On a virtual network functions placement and routing problem: Some properties and a comparison of two formulations

Bernardetta Addis, Giuliana Carello, Meihui Gao

On a Virtual Network Functions Placement and Routing Problem: Some Properties and a Comparison of Two Formulations

Bernardetta Addis[a], Giuliana Carello[b,], Meihui Gao[a]

[a] *Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France*
[b] *Dipartimento di Elettronica, Informazione e Bioingegneria Politecnico di Milano Milano, Italy*

Abstract

The mass diffusion of internet applications, both from computers and mobiles, has yielded to an increasing demand for network services with which the expensive and not flexible hardware appliances cannot keep up. On the other hand, computational capability has become available on the network nodes connected with computing servers and the cloud. This has suggested the Network Functions Virtualization paradigm: services are provided on a software basis thus giving a flexible and cost effective response to the request for services. The Network Functions Virtualization proposes challenging optimization problems such as the Virtual Network Functions chaining problem, where service instances must be located on some network nodes and each demand must be routed through the services it requires. Most of the literature is currently focused on heuristic solutions, rather than on studying the problem properties or comparing approaches. With the aim of investigating the problem properties and comparing existing formulations, both from the theoretical and the numerical points of view, we consider a single service Virtual Network Functions chaining problem, with different link and service capacities and the objective of minimizing the number of installed VNF instances.

Keywords: OR in telecommunications, Networks, Virtual Network Functions, ILP formulations, Location, Network routing

1. Introduction

The ever increasing diffusion of mobile devices and the consequent rise of the demand for network services (such as firewalls, proxies or WAN optimizers) have made it difficult for the hardware-based appliances to keep up with service requests: indeed such appliances cannot be flexibly operated or upgraded with new functionalities. The Network

Email addresses: bernardetta.addis@loria.fr (Bernardetta Addis), giuliana.carello@polimi.it (Giuliana Carello), arielle1023@hotmail.com (Meihui Gao)

Function Virtualization paradigm has been recently proposed to overcome such limitations: instead of satisfying demands using service-specific hardware-based appliances, virtualized services, a.k.a. Virtual Network Functions (VNFs), can be dynamically allocated on generic servers and made available for each demand when necessary. In this way, the ever increasing demand of existing and new services can be met in a more flexible and cost-efficient way.

Starting from 2012, release date of the ETSI white paper [17], Network Function Virtualization technology has received much attention from both industry and academia: several works have been published addressing challenging aspects spreading from the creation of Network Function Virtualization platforms [18, 11] to the optimization of VNF based networks with respect to, for instance, resource efficiency [12] or competitive goals [1].

A key problem arising in implementing the Network Function Virtualization paradigm is the VNF chaining problem: locating VNFs and routing demands to guarantee that each demand passes through a sequence (chain) of VNFs that provides the services it needs. The allocation of demands to VNF instances and the demand routing must satisfy quality requirements, such as delay, congestion, minimal resource utilization (CPU, energy, etc). We can schematically describe the VNF chaining problem as follows: we are given a telecommunication network, where the nodes can have computing capability. [1] Traffic demands must be routed on the network and must be served by a set of VNFs (services), which must be located where computing capability is available. Therefore a traffic demand that needs to access a VNF located in a node must traverse the node itself. For each demand, an order (possibly partial) according to which the VNFs must be traversed may be given. Several constraints can be taken into account, such as demand delay [4], flow compression/decompression [1], incompatibility between VNFs [3], thus leading to several versions of the problem.

1.1. Modeling paradigms: Virtual Network Embedding vs Placement and Routing

In the early works, the VNF chaining problem is modeled using the Virtual Network Embedding (VNE) paradigm: a virtual network is built for each demand representing the demand together with the chain of VNFs to serve it. Then, virtual nodes and links are mapped (embedded) on the physical network. As a consequence, the VNF location and demand routing are defined according to the mapping solution.

