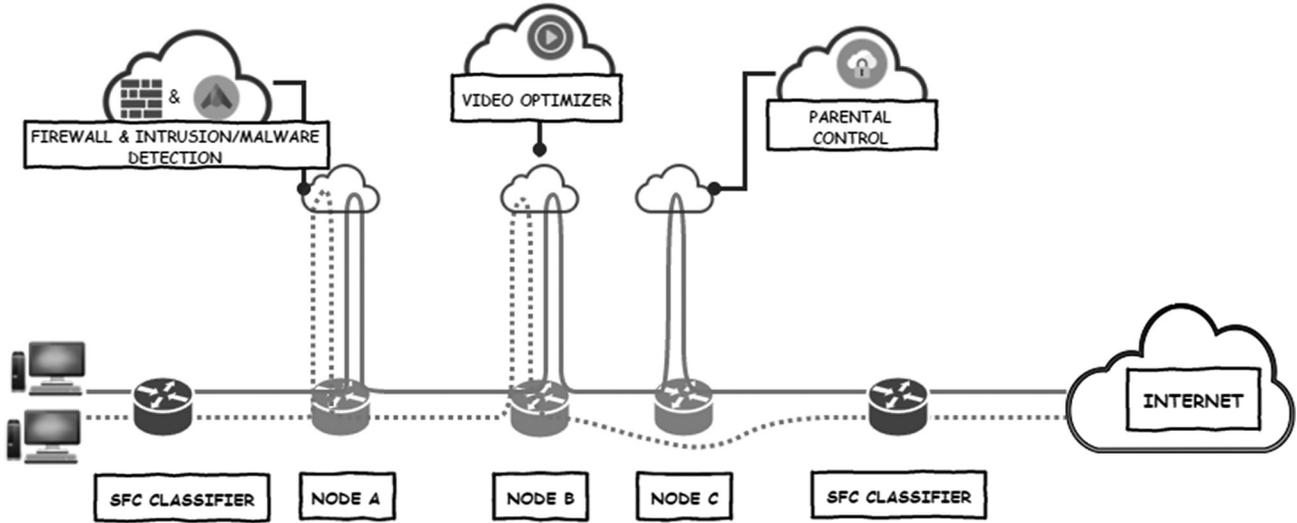
FIG. 1. Service Function Chaining use case. In this example, two SFCs share VNFs deployed over the network nodes to build two SFPs. The first SFP (dashed line) includes a Firewall, an Intrusion/Malware Detection and a Video Optimizer instantiated on the two first nodes. In addition to the three VNFs of the first SFP, the second commodity (solid line) needs a special hop to a third node containing a Parental Control Function.

3. PROBLEM STATEMENT

We consider a network whose nodes, in addition to their traditional role of handling packet forwarding, are equipped with processors that allow them to support the VNFs that are required by the different connection requests, which we will now refer to as *commodities*. They contain a limited number of cores that can each support a function instantiation. Each instantiation of those functions can handle multiple commodities up to a certain maximum rate (function capacity). If needed, functions can be replicated on the same node. However, this replication demands the dedication of an additional core. Each commodity is provided with an SFC that needs to be deployed along its chosen path. This list can be unordered, totally ordered, or partially ordered. A Service Function Path (SFP) is the physical path through an SFC instantiation that is chosen for a commodity.

Instantiating a service function has a cost (which may contain the price for set up, license, and maintenance), which motivates the need to assign as many commodities as possible to active instances if they have enough capacity.

Moreover, routing traffic through physical links has a per-rate cost and motivates the need to preferably allocate shortest paths.

The anti-affinity rules seek to enforce the reliability of a VNF structure. They forbid the assignment of certain VNFs to one commodity on the same location to ensure that SFPs do not entirely break down by a single server failure.

In addition to the orders, the anti-affinity rules may therefore force routing at higher costs (or longer paths) in order to run network functions at reduced cost. We seek to tackle this trade off by analyzing the problem by means of a total financial cost. Our aim is to find an SFP for all commodities while minimizing the cost of function instantiation and traffic engineering together. Figure 1 illustrates a concrete
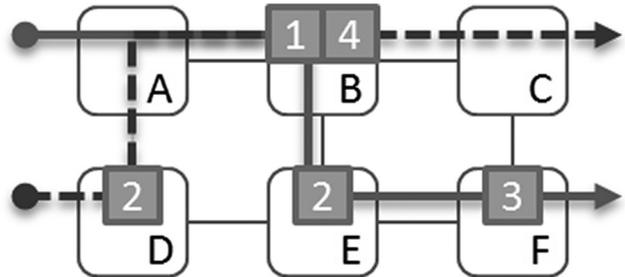

FIG. 2. SFCs and their SFP allocation. In this example, the solid SFP has SFC [1; 2; 3] and the dashed SFP has SFC [2; 1; 4]. This allocation could be improved in terms of VNF costs by mutualizing VNF 2 on node E, and the dashed SFP would become: in $\rightarrow D \rightarrow E \rightarrow B \rightarrow C \rightarrow$ out..

Service Function Chaining use case, and a simple example of SFCs and their instantiation on suitable SFPs is illustrated in Figure 2.

4. ILP FORMULATION

In this section, we start by presenting an ILP Formulation that addresses exactly our problem.

