Virtual Network Functions Placement and Routing Problem: *Path formulation*

Ahlam MOUACI
Orange Labs
Université Paris Dauphine- Lamsade
Paris, France
ahlam.mouaci@orange.com

Éric GOURDIN
Orange Labs
Paris, France
eric.gourdin@orange.com

Ivana LJUBIĆ
ESSEC Business School
Cergy, France
ivana.ljubic@essec.edu

Nancy PERROT
Orange Labs
Paris, France
nancy.perrot@orange.com

***Abstract*—Network Functions Virtualization (NFV) and *Software Defined Networking (SDN)* are two promising techniques for the next generation telecommunication networks. Their introduction allows time, energy and cost minimization. Placing *Virtual Network Functions (VNFs)* on network nodes and routing data through these nodes is a very challenging combinatorial optimization problem. Obviously, the problem becomes even more difficult, if in addition, we have to route the data using the concept of *Service Functions Chaining (SFC)* in which VNFs are chained according to a pre-defined order associated to each service.**

In this paper we study the Virtual Network Functions Placement and Routing problem in Software Defined Networks, in which a set of source-destination pairs representing clients demand and a set of VNFs are given. The problem consists in finding a routing path for each demand and the optimal associated placement of functions while minimizing functions installation and node activation costs. In this work, we propose a path-based MILP formulation to model the problem and we also demonstrate how to efficiently use it to derive high-quality heuristic solutions within a short computational time. We provide a case study derived from a set of scenarios in which we vary relevant problem parameters, including arc latency, input demand and node capacities. The study is conducted on a benchmark set of realistic telecommunication instances from the SNDlib library. To test the efficiency of our approach, we also compare the obtained results with a compact MILP formulation. Our computational study indicates that the path-based formulation outperforms the compact model both in terms of computing time and overall solution quality.

***Index Terms*—Virtual Network Functions, Software Defined Networking, Service Functions, Service functions Chaining, Heuristic, Combinatorial optimization.**

I. Introduction

In order to provide services such as: online gaming, video streaming or web services, Network Service Providers have to deal with an optimization problem consisting in routing the network traffic in a cost-effective way. Each service requires packet transfer between two nodes in the network, and has to be handled by a specific set of *Service Functions (SFs)*. Traditionally, Service Functions such as Firewall, Video optimizer, or Traffic sensors, were running on a specific hardware. Their orchestration, connection, deployment and reconfiguration are costly, very time consuming and generate long installation delays (caused by several manual interventions).

Network Function Virtualization (NFV) is a new paradigm that aims to reduce costs and accelerate the deployment of SFs for network operators by dissociating features such as firewall or encryption, from any dedicated hardware and moving them to virtual servers. That way, SFs are executed as network applications only when and where they are required. Homogeneous and heterogeneous SFs can be hosted by the same hardware server. The SFs are then called *Virtual Network Functions (VNFs)*.

Software Defined Networking (SDN) is a new architecture that aims to simplify the telecommunication networks management by separating the control plane from the data plane. There exists a single centralized controller that has a global view of the network state and takes the decisions regarding the package forwarding. Each service has to be handled by a specific set of VNFs in a pre-defined order. This order between VNFs is called *Service Functions Chain (SFC)*, and the mechanism that allows orchestration of SFCs is called *Service Functions Chaining*. Using SDN and NFV, VNFs can be installed anywhere and any-when on network nodes and packets can be routed through these nodes dynamically. This new flexibility and dynamism faces new challenges and creates new combinatorial optimization problems that are very difficult to model and to solve.

The problem we consider in this paper is the Virtual Network Functions Placement and Routing Problem (VNFP-RP), which can be summarized as follows: we are given a set of client demands (also called commodities), each of which is characterized by a source and a destination node and a set of VNFs with a specific total order. The problem consists in finding *(i)* the optimal VNFs placement on nodes and *(ii)* the associated feasible latency-constrained routing paths passing through VNFs in the right order. In addition, node capacities, function capacities, along with incompatibility rules (between pairs of functions) must be satisfied. In [1], we proposed a compact MILP formulation for the problem variant without node and function capacity constraints. We showed that in this case, integrality constraints on some variables can be relaxed, so that the problem can be solved using a separable Benders decomposition approach. Unfortunately, once the capacity constraints are added to the model, these theoretical findings are no longer valid. Consequently, Benders decomposition cannot be applied to the problem setting studied in this paper.

Our contribution. A compact MILP formulation to model the problem has been proposed in a recent literature. Unfortunately, due to the large number of variables and constraints, it has been shown that the computational performance of this formulation is fairly limited. Some papers divide the problem into two parts, and treat the VNFs placement and the routing problems separately. A large body of literature deals with heuristics for each part of the problem, or for their relaxed versions. In this paper we propose a computationally effective path-based MILP formulation to model the Virtual Network Functions Placement and Routing Problem. The Path Formulation (PF) uses Yen's algorithm to find the elementary latency-constrained shortest paths associated to each commodity. If all feasible paths are taken into the model, the PF model provides an exact problem formulation and can be used to calculate the optimal solution. Alternatively, if only a subset of feasible paths is included in the model, the result of the PF provides a high quality heuristic solution. We demonstrate the effectiveness of the PF (when compared to the compact MILP formulation) on a set of realistic benchmark instances derived from the SNDLib library. In addition, we analyze the effects of varying relevant problem parameters (like arc latency, input demand and node capacities) on the solution cost.

Outline of the paper. Section II contains a review of related literature. In section III we provide a formal problem definition and introduce the notation. Section IV is devoted to the Path formulation. In section V we present some computational results and the sensitivity analysis for which we vary the input parameters and discuss the results. In sections VI some observations and concluding remarks are provided.

II. Related works

Some articles were proposed in the literature to define how and where to place ordered chains of VNFs. According to [2], these studies can be divided into two categories: ILP-based and greedy-heuristic-based. Different objective functions were considered as well, such as: minimizing the number of activated nodes, minimizing the end-to-end latency [3], minimizing the energy consumption [4], or the bandwidth [5].