The VNF chaining problem with fixed ordered chain can thus be seen as a special case of VNE where the virtual networks to be embedded reduce to linear graphs whose both extreme nodes have a fixed location. Since the VNF chaining problem a special case of VNE, the VNE complexity results cannot be just extended to the VNF chaining problem. Indeed, in [2] it is shown that the VNF chaining problem is easier than the VNE in some particular cases and for some network topologies. Further, the VNF chaining problem differs from the general VNE problem and the VNE representation paradigm is not

[1] The computing capability is provided by servers or cloud which are directly connected to the network nodes. We work under the (commonly used) hypothesis that the network link capacity is tighter than the capacity of the connection between computational servers and nodes. Therefore, as each server is connected directly to only one routing node, we use a compact representation where routing nodes and server nodes are collapsed, simply assigning the computational capacity of a server to the connected node.

always suitable for representing VNF problems, for example when the VNF chain is not ordered, or just partially ordered, nor does it capture all the features of VNF problems, for example the possibility of sharing VNF resources among different demands.

More recently, another modeling framework (the Placement and Routing) has been proposed, where the location and the routing parts of the problem are modeled based on classical location and network design formulations and then bound together through suitable constraints.

1.2. A short literature review

As mentioned, the early papers mainly exploit the VNE paradigm. Even if this paradigm was proved not to be suitable for describing the general VNF chaining problem (see [1] and [3] for more details), we report, in what follows, also some of the works that are based on it, as VNE-based approaches were the first proposed in the telecommunication community. In [9], authors introduce a VNE based MIP formulation. In [4], a VNE based ILP formulation is proposed along with a dynamic programming based heuristic to deal with large size instances. Similarly, a VNE based representation of the problem is proposed in [15], where the focus is on the online version of the VNF chaining problem: three greedy algorithms and a tabu-search based heuristic method are proposed to deal with the VNF online mapping and scheduling. The problem of embedding VNF chains is addressed also in [14], where demands may be rejected and the subset of accepted demands must be maximized while limiting the number of service chains considered. In [12], the authors propose an ILP model based on the mapping of VNF chains on a physical network, although without naming it explicitly. The mapping exploits precomputed paths. The ILP model is used as a step in a heuristic procedure based on a dichotomic search on the number of located VNFs.

The more recent networking literature exploits the fact that the VNF chaining problem shares features with both facility location and network design problems. In [5], the specific Deep Packet Inspection VNF placement problem, where a single type of service is asked, is targeted and modeled as an adaptation of the multicommodity flow problem model. Small instances are solved with a standard ILP solver and for larger instances a centrality-based greedy algorithm is proposed: at each step a new VNF is located in the node that has the highest centrality until all the traffic flows are served or all nodes have a VNF. In [1], authors provide a facility location and demand routing MILP formulation with a generic number of services and no fixed order in the chain, taking into account different latency regimes and traffic compression properties. Their work investigates the trade-off between a legacy traffic engineering related goal (namely maximum link utilization) and the VNF installation cost (calculated as the number of allocated CPUs). An extension of this work [8] considers also ordered (or partially ordered) chains. A math-heuristic is presented to speed up the solution phase. A model similar to [8] is proposed in [3], where additional constraints are added to take into account the incompatibility of certain VNFs and thus imposing that they are located in different nodes. Furthermore, in this work, some demands can be rejected, therefore their routing and VNF assignment can be neglected. A greedy algorithm based on the decomposition of the problem into two steps (routing and location) is proposed. Flows are allocated one by one, and then the VNFs are located on the selected paths.