4.1. Model Parameters

The network topology is modeled as a bidirected graph $G = (N, A)$. Each node u of the graph corresponds to a physical switch equipped with a small data center that has a certain number $c_u \in \mathbf{N}$ of cores, and each link (u, v) corresponds to a wired connection between two switches with a bandwidth capacity $b_{uv} \in \mathbf{R}_+$ and a delay (latency) $l_{uv} \in \mathbf{R}_+$. We refer to F as the set of available SFs. This set can be in general large but we remark that not all functions in F need to be used for building SFPs. The set of commodities is denoted

C and each commodity $k \in C$ is characterized by a source $s_k \in N$, a destination $d_k \in N$, a bandwidth demand $b_k \in \mathbf{R}_+$, a maximum tolerated latency $l_k \in \mathbf{R}_+$ and a specific SFC $F_k \subset F$. For simplicity, we assume that each function occupies one core and has a maximum traffic rate it can process, $m_f \in \mathbf{R}_+$. It is however possible to support VNF-specific core requirements for different VNFs by modifying the node capacity constraints (4) (see Section 4.3).

The fixed relative orders between VNFs that take part in the SFC of a commodity k are given by a ternary operator p_k that maps the tuple of VNFs (f, g) to the value -1 (resp. 1) if f appears after (resp. before) g along the SFC. When no order is required between f and g, $p_k(f, g) = p_k(g, f) = 0$. This parameter depicts a well-defined total or partial order. The cost of routing per unit of traffic per link (u, v) is given by the number ψ_{uv} and the one for running VNF f is ψ_f.

Finally, we model anti-affinity rules as a set $S \subset F^2$ which contains the (unordered) tuple (f, g) if and only if these two VNFs cannot handle the same service chain on the same node: hence, if they coexist on the same node, then there are at least two different commodities that are assigned to exclusively one of them on that node.

The set F may contain several configurations of the same VNF type if one wants to model different setup states which make some SFPs unable to share the same VNF instance. For instance, if SFP k and SFP k' need to meet a certain VNF f, with a setup s and s' respectively, that cannot be provided by the same instantiation, then each of these SFPs need to meet a different instantiation of the VNF f, each of them being in the correct setup state. Our model enforces this feature by considering the two setups as different VNF types. In the same way, this trick may permit the service provider to feature commodity isolation by exclusively allocating a commodity to a VNF instance.

For more clarity, Table 1 gives an overview of the parameters and variables.

TABLE 1.   Variables and parameters.

Parameters	
s_k	Source node of commodity k
d_k	Destination node of commodity k
b_k	Demand of commodity k
c_u	Number of cores of node u
m_f	Maximum rate for function f
$p_k(f, g)$	1 (resp. -1) if f appears before (resp. after) g
	On SFC of commodity k, 0 otherwise
b_{uv}	Bandwidth of link (u, v)
l_{uv}	Latency of link (u, v)
l_k	Maximum tolerated latency for commodity k
ψ_{uv}	Cost per unit of traffic on link (u, v)
ψ_f	Cost of running function f

Variables		Within
Y_u^{kf}	1 if k meets its assigned f before or on node u	Binary
y_u^{kf}	1 if function f handles packet k on node u	Binary
x_u^f	Number of f located on node u	Integer
q_{uv}^k	1 if path for k contains link (u, v)	Binary

4.2. Decision Variables

We introduce the ILP exact formulation of the VNF-PR problem with partial order and anti-affinity rules. We denote by q_{uv}^k the routing binary variables that state whether link (u, v) is used for the path of commodity k or not. The number of required instantiations of VNF f on node u is denoted by $x_u^f \in \mathbf{N}$, and we denote by y_u^{kf} the binary variable stating whether VNF f is assigned to commodity k on node u or not. To address the order constraints, we define an auxiliary variable Y_u^{kf} that accounts for the state of the commodity: $Y_u^{kf} = 1$ when on node u, the commodity has already been assigned to f at some point along its route to u or on u itself.

We will represent the orders between VNFs with orders between values of this variable.

4.3. Constraints

We now turn to the formulation of our ILP's constraints. The first constraints are the classic Capacitated Multi-Commodity Flow constraints, enforcing conservation laws of the flows (1), the nonviolation of the bandwidth limitations of each link (2), and the delay maximum tolerance (3):

$$\sum_{(u,v)\in A} q_{uv}^k - \sum_{(v,u)\in A} q_{vu}^k = \begin{cases} 1 & \text{if } u = s_k \\ -1 & \text{if } u = d_k \quad \forall k \in C \; \forall u \in N \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\sum_{k\in C} q_{uv}^k b_k \leq b_{uv} \quad \forall (u, v) \in A \tag{2}$$

$$\sum_{(u,v)\in A} q_{uv}^k l_{uv} \leq l_k \quad \forall k \in C. \tag{3}$$

The following constraints state that a node cannot support more VNFs than its number of available cores (4), and that each function instantiation has a limited capacity and needs to be duplicated enough to handle all commodities that are assigned to it on the corresponding node (5):

$$\sum_{f\in F} x_u^f \leq c_u \quad \forall u \in N \tag{4}$$

$$\sum_{k\in C} y_u^{kf} b_k \leq m_f x_u^f \quad \forall u \in N \quad \forall f \in F. \tag{5}$$