Authors in [6] propose an ILP formulation and a heuristic for the problem of placing chained VNFs on a physical infrastructure while minimizing the number of VNFs. In [7], the authors consider both VNFs placement and path selection with precedence constraints, while maximizing the number of treated demands. The authors in [3] model the VNFs placement problem as a Mixed Integer Quadratically Constrained Program with two possible different objectives: minimizing the number of activated nodes or minimizing the paths latency. Their model was tested on small instances and solved using Gurobi Optimizer. Addis et al. in [8] model the VNFP-RP as a MILP considering flow constraints, paths latency constraints and resource capacities, while minimizing the number of used nodes and links. Also, authors in [8] propose a math-heuristic approach for the problem. Mijumbi et al. in [9] consider the problem of online mapping and scheduling VNFs without any routing aspect.

Cohen et al. in [10] provide a near-optimal approximation algorithm to address the problem of placing VNFs on physical network without order constraints. Their objective is the minimization of the installation and paths costs. In order to minimize the number of used network functions, Sang et al. in [11] fix the routing paths and focus their attention on the placement of VNFs without any order constraints. They leave the chaining constraints for their future works. Addis et al. in [12] address the Virtual Network Function Placement and Routing problem. They suppose that all demands require the same service and each demand can be routed only through a simple path. To deal with the problem they propose two formulations whose goal is to minimize the number of VNFs installed. Tomassilli [2] propose two approximation algorithms for the tree network topology in order to address the SFC placement problem.

The closest work to ours is presented by Allybokus et al. in [13]. The authors propose a heuristic and different MILP formulations to model the VNF placement and routing problem by considering partial order, end-to-end latency, anti-affinity rules, resource limitations and capacity constraints. They deal with two goals: minimizing the total network deployment cost, or minimizing the total number of rejected demands. The proposed heuristic is based on the continuous relaxation of one of the MILP formulations. The tests show that the heuristic minimising the number of rejected demands is very efficient on instances generated with GEANT topology. However, the proposed MILP formulations can not solve to optimality the realistic instances within reasonable time.

III. Problem statement

We consider a bi-directed graph $= (N, A)$ representing the telecommunication network: N denotes the set of nodes representing the physical locations that support VNF installation. Each node u in N has an installation capacity c_u. A represents the set of arcs between nodes. The graph being bi-directed, we assume that, for each arc (u, v) in A, the reverse arc (v, u) also exists. We denote by l_{uv} the latency of arc (u, v), a quantity representing the time needed to go from node u to node v. The set of all VNFs is denoted by F. A capacity m_f, representing the maximum amount of data that can be treated, is associated with each function $f \in F$. The cost of installing a function f on node u is denoted by ψ_u^f. We denote by C the set of all traffic demands or commodities: each demand $k \in C$ is defined by, a source node $s_k \in N$, a destination node $d_k \in N$, a maximum latency denoted $l_k \in \mathbb{R}_+$ and a set of incompatibilities between functions. Moreover, the order in which the functions of each demand must be visited is represented by the ordered set $F^k \subseteq F$: we denote by $f \prec_k g$ the fact that function f must be visited before function g for demand k. We assume that if f and g are installed on the same node u, the precedence constraints $f \prec_k g$ are trivially satisfied. All these notations are summarized in Table (I).

sets		
N	:	Set of nodes.
A	:	Set of arcs.
C	:	Set of commodities (demands).
F	:	Set of functions or VNFs (Virtual Network Functions).
$(F^k, \prec_k)$	:	Ordered set of VNFs associated to commodity k.
S^k	:	Set of pairwise conflicts between functions associated to commodity k.
Parameters		
b_k	:	Amount of traffic for commodity $k \in C$.
s_k	:	Source node for commodity $k \in C$.
d_k	:	Destination node for commodity $k \in C$.
l_k	:	Maximum latency of commodity $k \in C$.
l_{uv}	:	Latency of link $(u,v) \in A$.
ψ_u^f	:	Installation cost of function f at node $u \in N$.
ψ_u	:	Activation cost of node $u \in N$.

TABLE I: Main notations

Definition 1: The Virtual Network Functions Placement and Routing Problem consists in finding the optimal placement of VNFs $f \in F$ on nodes $u \in N$, so that, for each commodity $k \in C$ the function are encountered, along the path used to carry the whole commodity traffic, in the order specified by F^k, and such that the following constraints are satisfied:

- **Routing constraints:** for each commodity $k \in C$, the chosen $s_k - d_k$ path is elementary (no cycles allowed) and satisfies the latency constraint (the sum of latency of the path arcs must be less than or equal to l_k).
- **Node capacity constraints:** the number of VNFs installed at the node $u \in N$ must be less or equal to its capacity c_u.
- **Functions capacity constraints:** the volume of data treated by each installed VNF $f \in F$ must be less or equal to its capacity m_f.
- **Precedence constraints:** The functions associated with each commodity $k \in C$ must be encountered, when traveling along the chosen path, in the order specified by F^k. We assume that no function can be installed on the source node for any commodity $k \in C$.
- **Conflict (Anti-affinity) constraints:** for each commodity $k \in C$, two functions f and $g \in F^k$ in conflict cannot be installed at the same node.

As usual, a solution of the problem is called feasible if it satisfies all the required constraints, and it is called optimal if the associated cost is minimized. The cost of a solution is defined as the sum of nodes activation cost ψ_u, $\forall u \in A$ plus the functions installation costs ψ_u^f, $\forall f \in F$ and $\forall u \in N$.

Theorem 3.1: The Virtual Network Functions Placement and Routing Problem (VNFP-RP) is strongly *NP-Hard* even for a single commodity and without latency and precedence constraints [1].