1.3. Some remarks on the current state of the art

As shown by the above brief review (see [7] for a comprehensive view), most of the works tackle application related problems and focus on developing heuristic methods to solve the problem within a reasonable computational time, mostly by decomposing the problem into two steps (placing the VNFs and then routing the demands). To the best of our knowledge, the different formulations and solution strategies have not been compared on a common data set yet (aside from [8], where a small data set is provided to compare two matheuristics) and the theoretical properties of the problem have not been thoroughly studied, except for some results on the computational complexity [2, 13]. We believe that studying more accurately the VNF chaining problem is of interest, not only for the telecommunication community, where more efficient solution strategies can significantly reduce the management cost and improve the quality of service, but also for the optimization community, indeed, the VNF chaining problem shares features with network design problems (for the demand routing part) and with facility location problems (for the VNF location and the server dimensioning). Both problems are widely studied in the literature. Even if some combinations of facility location and network design problem have been studied in the optimization literature [6], the VNF chaining problem peculiarities (completely general network, different level of capacities: nodes, links, etc.) are still not explored in the optimization literature and provide a challenging topic. Further, as mentioned before, the problems addressed in the literature differ in the considered technical features and assumptions, therefore it is almost impossible to compare the quality of existing solution strategies on a common base.

1.4. Paper contribution

Our goal is to study the VNF chaining problem and its properties from an optimization point of view, starting from a simplified, yet significant version of it: VNF instances are capacitated, as well as network connections. Further, each demand requires a single service, which is the same for all the demands, and must be routed on a simple path (as in [3, 8]). [2] The goal is to minimize the number of installed VNF instances. First of all, we show some properties of the aforementioned problem. Then, we compare the formulations proposed in the literature both theoretically and computationally on a set of instances derived from the SNDLib [16]. The two formulations that can be derived from the literature can be summarized as follows:

- the Split Path (SP) formulation, where the path of each demand is split into two subpaths, one from the origin to the service node and one from the service node to the destination. The SP shares some modeling features with VNE based models, but different from them, it can be easily generalized to the many services chain case (increasing the number of subpaths with the increase of VNFs in the chain,

[2] This assumption is justified by some considerations derived by the real application. First of all, for some telecommunication technologies implementing paths with cycles is not straightforward: for example in label-switching where for each packet (of a given demand) labels are used to signal the next node to be reached (therefore, without adding an additional "memory-policy" of the type "when you pass by a given node change the forwarding path", paths with cycles cannot be implemented); second, but not less important, the introduction of VNF must be transparent for the user and therefore it must produce routing similar to the ones used before introducing the function virtualization paradigm (that usually do not have cycles).

see Appendix B). It is worth noticing that, whereas the VNE paradigm is not correct to describe the general VNF chaining problems, the resulting formulation is quite effective to solve some special cases (see [8]).

- the Placement and Routing (PR) formulation, which is directly derived by the combination of network routing and facility location formulations.

Our results show that the SP formulation must be preferred to the recently most commonly used PR formulation, since SP allows both to solve exactly larger instances (where the PR formulation fails even to find a feasible solution) and to reduce the computational times for small size instances, allowing also to improve existing (and future) ILP-based heuristics.

The remainder of the paper is organized as follows. In Section 2, we formally define the problem and investigate its properties. In Section 3, we introduce the two aforementioned problem formulations and describe their features and relation in terms of continuous relaxation bounds and solutions. We present and analyze the results of computational experiments in Section 4. Finally, Section 5 concludes the paper.

2. Problem description and properties

In this work, we focus on a VNF chaining problem where a single type of VNF is considered. The network is represented by a graph $G(N, A)$, where N represents the set of nodes and A represents the set of arcs (or links). An instance of the VNF can be installed on any node in N. Both links and VNFs are capacitated: let u denote the arc capacity and q the service (VNF) capacity. The network demands are represented by the set D: each demand $k \in D$ is characterized by a source (origin) node o_k, a destination (terminal) node t_k, a demand amount d_k. The demand must be served by (pass through) an instance of the VNF (service), but a demand can pass through a node without using a VNF installed on it. Demands cannot be split and must be routed on simple paths, i.e., the demand is not allowed to deviate from its path to "search" the VNF. Let us denote the problem as *Virtual Network Function Placement and Routing with Simple Path* (VNF-PR_{SP}). In Figure 1, a small example is reported. Two demands are considered with their respective origin and destination nodes. The node in gray represents the service location which serves both the demands. The dashed and dotted lines represent the routing for demand 1 and demand 2, respectively.