The anti-affinity rules modeled by the set S can be formulated as follows: given a tuple of two functions that belongs to S, no commodity can be assigned to both of them on the same location.

$$y_u^{kf} + y_u^{kg} \leq 1 \quad \forall k \in C \quad \forall (f, g) \in S \cap F_k^2 \quad \forall u \in N. \tag{6}$$

The following constraints allow one to properly impose the orders within each SFC. Moreover, one can easily establish that they also avoid the creation of cycles along a path:

$$Y_{s_k}^{kf} = 0 \quad \forall k \in C \quad \forall f \in F \tag{7}$$

$$Y_{d_k}^{kf} = 1 \quad \forall k \in C \quad \forall f \in F_k \tag{8}$$

$$(q_{uv}^k - 1) + (Y_v^{kf} - Y_u^{kf}) \leq y_v^{kf} \quad \forall k \in C \quad \forall f \in F_k$$
$$\forall (u, v) \in A \tag{9}$$

$$(Y_u^{kf} - Y_u^{kg})p_k(f, g) \geq 0 \quad \forall k \in C \quad \forall u \in N \quad \forall f, g \in F_k. \tag{10}$$

Constraints (7) and (8) state that at the source, a commodity has met none of the functions, whereas at the destination, it has met all the functions included in its SFC, respectively. Constraint (10) accounts for the order of the functions within the same SFC: when a function g appears before f on the same SFC (therefore, $p_k(f, g) = -p_k(g, f) = -1$), at each node, we should have $Y_u^{kf} \leq Y_u^{kg}$. This constraint becomes trivial when no order is given between two functions of an SFC. According to constraint (9), when the link (u, v) is used - otherwise the constraint is always feasible—for commodity k, the commodity meets its assigned copy of VNF f on v if its state variable Y_v^{kf} jumps from 0 to 1. Constraint (8) ensures that this jump happens once, and constraint (10) ensures that the jumps occur in the correct order given by p_k. Those four constraints therefore give a profile of the Y variable that along each chosen path reads the state of the commodity with respect to function traversal constraints.

Moreover, in order to improve the linear relaxation (LP) bound, we add the following constraints:

$$y_u^{kf} + y_u^{kg} \leq \sum_{(v,u) \in A} q_{vu}^k \quad \forall k \in C \quad \forall u \in N \quad \forall (f, g) \in S \tag{11}$$

$$y_u^{kf} \leq \sum_{(v,u) \in A} q_{vu}^k \quad \forall f \in F \quad \forall u \in N \quad \forall k \in C \tag{12}$$

$$\sum_{u \in N} y_u^{kf} = \begin{cases} 1 & \text{if } f \in F_k \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in C \quad \forall f \in F \tag{13}$$

$$y_u^{kf} \leq x_u^f \quad \forall u \in N \quad \forall f \in F \quad \forall k \in C. \tag{14}$$

Constraint (11) specifies the anti-affinity rules in terms of fractional rates: no traffic is assigned in its totality to two incompatible VNFs on the same location. In turn, constraint (12) states that an assignment can be made on a node only if the commodity passes through this node. Constraint (13) sets the total value of a required assignment to 1 to avoid unnecessary additional assignments. Finally, constraint (14) ensures that no assignment can be made if a VNF is not instantiated. It is redundant with constraint (5) but this kind of cut is well known to improve the LP-bound.

4.4. Formulation

Our aim is to minimize the total financial cost encompassing the routing (ψ_{uv}) and VNF (ψ_f) costs. Therefore, we formulate the VNF Placement for Service Chaining with Partial Orders and Anti-Affinity Rules ILP as the following:

$$\min \sum_{(u,v) \in A, k \in C} q_{uv}^k b_k \psi_{uv} + \sum_{u \in N, f \in F} x_u^f \psi_f$$

$$\text{s.t.} \quad (1)\text{--}(14)$$

$$q_{uv}^k, y_u^{kf}, Y_u^{kf} \in \{0, 1\}, x_u^f \in \mathbf{N}.$$

4.5. Handling Commodity Rejection

Our model formulation is flexible enough to easily incorporate rejection decisions. First of all, this feature allows one to always have a non-empty admissible set that consists in artificial solutions where commodities are allocated and others are rejected. This variation can also respond to situations where resources are congested and a decision has to be made on which commodities to admit in the network and which ones to reject. In this work, we use this feature to handle complicated SFCs (see Section 6.2) for which the fractional optimal solution does not provide a feasible SFP. To provide this option, we introduce a binary decision variable, $\{\bar{q}_k\}_{k \in C}$, that equals 1 when a commodity is rejected or 0 when it is accepted in the network. This variable will be added to the flow conservation constraints to allow a certain commodity k flow to be identically zero:

$$\sum_{(u,v) \in A} q_{uv}^k - \sum_{(v,u) \in A} q_{vu}^k = \begin{cases} 1 - \bar{q}_k & \text{if } u = s_k \\ \bar{q}_k - 1 & \text{if } u = d_k \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\forall k \in C \quad \forall u \in N. \tag{15}$$