IV. Path Formulation

In this Section, we provide a path-based MILP formulation for our problem. We first describe the variables, and then the constraints defining the model.

1) Variables: Our path formulation is based on the five sets of variables described in Table II. We denote by $\mathcal{P}_k$ the set of all elementary $s_k - d_k$ paths (associated with the commodity k). We also assume $\mathcal{P}_k$ contains only paths that satisfy the latency constraint. For each commodity k, we suppose that all feasible paths $\mathcal{P}_k$ are given (which might be a problem in practice, when the instances become large). Let t_{uv}^{pk} be the parameter that is equal to 1 if the arc (u,v) is in the path p for the commodity k, 0 otherwise.

Variables		Type
λ_p^k	1, if path p is chosen for commodity k; 0, otherwise.	Binary
x_u^{fk}	1, if virtual network function f is installed at or before node u for commodity k; 0, otherwise.	Binary
y_u^{fk}	1, if virtual network function f is installed at node u for commodity k; 0, otherwise.	Binary
w_u	1, if node u is activated; 0, otherwise.	Binary
z_u^f	the number of service functions f installed at node u.	Integer

TABLE II: Decision variables of the path-based formulation

A. Constraints

1) Routing constraints:

$$\sum_{p \in \mathcal{P}_k} \lambda_p^k = 1, \qquad \forall k \in C \tag{1}$$

Each commodity k must be routed on exactly one $s_k - d_k$ path $p \in \mathcal{P}_k$ satisfying the latency constraint.

2) Installation constraints:

$$\sum_{u \in N} y_u^{fk} \geq 1, \qquad \forall k \in C, \quad \forall f \in F^k \tag{2}$$

Constraints (2) ensure that all virtual network functions associated to a given commodity k are indeed installed on some nodes.

3) Capacity constraints:

$$\sum_{f \in F} z_u^f \leq c_u\, w_u, \qquad \forall u \in N \tag{3}$$

$$\sum_{k \in C} y_u^{fk} b_k \leq m_f z_u^f, \qquad \forall f \in F, \quad \forall u \in N \tag{4}$$

The node capacity constraints (3) guarantee that the number of functions installed on any activated node u does not exceed its capacity c_u. The function capacity constraints (4) ensure that the volume of all data treated by any VNF f does not exceed its capacity m_f.

4) Incompatibility constraints:

$$y_u^{fk} + y_u^{gk} \leq 1 \qquad \forall k \in C, \quad \forall (f,g) \in S^k, \quad \forall u \in N \tag{5}$$

Constraints (5) guarantee that two functions f and g in conflict can not be installed on the same node u.

5) Precedence constraints:

$$\Big(\sum_{\substack{p\in\mathcal{P}_k\\(u,v)\in p}} t_{uv}^{pk}\ \ \lambda_p^k - 1\Big) + (x_v^{fk} - x_u^{fk}) \quad \le y_v^{fk}, \qquad \forall k\in C,\quad \forall f\in F^k,\quad \forall (u,v)\in A \tag{6}$$

$$(x_u^{fk} - x_u^{gk})p_k(f,g) \quad \le \quad 0, \qquad \forall k\in C, \qquad \forall f,g\in F^k,\quad \forall u\in N \tag{7}$$

$$y_u^{fk} \quad \le \quad x_u^{fk}, \quad \forall k\in C,\quad \forall f\in F^k,\quad \forall u\in N \tag{8}$$

$$x_{s_k}^{fk} \quad = \quad 0, \qquad \forall k\in C,\quad \forall f\in F^k \tag{9}$$

$$x_{d_k}^{fk} \quad = \quad 1, \qquad \forall k\in C,\quad \forall f\in F^k \tag{10}$$

Constraints (6) represent a linking between routing variables, installation variables and precedence variables, ensuring that, for each commodity k: *(i)* if the routing path p passes through an arc (u,v), and *(ii)* the VNF f is installed at or before the node v and *(iii)* the VNF f is not installed at or before the node u, then the left-hand-side is forced to be equal to 1 (imposing the installation of the function f at node v).

The second set of constraints uses the sorting parameter $p_k(f,g)$ that is equal to 1 if VNF f must be installed before the VNF g, equal to -1 if f must be installed after g, and 0 otherwise. Constraints (7) is hence modelling the required precedence between functions.

Constraints (8) force the precedence variables x to be 1 if the associated installation variables y is equal to 1. In other words, if VNF f is installed on the node u, then it is necessary also installed at or before u.

6) Linking constraints:

$$y_u^{fk} \quad \le \quad \sum_{(v,u)\in A}\sum_{p\in\mathcal{P}_k} t_{vu}^{pk}\ \ \lambda_p^k, \quad \forall k\in C, \qquad \forall f\in F^k,\quad \forall u\in N \tag{11}$$

$$y_u^{fk} \quad \le \quad w_u, \quad \forall k\in C,\quad \forall f\in F^k,\quad \forall u\in N \tag{12}$$

Constraints (11) impose that the chosen path for each commodity k passes through the nodes where the required functions are installed. Constraints (12) link the variables y and w and ensure that if function f is installed at the node u then u is activated.

B. Objective function

$$\min \quad \sum_{u\in N}\sum_{f\in F} \psi_u^f \ \ z_u^f + \sum_{u\in N}\psi_u \ \ w_u$$

The objective function to be minimized is the sum nodes activation and functions installations costs.