The VNF-PR_{SP} in its decision form is NP-complete even with only one type of capacity (either link or service capacity): in [13] the problem is proved to be difficult if nodes are capacitated (the same approach can be used for the service capacity case), while in [2] it is proved to be difficult if only arc capacity is considered.

2.1. General remarks

We assume that only one type of service is available and all the demands require it. However many results can be extended to the multiple service case. In particular, if all the demands ask for the same sequence of services (same types of VNF and same order) the single service and the multiple services cases are equivalent [3].

[3]Despite the fact that these assumptions may seem quite strong to hold for realistic settings, the property is interesting: it can be usefully exploited in ILP-based and decomposition heuristics methods

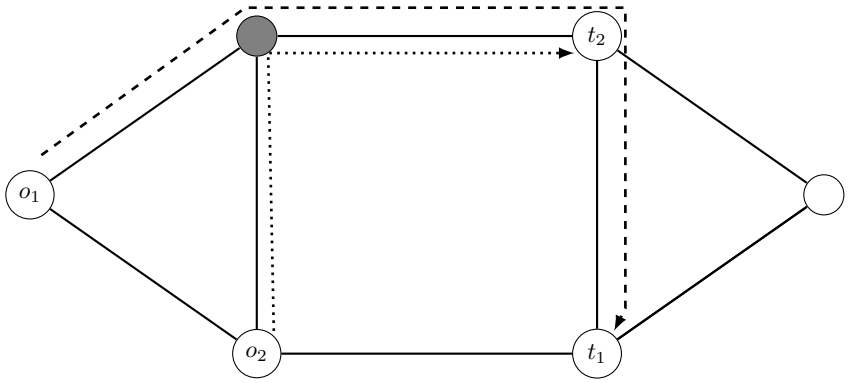


Figure 1: An example of a VNF-PR$_{\text{SP}}$ solution: two demands are considered, the gray node represents the service node.

Property 2.1. Let us consider two problems $\mathcal{P}_1$ and $\mathcal{P}_2$ that share the same features except for the cardinality of the required chain of services:

- in $\mathcal{P}_1$ each demand requires the same service a;
- in $\mathcal{P}_2$ a set of VNF types T is given; each VNF type in T has the same capacity of service a, all the demands require the same sequence of services, both in terms of service types and order.

If there exists an optimal solution for problem $\mathcal{P}_1$, then an optimal solution for $\mathcal{P}_2$ can be derived by installing all the required services in the sites selected by the optimal solution of $\mathcal{P}_1$ and routing the demands according to the optimal solution of $\mathcal{P}_1$.

Proof. A feasible solution for $\mathcal{P}_1$ is given, that is to say a subset of nodes hosting the instances of service a such that all the demands can be routed from their source to their destination passing through an instance of the service a, and respecting link capacity, service capacity and simple path constraints.

Now, let us consider the problem $\mathcal{P}_2$. By installing an instance of each service type in T on the same nodes where the instances of service a are located, and by keeping the same demand-service instance assignment and routing as in the $\mathcal{P}_1$ solution, we obtain a feasible solution for $\mathcal{P}_2$. In fact, installing instances of several different VNF types on one node does not affect either the link capacity constraints or the routing constraints. Furthermore, as the VNF types have the same capacity, if a service a instance installed in a given node i can serve all the demands assigned to it, then any service instance installed on node i can serve the same set of demands.

As any feasible solution of $\mathcal{P}_1$ has the same number of service instances per service type as the derived solution of $\mathcal{P}_2$, if the $\mathcal{P}_1$ solution location is optimal then it is optimal also for $\mathcal{P}_2$.

■

(for example clustering demands according to their requested VNF chain and solving the problem on such clusters) to tackle large size realistic versions of the VNF chaining problem.