In addition, when a commodity is rejected, there is no VNF assigned to it:

$$y_u^{kf}, Y_u^{kf} \leq 1 - \bar{q}_k \quad \forall k \in C \quad \forall f \in F \quad \forall u \in N \tag{16}$$

and constraint (8) is replaced with constraint (17):

$$Y_{d_k}^{kf} = 1 - \bar{q}_k \quad \forall f \in F_k \quad \forall k \in C. \tag{17}$$

In turn, we replace the constraint (13) by the following:

$$\sum_{u \in N} y_u^{kf} = \begin{cases} 1 - \bar{q}_k & \text{if } f \in F_k \\ 0 & \text{otherwise} \end{cases} \quad \forall f \in F \quad \forall k \in C. \tag{18}$$

The new ILP thus reads:

$$\min \sum_{(u,v) \in A, k \in C} q_{uv}^k b_k \psi_{uv} + \sum_{u \in N, f \in F} x_u^f \psi_f + r \sum_{k \in C} \bar{q}_k b_k$$

$$\text{s.t.} \quad (2)\text{--}(7), (9)\text{--}(12), (14)\text{--}(18)$$

$$\bar{q}_k, q_{uv}^k, y_u^{kf}, Y_u^{kf} \in \{0, 1\}, x_u^f \in \mathbf{N}, \tag{19}$$

where r is a sufficiently large parameter, for instance:

$$r > \frac{\max b_k |A||C|\psi_{uv} + |F||N| \max c_u \max \psi_f}{\min b_k}.$$

From now on, we always consider this variant of the problem.

5. GREEDY ALGORITHM

We first illustrate a simple greedy algorithm (Algorithm 1) that we will compare with the LP-based heuristic approach proposed in the next section. The greedy approach treats commodities in a sequential fashion: it allocates the commodities on the network one-by-one, while instantiating VNFs on-the-fly that can remain available for subsequent demands. Of course, the algorithm may reject commodities when their allocation is infeasible. It is simple to formulate an ILP problem that serves our needs and that treats commodities in sequence: we define an additional parameter that accounts for the actual use rate of VNFs on each location given previous allocations. At each step, the algorithm can decide to either instantiate new VNFs or use already existing ones, depending on which is less costly. Let $\left\{r_u^f\right\}$ be this family of parameters. Constraint (5) then becomes for step k, for each node u and each VNF f :

$$y_u^{kf} b_k + r_u^f \leq m_f x_u^f. \tag{20}$$

This rate has to be updated at each step output.

Algorithm 1 Greedy
___
Input: Network Topology (N, A), set of commodities C and their SFCs
Output: Integer solution $\left\{q_{uv}^k\right\}_{k,(u,v)}, \left\{y_u^{kf}\right\}_{u,k,f}, \left\{x_u^f\right\}_{u,f}$ and a rejected set S
1: $r_u^f = 0 \quad \forall u, \forall f, S = \emptyset$
2: **for** $k \in C$ **do**
3:     **if** k is feasible **then**
4:         allocate k with ILP
5:     **else**
6:         $S \leftarrow S \cup \{k\}$
7:     **end if**
8:     update x
9:     update all link capacities
10:     update use rates r_u^f for each newly installed f on each node u
11: **end for**

---

6. LP-BASED HEURISTIC

To solve efficiently our problem, we propose a heuristic approach based on the LP-relaxation of the original ILP in 4. Before delving into the details of our heuristic algorithm, we first describe in the next paragraph its main subroutine.

6.1. Restricted Problem: VNF Deployment on Selected Routes

We now describe a subroutine used by our heuristic approach, which optimizes the VNF deployment given pre-computed routes. As a by-product, the following subroutine can be thought of as a stand-alone routine, which is useful
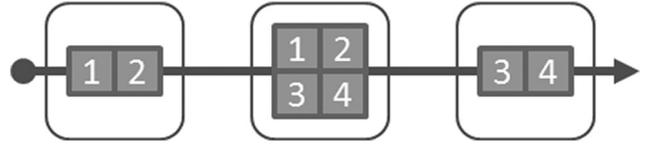

FIG. 3. An example of an admissible LP-solution that does not provide integer solution. In this example, the commodity's SFC is [1;2;3;4], and all couples of functions are disjoint. The LP-SFP where all assignment values equal 0.5 is admissible but the shortest integer SFP has length 4.

FIG. 4. Any feasible path allocation for the SFC in Figure 3 requires at least one hop between each VNF traversal.

for reallocating all VNFs on a set of established routes, for example, after costly deviations caused by greedy allocations of SFPs, or after a resource breakdown which would demand a new deployment. To this aim, the rejection variables are here added to the constraint (9) so that they always remain feasible for rejected commodities:

$$(q_{uv}^k - 1) + (Y_v^{kf} - Y_u^{kf}) \leq y_v^{kf} + \bar{q}_k \tag{21}$$

The ILP we provide to address this situation hence reads:

$$\min \sum_{u \in N, f \in F} x_u^f \psi_f + r \sum_{k \in C} \bar{q}_k b_k$$

$$\text{s.t.} \quad (4)\text{–}(7), (9)\text{–}(12), (14), (16)\text{–}(18)$$

$$y_u^{kf}, Y_u^{kf}, \bar{q}_k \in \{0, 1\}, x_u^f \in \mathbf{N}.$$

6.2. Algorithm

We are finally ready to describe our heuristic approach to jointly compute the routing paths and the virtual functions placement with partial order and anti-affinity constraints. Algorithm 2 outlines the basic steps of our approach.