C. Model

The Path-based formulation then reads as follows:

$$\min \quad \sum_{u\in N}\sum_{f\in F} \psi_u^f \ \ z_u^f + \sum_{u\in N}\psi_u \ \ w_u$$

$$(\lambda, x, y, z, w) \qquad \text{satisfy} \quad (1) - (12)$$

$$\begin{array}{rr} \lambda_p^k \in \{0,1\} & \forall k\in C,\quad \forall p\in\mathcal{P}_k \\ x_u^{fk}, y_u^{fk} \in \{0,1\} & \forall k\in C,\quad \forall u\in N,\quad \forall f\in F \\ w_u \in \{0,1\} & \forall u\in N \\ z_u^f \in \mathbb{N} & \forall u\in N,\quad \forall f\in F \end{array}$$

D. Getting routing paths

In order to generate routing paths associated with each commodity k we use Yen's algorithm [14]. This algorithm aims at finding a bounded number of loopless shortest paths between a pair of nodes. Yen's algorithm is based on Dijkstra's algorithm [14]. In reasonnably small instances, instead of recovering a fixed number of paths per commodity, we can run Yen's algorithm to get all $s_k - d_k$ paths.

In order to take into account the latency constraints (13) in the PF, we stop enumerating paths in Yen's algorithm when they do not satisfy the latency constraint (*i.e.,* we keep only paths whose length is less or equal to l_k, $\forall k\in C$).

$$\sum_{(u,v)\in p} (l_{uv}\ t_{uv}^{pk})\lambda_p^k \quad \le \quad l_k, \qquad \forall k\in C,\quad \forall p\in\mathcal{P}_k \tag{13}$$

Theorem 4.1: If Yen's algorithm generates all elementary latency-constrained paths for each commodity $k\in C$, then the path-based formulation gives an optimal solution. Otherwise it provides a heuristic solution for VNFP-RP.

V. Numerical results

The purpose of this section is to test the efficiency and the sensitivity of the proposed path formulation. We present some computational results to compare the PF formulation with the compact MILP formulation proposed in [1] (which we extended with capacity and incompatibility constraints in this paper). In this comparison of the two models, we focus on the CPU time, the quality of obtained solutions and the final gaps between the global lower and upper bounds. We vary the input parameters and conduct a sensitivity analysis to determine the most relevant parameters that affect the empirical performance of the model.

All tests were made using the commercial solver CPLEX and a machine with Intel(R) Xeon(R) CPU E5-2650 v2 processor clocked at 2.60GHz and 252GB RAM, under Linux operating system. All methods are implemented using the Python API for CPLEX, which is run in single-threaded mode and all CPLEX parameters were set to their default values. A default time limit of one hour is set for each tested instance.

A. Benchmark instances

Our benchmark instances are generated using the SNDlib library [15] (Survivable fixed telecommunication Network Design) which is a repository of realistic telecommunication network design instances. The number of nodes, arcs and demands vary for each instance. SNDlib provides the graph topology, along with the node coordinates and a set of communication demands with the associated source node, destination node and a bandwidth. The remaining parameters required for our setting are generated as follows:

To calculate the distance between two nodes u and v in km having only their longitude φ_u, φ_v and latitude ς_u, ς_v, we use the Spherical Law of Cosine [16]. First we use the following formula to calculate the angular distance in radians:

$$S_{uv} = arc\cos(\cos\varsigma_u \cos\varsigma_v + \sin\varsigma_u \sin\varsigma_v \cos d\varphi)$$

with, $d\varphi = \varphi_v - \varphi_u$.

The distance in km is then obtained as:

$$d_{uv} = R \times S_{uv},$$

where R is earth's radius ($R = 6378137km$). The fiber propagation delay per km is roughly equal to $10\mu s/km$, see, e.g., [17], [18]. To define the latency l_{uv} of an arc (u, v), we multiply the distance between u and v by the fiber propagation delay, and we set $l_{uv} = l_{vu}$.

We consider the following six Virtual Network Functions, typically employed in service function chaining ([5], [19]) to construct our VNFs set, namely $F = \{$*NAT, FW, TM, WOC, IDPS, VOC*$\}$. A detailed description of these VNFs is given in Table III. The maximum capacity rate associated to each VNF is given as an integer number in $[a, b]$, where *a (resp. b)* is the minimum *(resp. maximum)* bandwidth of all demands in the same set of demand (*i.e.,* $a = \min_{k \in C} b_k$ and $b = \max_{k \in C} b_k$), the capacity is measured on $Mbits/s$. The functions installation cost $\psi_u^f \in \mathbb{N}$ is chosen randomly as integer in $[50, 1000]$, and is expressed in dollars \$.

id-VNF	VNF name	Capacity	Cost
NAT	Network Address Translator	$[\min_{k \in C} b_k, \max_{k \in C} b_k]$	$[50, 1000]$
FW	Firewall		
TM	Traffic Monitor		
WOC	WAN Optimization Controller		
IDPS	Intrusion Detection Prevention System		
VOC	Video Optimization Controller		

TABLE III: Virtual Network Functions

The total number of demands varies for each SNDlib instance. For each commodity $k \in C$ the source node s_k, destination node d_k and the bandwidth b_k in $Mbits/s$ are given.

We divide the set of demands in five categories: *Online Gaming*, *Video Streaming*, *Voice over IP*, *Web Services*, and *Other Services*. Each category is characterized by a latency value and a set of chained service functions as depicted in Table IV.

To define the value of l_k and the set of chained VNFs associated to each demand k, the demands are first assigned to exactly one category. To do so we calculate the shortest path between s_k and d_k which is equal to the sum of the arcs latency composing it. Based on its length, we assign randomly our demands in one of the five categories, and so we set the value of l_k and the set of chained VNFs. For example, if the length of the shortest path is equal to $l(SP_k) = 40ms$ then we can assign the demand to any of the five categories, and we randomly choose one.

Latency value	Service chain	Chained VNFs
$\leq 60\ ms$	Online Gaming (O-G)	NAT-FW-TM-WOC-IDPS
$\leq 100\ ms$	Video Streaming (V-S)	NAT-FW-TM-VOC-IDPS
	VoIP	NAT-FW-TM-FW-NAT
$\leq 500\ ms$	Web Services (W-S)	NAT-FW-TM-WOC-IDPS
$\leq \infty\ ms$	Other services (O-S)	NAT-FW-TM-WOC-VOC

TABLE IV: Service functions chains

For each commodity $k \in C$ we have at most one anti-affinity constraints (AAC) between VNFs. We suppose that we cannot install Firewall and Network Address Translator at the same node. The maximum number of AAC is fixed to five per instance, because adding too many of them can make some instances infeasible.