We can observe that under the hypothesis that the set of required service types is exactly the same, the two problems are equivalent even if they account for different service orders. [4]

2.2. Impact of biconnected components and articulation points

A lower bound on the number of VNF instances (and thus on the objective function) can be obtained if the graph contains at least one *articulation point*. Let us recall that a graph is *biconnected* if by removing any node the graph remains connected. Further, a *biconnected component* of graph G is a maximal induced biconnected subgraph of G. Biconnected components are connected to each other at shared vertices called *articulation points*: an articulation point of a graph G is a vertex v such that $G \setminus v$ has more connected components than G.

Let us consider a VNF-PR$_{\text{SP}}$ problem defined on a graph G containing biconnected components and suppose that there exists at least one demand whose origin and destination nodes are both in the same biconnected component. Further, suppose that such a biconnected component has only one articulation point. Let us refer to such biconnected components as *biconnected components with internal demands and a single articulation point*. The following property can be stated.

Property 2.2. It is necessary to install at least one service inside each biconnected component with internal demand and a single articulation point. Furthermore, there exists an optimal solution[5] where each of such articulation points hosts a service.

Proof. Let us consider a biconnected component, a demand with both endpoints within it and a node i outside it. If the demand is served by a service installed on node i then its routing must pass first through the articulation point to reach the service and then it must pass again through the same articulation point to complete its routing, thus violating the simple path routing constraint. Thus, in a feasible solution, a service must be installed within a biconnected component to serve the demands whose source and destination belong to the biconnected component itself. Similarly, a demand with both endpoints outside the biconnected component cannot be served by a node in the biconnected component, if it is not the articulation point. Thus a solution where a service is installed in the articulation point is always at least as good as one in which the service is in the biconnected component but not in the articulation point. Therefore, the minimum number of service instances is at least equal to the number of articulation points connecting biconnected components with internal demands and a single articulation point. ■

Thanks to Property 2.2, it is possible to determine a lower bound on the number of VNFs to install combining the number and structure of the biconnected components with the source and destination of the demands. Furthermore, a partial solution can be built, installing services on articulation points. A preprocessing can be devised, which aims at

[4] We recall that it is always possible to schedule the services allocated on the same server in any order, and that in the application problem this order is determined by a suitable scheduling algorithm performed at the server/cloud level.

[5] if the instance is feasible

1. detecting the number of biconnected components with internal demand and a single articulation point, thereby installing one service on each of such articulation points; the search for articulation points can be done in polynomial time through a modified graph search (see [10] for details);

2. forbidding the assignment of a demand to the VNF which is outside of the biconnected component the demand origin and destination belong to.

3. Mathematical formulation

In this section, we present the two formulations derived from the literature which we aim to analyze and compare: the Split Path formulation and the Placement and Routing one.

The first formulation (SP) is based on the decomposition of the path of each demand into several subpaths, each associated with a service instance serving the demand. A similar formulation is presented in [5] (also here a single VNF type is considered, but there is no simple path assumption). The second one (PR) is directly inspired by network design and facility location problems: a set of variables and constraints represent the demand routing and a set of variables and constraints represent the service (facility) location and the demand to service allocation part, connecting constraints are used to couple the two subproblems. It can be considered as the adapted version of the formulation presented in [1] to our VNF-PR$_{\text{SP}}$ problem.

In Table 1, the notation is summarized and the variables used by the two formulations are reported. As some variables are common to both formulations and some are formulation dependent, in the last column we report the formulation in which the parameter/variable is used.

In both formulations, binary variable y_i represents the location of an instance of the VNF on node $i \in N$ and binary variable z_i^k represents the assignment of demand k to the instance of the VNF located on node i. The two formulations differ in the way the routing is modeled. In both, arc binary variables are used, which are equal to one if a given arc is used by a given demand. In PR, these variables are x_{ij}^k. In SP, the path is explicitly divided into two subpaths: the first from the origin o_k to the service node (described by variables x_{ij}^{k1}) and the second from the service node to the destination t_k (described by variables x_{ij}^{k2}).