Given the optimal solution of the relaxed version of the integer problem (19), we use a rounding scheme to retrieve a best candidate path for each commodity. We then deploy the VNFs optimally with respect to this path selection using the restricted problem described above. We claim that the fractional paths given by the LP relaxation are good candidates for feasible SFPs. However, this may not necessarily be the case for all SFCs. A complicating combination of anti-affinity rules and precedence constraints may blur the LP evaluation of the feasible minimal number of hops for a commodity's unsplittable SFP. We give a simple example of a complicating order and anti-affinity combination on an SFC in Figures 3 and 4 to illustrate this situation. In those cases, we retrieve rejected commodities and allocate them at the end of the procedure with the greedy algorithm. The concerned commodities can then be allocated on a path built on new VNF instantiations, already existing VNFs, or both.

As a result, the validity of the LP solution is a crucial question that deserves commenting on. The linearized variables $q, \bar{q}, y$ and Y keep their definitions, except they always account for the fraction of the traffic given by their value. In particular, for $k \in C, f \in F_k, u \in N, y_u^{kf}$ defines the portion of the commodity k that is assigned to VNF f on node u, and Y_u^{kf} defines the fraction of commodity k that has already met VNF f during its route to u. As such, they do not entirely enforce anti-affinity rules and orders. One could further add linear cuts in order to improve the quality of the relaxation, but we remark that this has the tendency to considerably increase the solver execution time for large scale instances. Moreover, we argue that complicated overlapping of anti-affinity rules and precedence constraints should be relatively rare and can be considered as particular cases for which it is acceptable to handle a first rejection at the end of the procedure. We remark that SelectPath procedure, at line 3 of Algorithm 2, stands for any rounding procedure that converts a fractional solution into a feasible path over the capacitated graph. In our numerical evaluations, for instance, we implemented Select-Path as a deterministic walk that, for each commodity, starts at the source and adds the (not yet visited) link with highest traffic rate to the path until it reaches the destination. For example, in [5], the authors form an unsplittable SFP from the fractional optimum by following a random walk upon the support of the fractional SFP.

Algorithm 2   Heuristic
_____
Input: Network Topology (N, A), set of commodities C and their SFCs
Output: Integer solution $\left\{ q_{uv}^k \right\}_{k,(u,v)}, \left\{ y_u^{kf} \right\}_{u,k,f}, \left\{ x_u^f \right\}_{u,f}$ and a rejected set S
1: $S = \emptyset$
2: $\hat{q}_{uv}^k \leftarrow$ route values from LP-optimal solution (ILP (19))
3: $q \leftarrow$ SelectPath$(\hat{q})$
4: Solve ILP for VNF deployment on selected routes with q, C
5: **for** $k \in C$ **do**
6:      **if** k is allocated **then**
7:          update $y_u^{kf}, f \in F, u \in N$
8:      **else** *(k is rejected)*
9:          set $q_{uv}^k = 0 \quad \forall (u, v) \in A$
10:        $S \leftarrow S \cup \{k\}$
11:      **end if**
12: **end for**
13: define use rates as $r_u^f = \sum_{k \notin S} y_u^{kf} b_k, f \in F, u \in N$
14: update all network capacities
15: run Greedy on S with initialized use rates r_u^f
16: update q, y, x and S
_____

7. PERFORMANCE EVALUATION

This section presents the simulation setup, experimental results and overall observations.

7.1. Simulation Setup

We ran all our tests using as solver IBM ILOG CPLEX 12.6 and developed the nonoptimization parts of our algorithms in Python. Each point we represented corresponds to the average value for 15 instances generated on the GEANT topology which contains 22 nodes and 36 bidirected edges. For all our instances, the nodes each had a total of 20 available cores and the bandwidth limit of the links were each 40 Gb, while the VNFs capacities were fixed to 5 Gb. The average traffic cost per unit per link was fixed to 10\$/Mb/month and the average price of running a VNF was either 200\$/month or 1,000\$/month. The ratio between the two costs, denoted s, is in the former case 5%, and in the latter case 1% (costly VNFs). Those prices were generated uniformly within an interval of $\pm$ 20%. In all our instances, the commodity sources and destinations were chosen uniformly at random amongst the network nodes. In a first experiment, we show the scalability of the LP-based heuristic approach in terms of the number of commodities to allocate. Next, we point out its efficiency as we stress the orders and anti-affinity rules. All instances feature SFCs of lengths varying from 4 to 8 VNFs generated uniformly from a total of 10 existing VNFs. We only consider small demands from 100 Mb to 500 Mb so that no network resource is stressed in any of the instances. Indeed, in this work, we focus on the sole effects of orders and anti-affinity rules as we stress them.

7.2. Results

We now comment upon the results of our numerical investigations shown in Figures 5–7. Each figure compares the execution time, the optimality gap and the total cost of VNFs in different scenarios and present error bars accounting for an estimated 95%-confidence interval calculated following the t-distribution model.