In order to get feasible instances, the node capacities are based on the number of demands and the number of VNFs per commodity which is equal to 5, $\forall k \in C$. Thus, the node capacity is an integer chosen randomly $c_u \in [\frac{|C| \times 5}{|N|}, 2 \times \frac{|C| \times 5}{|N|}]$, $\forall u \in N$. Node activation cost $\psi_u \in \mathbb{N}$ is chosen uniformly at random from $[3000, 5000]$ [20].

Instance_type	n	m / bi-directed	$\lvert C \rvert$	$\lvert F \rvert$	# AAC
Abilene	12	15/ 30	132	6	5
Atlanta	15	22/ 44	210	6	4
Dfn-bwin	10	45/ 90	90	6	0
Dfn-gwin	11	47/ 94	110	6	0
Di-yuan	11	42/ 84	22	6	0
France	25	45 / 90	300	6	5
Geant	22	36/ 72	462	6	1
Janos-us	26	84/ 168	650	6	1
Newyork	16	49/ 98	240	6	0
Nobel-eu	28	41/ 82	378	6	0
Nobel-Germany	17	26/ 52	121	6	5
Nobel-us	14	21/ 42	91	6	0
Pdh	11	34/ 68	24	6	0
Polska	12	18/ 36	66	6	1
Ta1	24	55/ 110	396	6	0

TABLE V: Details about the instances. *n: the number of nodes, m: the number of arcs, $|C|$: the number of demands, $|F|$: the number of functions, # AAC: the number of anti affinity constraints.*

To test the efficiency of the proposed Path formulation we solve fifteen different instances from SNDlib library. Table V summaries the details about the instances used in this study. In order to vary the demand types per instance we generated ten instances for each instance type. Table VI summaries the average demand per service function chain and per instance type.

Instance_type	O-G	V-S	VOIP	W-S	O-S
Abilene	4.2	6.2	7.4	57.6	55.6
Atlanta	1.7	6.1	4.9	98.7	97.6
Dfn-bwin	0.0	22.1	21.4	21.9	23.6
Dfn-gwin	0.0	23.6	23.5	31.0	30.9
Di-yuan	0.6	1.6	1.2	8.1	9.5
France	2.5	11.0	8.8	140.1	136.6
Geant	0.8	20.9	17.5	208.1	213.7
Janos-us	10.5	24.2	22.5	302.9	288.9
Newyork	4.5	12.8	11.3	105.6	104.8
Nobel-eu	0.3	10.8	9.4	168.9	187.6
Nobel-Germany	0.0	6.6	5.7	52.0	55.7
Nobel-us	1.6	5.3	6.2	39.0	37.9
Pdh	0.0	6.3	4.9	5.5	6.3
Polska	0.0	5.0	4.1	27.5	28.4
Ta1	6.5	20.5	20.1	169.2	178.7

TABLE VI: Average demands per service function chain and per instance type.

B. Models analysis

In this subsection we analyse the proposed path-based formulation in terms of the number of variables and constraints of the model. We also compare them with the number of variables and constraints in the compact MILP formulation from [1] (extended with capacity and incompatibility constraints). In the following we put only the difference in terms of the number of variables and constraints between the two models.

Variables: Both formulations have practically the same set of variables, except for the routing part. For the PF formulation we have $|C| \times |\mathcal{P}_k|$ path variables, for the MILP compact formulation we have $|C| \times |A|$ arc variables.

Constraints: As we are generating only latency-constrained paths for the Path formulation we are gaining in terms of the number of constraints. Because of that, the compact MILP formulation has $|C| \times |N|$ more constraints.

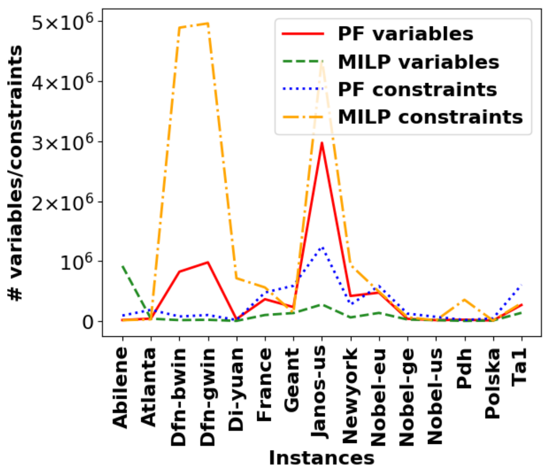

Fig. 1: The number of variables and constraints for the two formulations tested in this paper.

Figure 1 shows that the number of variables in the PF formulation is on average greater than the number of variables of the compact MILP formulation. This is due to the number of feasible paths associated to each commodity. We can also observe that the number of constraints in the compact MILP formulation is on average greater than the number of constraints in the PF formulation – this number depends on the number of nodes and demands.

C. Obtained results

The following two settings are compared in our computational study:

- MILP : The compact MILP formulation proposed in [1], which we have extended by the capacity and incompatibility constraints (3)-(5).
- PF : The path-based formulation proposed in Section IV.

In the computational results presented in Table VI we solved 10 instances of each instance type. The CPU time, GAP, and the costs of each instance type are obtained by calculating the average over all 10 instances for which a feasible solution was found. We fixed the maximum number of latency-constrained paths generated by Yen's algorithm to 5000 paths.

In order to define the nature of the PF formulation, for each instance type, we pre-calculate the maximum number of latency constrained paths over all commodities. This number is shown in column #paths in Table VII. Based on that, the second column denoted by PF(E/H), describes the nature of the PF formulation: exact "E" means that the maximum number of feasible paths was below our threshold of 5000 paths per commodity; or heuristic "H" method, meaning that not all feasible paths were taken into consideration by the PF formulation. Path formulation is a heuristic for only two instance types: "Dfn-bwin" and "Dfn-gwin".

In Table VII we also report the number of instances solved to optimality by the two models. The PF formulation solves 8 instances to optimality within one hour, whereas MILP manages to find the optimum in only 4 cases. Moreover, there are five instances for which the compact MILP formulation cannot provide any feasible solution within the given timelimit. On the contrary, as demonstrated below, high quality solutions are found for all instances, using the PF formulation.

Instance_type	# paths	PF(E/H)	# optimal		no feasible solution found	
			PF	MILP	PF	MILP
Abilene	16	E	6	2	0	0
Atlanta	53	E	0	0	0	0
Dfn-bwin	9061	H	0	0	0	0
Dfn-gwin	8801	H	0	0	0	0
Di-yuan	1419	E	0	0	0	0
France	969	E	0	0	0	0
Geant	286	E	0	0	0	0
Janos-us	4316	E	0	0	0	0
Newyork	1588	E	0	0	0	0
Nobel-eu	977	E	0	0	0	4
Nobel-Germany	149	E	0	0	0	0
Nobel-us	45	E	0	0	0	0
Pdh	872	E	2	2	0	0
Polska	35	E	0	0	0	0
Ta1	440	E	0	0	0	1

TABLE VII: Nature of the Path formulation (exact or heuristic) and a comparison between the two formulations with respect to the number of instances solved to optimality (#optimal) and the number of instances for which no feasible solution was found.

Table VIII summarizes the results obtained by solving 15 SNDlib instance types with different graph topologies and different number of commodities. Columns one and two in the table represent the average CPU time calculated from instances solved to optimality for PF and MILP. We observe that when we have instances solved to optimality the PF is outperforming in terms of CPU time and also that for most of the instances both methods cannot provide an optimal solution within one hour. TL in the CPU time column means that the time limit was reached. When the optimal solution is not found after reaching the time limit, we also report the final gap in percent calculated as $GAP(\%) = (UB - LB)/LB * 100\%$, where UB represents the best feasible solution found and the LB represent the global lower bound found. The global upper bound is shown in column Costs($), and the value of the LP-relaxation of the two models is given in the last two columns.

For all instances for which PF is an exact method the final gap provided (CPLEX exit gap) by PF is consistently better than the final gap provided by MILP (see Figure 2). Each coordinate (x, y) in Figure 3 indicates that for y instance types, the average final gap provided was bellow x. From Figure 3 we can observe that all instance types solved by PF as an exact method are solved within a gap less than 55%, while the same instance types are solved within a gap of 75% by the MILP compact formulation.

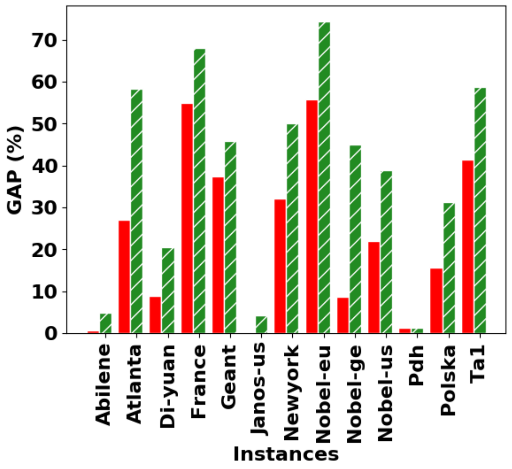

Fig. 2: GAP comparison between PF and MILP

Similarly, from Figure 4 we observe that PF consistently provides solutions of a significantly higher quality when compared with solutions given by the MILP compact formulation. The difference between the two methods in terms of costs is very visible.

Figure 5 shows the relative improvement of the LP-relaxation value provided by PF with respect to MILP. We notice that the PF formulation provides a much better relaxation bound compared with the compact MILP formulation. In Figure 5 we observe that for all solved instances the value of the relaxation at the root node provided by the PF formulation is always greater than or equal to the given LP-relaxation of the compact MILP formulation, so closer to the optimal solution.

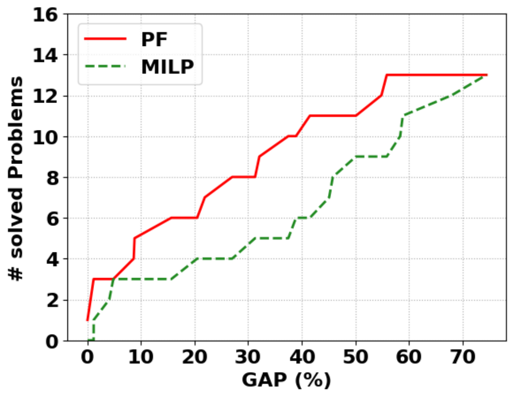

Fig. 3: GAP comparison between PF and MILP

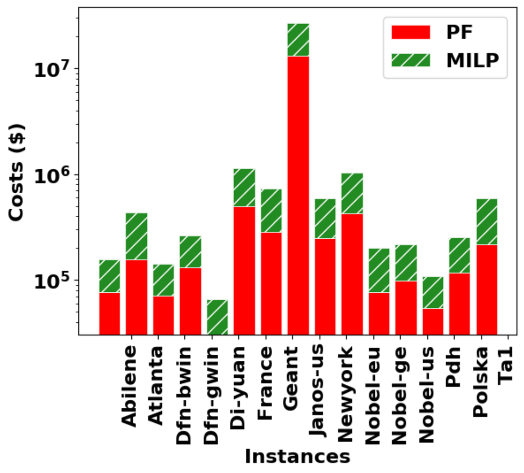

Fig. 4: Costs comparison between PF and MILP

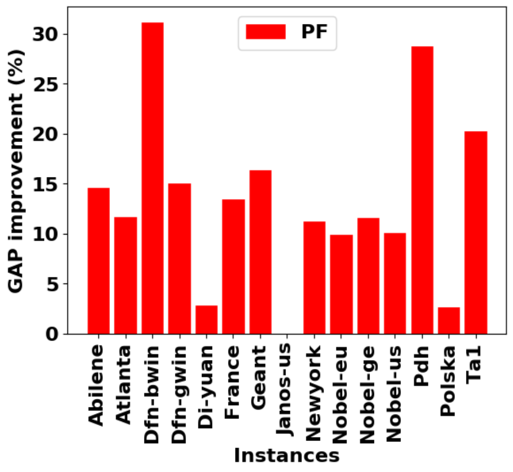

Fig. 5: LP-relaxation improvement by the Path formulation.