In Figure 5, the instances feature partially ordered SFCs with six anti-affinity rules generated randomly. The cost ratio s (see Section 7.1) is fixed to 5%. According to Figure 5, the LP-based heuristic can solve the instances with up to 1,000 commodities whereas the ILP cannot manage all instances with 500 or 1,000 commodities within reasonable time. Our time performances remain acceptable up to 200 commodities whereas the optimality gaps do not exceed 2%.

In the next experiments (Figs. 6 and 7), more importance is given to the VNF costs by setting s to 1%, and we remark that this increases considerably all solver times. In order to keep the optimal value tractable, at least for small parameters, we fix the number of commodities to 100.

In Figure 6, we first analyze the performances as the number of order constraints increases. We use the "partial order" parameter $\theta = \frac{2|\text{supp}(p_k)|}{|F_k|(|F_k|-1)}$[1], in other words the number of precedence constraints over the total number of VNF couples within each SFC, and plot the results for
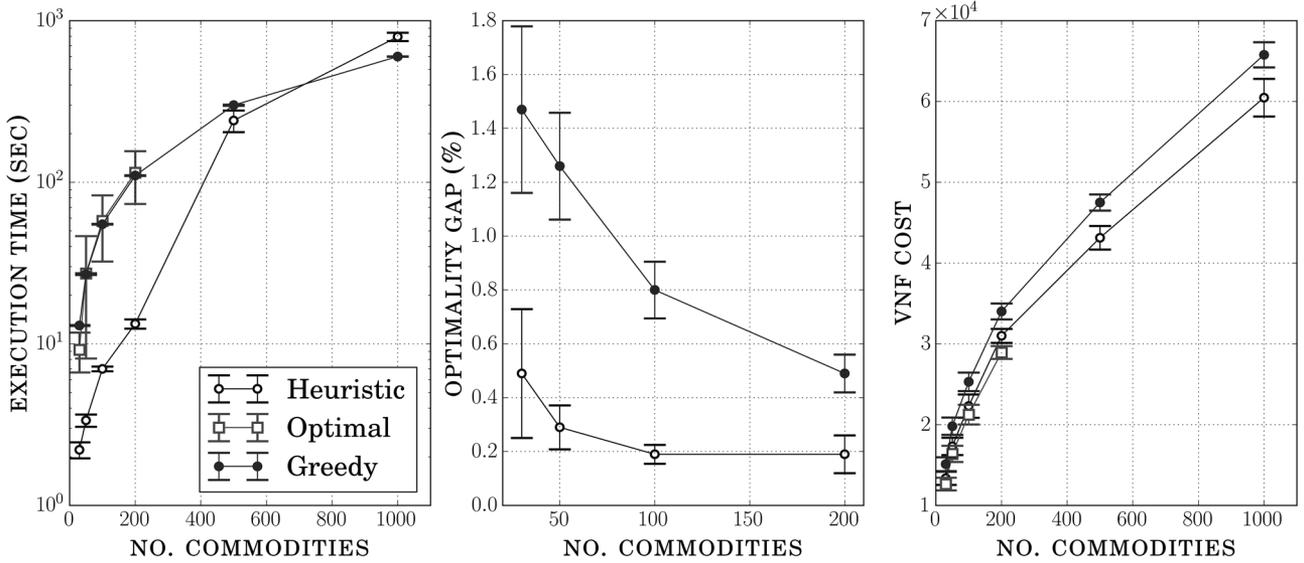

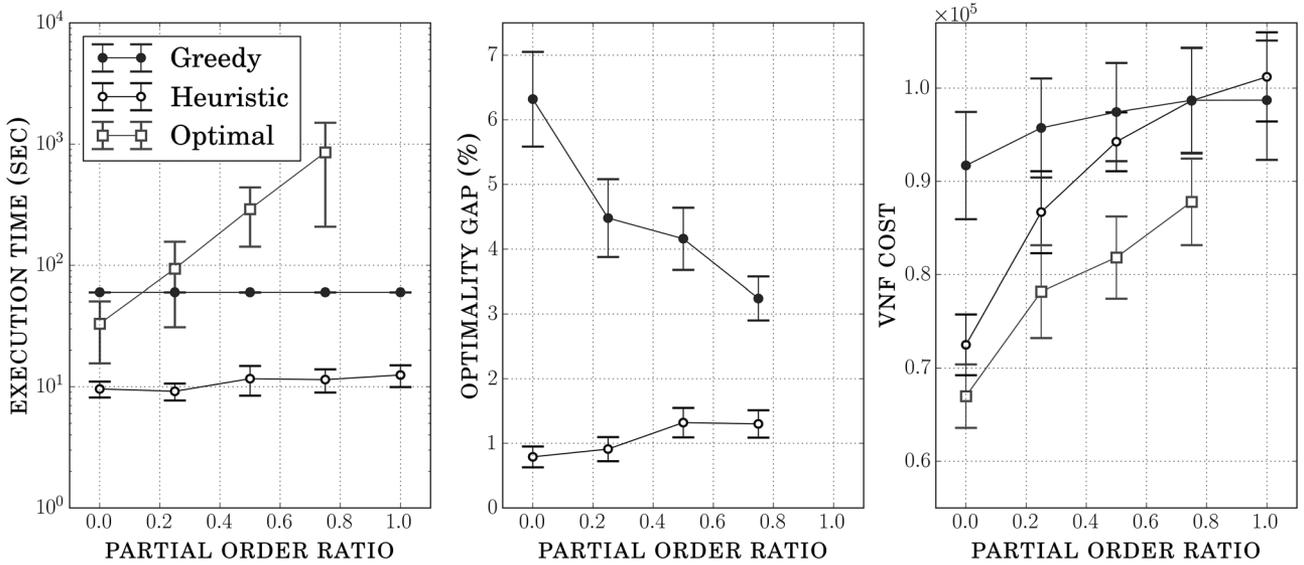FIG. 5. Scalability in terms of number of commodities.