1) Sensitivity analysis: For the sensitivity analysis we vary the instances parameters such as: latency value l_k, bandwidth b_k, node capacities c_u and functions capacities m_f and observe

Instance_name	CPU_time(s)		GAP(%)		Costs($)		Relaxation value	
	PF	MILP	PF	MILP	PF	MILP	PF	MILP
Abilene	**1123.64**	1688.92	**0.61**	4.85	**77158.50**	79990.30	**67547.38**	58929.67
Atlanta	TL	TL	**27.03**	58.37	**156865.60**	278836.30	**103478.55**	92640.77
Dfn-bwin	TL	TL	–	**1.58**	70717.00	**70656.50**	**63927.16**	48731.37
Dfn-gwin	TL	TL	–	8.79	**131284.00**	131783.70	**116131.98**	100959.17
Di-yuan	TL	TL	**8.81**	20.49	**30759.70**	35104.00	**26154.09**	25435.27
France	TL	TL	**54.87**	67.97	**497767.20**	651469.10	**213059.66**	187780.30
Geant	TL	TL	**37.50**	45.80	**284847.80**	451038.30	**127580.37**	109636.63
Janos-us	TL	TL	**0.04**	4.10	**13221546.20**	13775446.70	13204901.84	13204901.84
Newyork	TL	TL	**32.10**	50.11	**248954.50**	342506.20	**157088.98**	141291.17
Nobel-eu	TL	TL	**55.86**	74.45	**430106.33**	600379.83	**152708.92**	139001.24
Nobel-germany	TL	TL	**8.66**	45.11	**77152.50**	124062.90	**65700.13**	58908.65
Nobel-us	TL	TL	**21.91**	38.93	**97874.20**	118642.50	**70360.64**	63906.55
Pdh	**1279.91**	1587.34	**1.17**	1.20	**54508.10**	54561.60	**50051.47**	38874.36
Polska	TL	TL	**15.67**	31.30	**116640.40**	137285.30	**90635.47**	88343.00
Ta1	TL	TL	**41.52**	58.86	**217968.22**	376160.33	**94323.52**	78467.77

TABLE VIII: CPU time, Gap, Cost and relaxation value comparison between PF and MILP.

how the algorithm will behave, by putting $l_k = \alpha \times l_k$, $b_k = \alpha \times b_k$, $c_u = \alpha \times c_u$ and $m_f = \alpha \times m_f$ with $\alpha \in \{1.0, 1.5\}$. The aim of these tests is to show the parameters that affect directly the solution, and so the costs.

Figures 6 and 7 show the obtained results by varying the number of paths generated by Yen algorithm, with $max\_paths \in \{5000, 500, 100, 50\}$ and $\alpha = 1.0$. Fixing the number of paths for Yen's algorithm means that we are tightening the demands latency. We compare the values of the best found solutions within one hour (averaged over 10 graphs per instance type) for each of these settings.

In Figure 6 we have instances that need at most 500 paths to be solved to optimality. For these instances we varied the number of max_paths in $\{50, 100, 500\}$. We observe that adding a small number of paths permits PF to converge quickly to a very good solution for "Geant" and "Ta1" for which the number of required paths exceeds 200 paths. Conversely, for instances "Atlanta" and "Nobel-germany" which required only 53 and 149 paths, respectively, to be solved to optimality, 50 paths were insufficient to find a high quality solution. This can be explained by the sparsity of their graphs and the number of demands.

In Figure 7 we have instances that require more than 500 paths to be solved to optimality. We notice that for instance "Janos-us", the PF formulation was struggling to find a good quality solution. It should be mentioned that CPLEX does not manage to solve models with a large number of variables very well. For all other instances we can see that the number of fixed paths does not affect a lot the behaviour of PF.

Figure 8 shows that increasing the function capacities allows for a significant cost reduction for all instances. A similar effect can be achieved by increasing node capacities.

From Figure 9 we notice that increasing bandwidth of demands increases costs for all solved instances. On the other hand, increasing latency for "Dfn-bwin" and "Pdh" allows cost reduction, both instances have almost the same sparse graph topology. Increasing the bandwidth value for some instances like "Polska" and "Dfn-gwin" makes them infeasible. Conversely increasing the latency value makes the problem easy to solve, so we can have an optimal solution very quickly.