FIG. 6. Order constraints influence.

$\theta = 0$ (no order), $0.25, 0.5, 0.75, 1$ (total order). The LP-based heuristic solves all instances within seconds whereas the ILP execution time quickly explodes. Regarding the VNF costs, one can remark that while the two algorithms, Greedy and the LP-based heuristic, seem to instantiate more VNFs, the optimality gaps remain quite satisfactory, especially for the LP-based heuristic. Indeed, the VNF cost is only a part of the total cost to minimize, the other part being the routing price, which the LP-based heuristic seems to optimize better.

Finally, we vary the number of anti-affinity rules and analyze the performance of the LP-based heuristic on the exact same scenario. Despite the low ratio $s = 1\%$, we observe in Figure 7 that the LP-based heuristic instantiates more VNFs than Greedy but routes at lower costs and hence performs better. The time performance indicates that the LP-based heuristic is a good solution to smooth out the sensitivity of

the problem with respect to the number of anti-affinity rules. The same remark can be made concerning the partial order parameter.

In all scenarios, the greedy and LP-based heuristic algorithms both give acceptable objectives, but the latter always gives closer to optimal results - not exceeding the optimal value by more than 2%. Moreover, Greedy cannot compete with the LP-based heuristic in terms of time performance. In the first experiment, Greedy is at first as slow as the ILP, as result of the numerous summoning of the solver to allocate each demand, whereas the LP-based heuristic can perform better and 4 to 8 times faster than the solver. The solver time explodes quickly as the precedence constraints become more numerous, whereas the LP-based heuristic is from 3 to 70 times faster and steady as the number of those constraints increase. The same phenomenon is observable
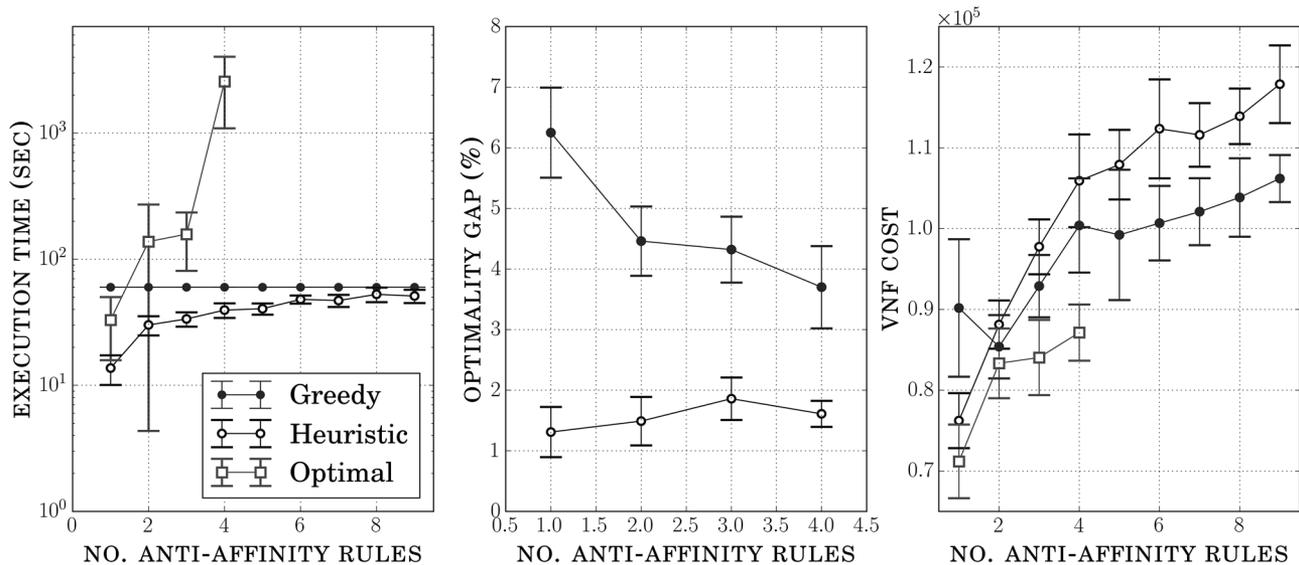
FIG. 7. Anti-affinity rules influence.

when we increase the number of anti-affinity rules, except that the growth of execution time for the LP-based heuristic corresponds to 1) the growth of LP time as the number of anti-affinity rules grows, 2) the more numerous commodities that are allocated greedily at the end of the procedure. Because of the solver's inability to solve some instances when they become large or highly constrained, the optimality gaps in all experiments are not always available.

7.3. Overall Observations

In a high-capacitated network in terms of link and nodes resources, the LP-based heuristic performs very close to optimum with a competitive time that beats the greedy approach in all the instances that we created up to 1,000 commodities. Through all runs, the crushing majority of the LP-based heuristic's execution time is used by the LP step. Onto the same scenario, we observe three stressing factors along which the execution time explodes: the number of precedence constraints within SFCs, the number of anti-affinity rules, and the cost ratio *s*. Beyond time performances, the numerical results show, as intuitively expected, that more flexibility in the orders can remarkably reduce the VNF costs, while anti-affinity rules force us to instantiate more VNFs. Our heuristic nevertheless smooths out the two difficulties and solves - with reduced and steady time—instances that the ILP cannot solve within reasonable time. Of course, those factors are not the only ones. The case where the node resources are stressed is still problematic. Nonetheless, the LP relaxation should be accurate enough to handle commodity rejection on networks with stressed link resources.

8. CONCLUSION AND PERSPECTIVES

In this article, we tackled the joint optimization problem consisting in placing SF and routing multiple commodities over a general capacitated network. Each commodity requires handling by a set of SFs with *partial* order constraints. The VMs running virtual functions have limited processing capabilities, while the nodes hosting VMs are characterized by limited memory. Moreover, *anti-affinity* rules forbid the assignment of certain VNFs to one commodity on the same location, in order to ensure that SF paths (SFPs) do not entirely break down by a single server failure.

Since the problem is NP-hard, we proposed a heuristic approach based on the relaxation of the optimal ILP formulation. We added redundant linear constraints to the original ILP formulation that permit us to reduce the optimality gap.

We evaluated the performance of our heuristic approach by benchmarking it against both the optimal ILP solution and a greedy method that processes the commodities one by one. We demonstrated numerically that our heuristic algorithm strikes a good balance among optimality gap, VNF cost and execution time.

In terms of future development, we plan to refine our heuristic approach by boosting its performance in the presence of stressed node resources. Moreover, we envision adapting our heuristic to be able to better accommodate in an online fashion the arrival of new commodities, while still outperforming the greedy approach.

REFERENCES

[1] Network Functions Virtualisation (NFV), Service quality metrics, ETSI GS NFV-INF 010, V1.1.1, December 2014.

[2] B. Addis, D. Belabed, M. Bouet, and S. Secci, Virtual network functions placement and routing optimization, 2015 IEEE 4th Int Conf Cloud Netw (CloudNet), Niagara Falls, ON, 2015, pp. 171–177.

[3] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba, Virtual network embedding with coordinated node and link mapping, IEEE INFOCOM 2009, Rio de Janeiro, 2009, pp. 783–791.

[4] R. Cohen, L. Lewin-Eytan, J.S. Naor, and D. Raz, Near optimal placement of virtual network functions, 2015 IEEE Conf Comput Commun (INFOCOM), Kowloon, 2015, pp. 1346–1354.

[5] G. Even, M. Rost, and S. Schmid, "An approximation algorithm for path computation and function placement in SDNs," Structural Information and Communication Complexity: 23rd Int Colloq, SIROCCO 2016, Helsinki, Finland, July 19–21, 2016, Revised Selected Papers, J. Suomela (Editor), Springer International Publishing, Cham, Switzerland, 2016, pp. 374–390.

[6] A. Fischer, J.F. Botero, M.T. Beck, H. de Meer, and X. Hesselbach, Virtual network embedding: A survey, IEEE Commun Surv Tutor 15 (2013), 1888–1906.

[7] C. Fuerst, S. Schmid, and A. Feldmann, Virtual network embedding with collocation: Benefits and limitations of preclustering, 2013 IEEE 2nd Int Conf Cloud Netw (CloudNet), San Francisco, CA, 2013, pp. 91–98.

[8] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba, Service function chaining simplified, CoRR, 2016, abs/1601.00751.

[9] Y. Li and M. Chen, Software-defined network function virtualization: A survey, IEEE Access, (2015), 2542–2553.

[10] 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, 2015 IFIP/IEEE Int Symp Integr Netw Manage (IM), Ottawa, ON, 2015, pp. 98–106.

[11] T. Lukovszki, M. Rost, and S. Schmid, It's a match!: Near-optimal and incremental middlebox deployment, ACM SIGCOMM Comput Commun Rev 46 (2016), 30–36.

[12] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," Structural Information and Communication Complexity: 22nd Int Colloq, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings, C. Scheideler (Editor), Springer International Publishing, Cham, 2015, pp. 104–118.

[13] S. Mehraghdam, M. Keller, and H. Karl, Specifying and placing chains of virtual network functions, 2014 IEEE 3rd Int Conf Cloud Netw (CloudNet), Luxembourg, 2014, pp. 7–13.

[14] R. Mijumbi, J. Serrat, J.L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, Network function virtualization: State-of-the-art and research challenges, IEEE Commun Surv Tutor 18 (2016), 236–262.

[15] H. Moens and F.D. Turck, VNF-P: A model for efficient placement of virtualized network functions, 10th International Conference on Network and Service Management (CNSM) and Workshop, Rio de Janeiro, 2014, pp. 418–423.