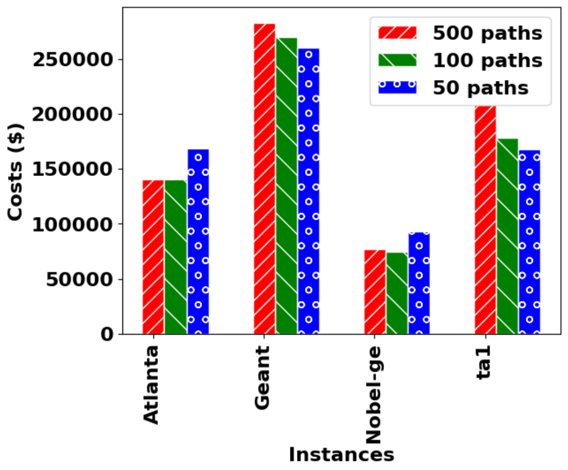

Fig. 6: Costs comparison between path formulation with 500, 100 and 50 paths

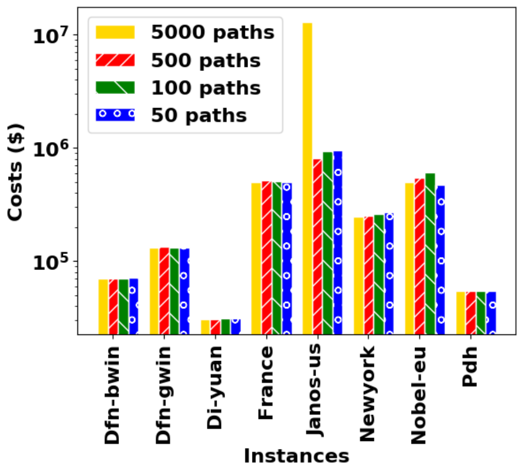

Fig. 7: Costs comparison between path formulation with 5000, 500, 100 and 50 paths

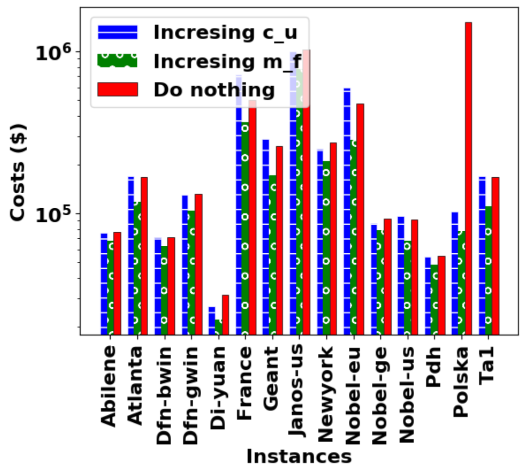

Fig. 8: Costs comparison between Path formulations with 50 path and with/without increasing node and functions capacities

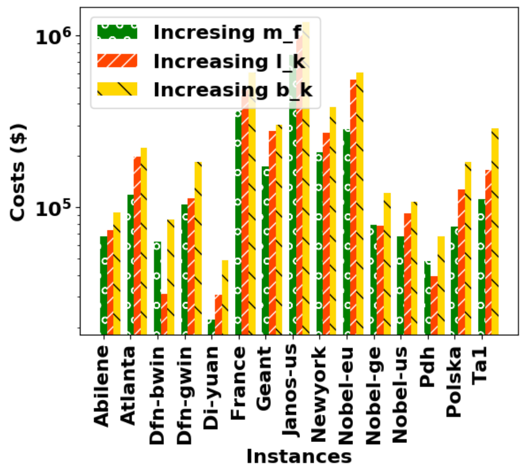

Fig. 9: Costs comparison between Path formulations with 50 path and with increasing function capacities, bandwidth and latency

VI. Conclusions

In this paper we have studied the Virtual Network Functions Placement and Routing Problem for which the sum of the function installation and node activation costs has to be minimized. The studied problem was considered with routing constraints, latency constraints, node and function capacity constraints, incompatibility and chaining constraints. The problem is *NP-hard* and finding an optimal solution in reasonable computing time is challenging. A compact MILP formulation proposed in the earlier literature does not seem to be strong enough for finding a solution using an off-the-shelf solver. To tackle the problem from a computational perspective, we have proposed a path-based formulation which provides an optimal solution for some realistic instances from literature.

We have compared the compact MILP formulation and the path-based formulation in terms of the CPU time and the quality of obtained solution. We have also tested the path formulation with different values of problem parameters and discussed the difference in terms of costs and (in)feasibility.

References

[1] I. Ljubić, A. Mouaci, N. Perrot, and E. Gourdin, "Benders decomposition for the uncapacitated virtual network functions placement and routing problem," *Submitted to Computers & Operations Research*, 2019.

[2] A. Tomassilli, "Towards next generation networks with sdn and nfv," Ph.D. dissertation, 2019.

[3] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on.* IEEE, 2014, pp. 7–13.

[4] N. Huin, A. Tomassilli, F. Giroire, and B. Jaumard, "Energy-efficient service function chain provisioning," *Journal of Optical Communications and Networking*, vol. 10, no. 3, pp. 114–124, 2018.

[5] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1320–1333, 2018.

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

[7] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 4, pp. 1562–1576, 2018.

[8] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on.* IEEE, 2015, pp. 171–177.

[9] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft).* IEEE, 2015, pp. 1–9.

[10] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Computer Communications (INFOCOM), 2015 IEEE Conference on.* IEEE, 2015, pp. 1346–1354.

[11] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications.* IEEE, 2017, pp. 1–9.

[12] B. Addis, G. Carello, F. De Bettin, and M. Gao, "On a Virtual Network Function Placement and Routing problem: properties and formulations," Nov. 2018, working paper or preprint. [Online]. Available: https://halshs.archives-ouvertes.fr/halshs-01643064

[13] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," *Networks*, vol. 71, no. 2, pp. 97–106, 2018.

[14] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[15] "SNDlib," Dec 10, 2019, http://sndlib.zib.de/.

[16] T. Monawar, S. B. Mahmud, and A. Hira, "Anti-theft vehicle tracking and regaining system with automatic police notifying using haversine formula," in *2017 4th International conference on Advances in Electrical Engineering (ICAEE).* IEEE, 2017, pp. 775–779.

[17] L. M. Larsen, A. Checko, and H. L. Christiansen, "A survey of the functional splits proposed for 5g mobile crosshaul networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 146–172, 2018.

[18] D. Chitimalla, K. Kondepu, L. Valcarenghi, M. Tornatore, and B. Mukherjee, "5g fronthaul-latency and jitter studies of cpri over ethernet," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 2, pp. 172–182, 2017.

[19] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN).* IEEE, 2015, pp. 191–197.

[20] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.