As we want to enlighten the common points and differences between the two formulations, we present them in parallel, starting from the common part (a summary of the constraints characterizing the two formulations is reported in Table 2).

$$\min \sum_{i \in N} y_i \tag{1}$$

$$\sum_{i \in N} z_i^k = 1 \qquad \forall k \in D \tag{2}$$

$$z_i^k \leq y_i \qquad \forall k \in D, i \in N \tag{3}$$

$$\sum_{k \in D} d_k z_i^k \leq q \qquad \forall i \in N \tag{4}$$

Notation		Formulation
	Sets	
N	set of nodes	both
A	set of arcs	both
D	set of demands	both
	Capacities (Network and Services)	
u	arc capacity	both
q	service capacity	both
	Demand parameters	
o_k	origin of demand $k \in D$	both
t_k	destination of demand $k \in D$	both
d_k	bandwidth of demand $k \in D$	both
	Variables common to both formulations (binary)	
y_i	1 if a service is located on node $i \in N$	both
z_i^k	1 if demand $k \in D$ uses the service on node $i \in N$	both
	Routing variables (binary)	
x_{ij}^k	1 if arc $(i,j) \in A$ is used by demand $k \in D$	PR
x_{ij}^{k1}	1 if arc $(i,j) \in A$ is used by demand k on subpath 1	SP
x_{ij}^{k2}	1 if arc $(i,j) \in A$ is used by demand k on subpath 2	SP
	TSP-like labeling variables (continuous non-negative)	
π_i^k	position of node $i \in N$ in the path used by demand $k \in D$	PR

Table 1: Mathematical notation

The objective function (1) minimizes the sum of the opened services (i.e., instances of the VNF). Constraints (2) impose that each demand is assigned to exactly one instance of the service. Inequalities (3) guarantee that if no VNF instance is installed on a node, then no demand can be assigned to it. Constraints (4) impose that each instance of the VNF can serve a maximum quantity q of demand.

The link capacity constraints are similar for the two formulations:

SP:

$$\sum_{k \in D} d_k (x_{ij}^{k1} + x_{ij}^{k2}) \leq u \qquad \forall (i,j) \in A \tag{5}$$

PR:

$$\sum_{k \in D} d_k x_{ij}^k \leq u \qquad \forall (i,j) \in A \tag{6}$$

We now present the constraints characterizing each formulation. The main difference is in the way the routing is managed, and, as a consequence, in how the formulations deal with the coherence between service assignment and routing. In short, in the SP formulation routing and assignment are implied by modified flow balance constraints, while in the PR formulation the routing is implied by the classical flow balance constraints and

Constraint family	SP	PR
VNF assignment	(2) (3)	(2) (3)
VNF capacity	(4)	(4)
link capacity	(5)	(6)
demand routing	(7) (8)	(11)
simple path	(9) (10)	(13)
additional/connecting	-	(12)

Table 2: Summary of the constraints defining the SP and PR formulations

the consistency between assignment and routing is implied by coherence constraints and isolated loop elimination constraints.

Routing and assignment are modeled as follows:

SP:

$$\sum_{j:(i,j)\in A} x_{ij}^{k1} - \sum_{j:(j,i)\in A} x_{ji}^{k1} = \begin{cases} 1 - z_i^k & \text{if } i = o_k \\ -z_i^k & \text{otherwise} \end{cases} \quad \forall k \in D,\ i \in N \tag{7}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{k2} - \sum_{j:(j,i)\in A} x_{ji}^{k2} = \begin{cases} z_i^k - 1 & \text{if } i = t_k \\ z_i^k & \text{otherwise} \end{cases} \quad \forall k \in D,\ i \in N \tag{8}$$

$$\sum_{j:(j,i)\in A} (x_{ji}^{k1} + x_{ji}^{k2}) \leq 1 \ \forall k \in D, i \in N \tag{9}$$

$$\sum_{j:(i,j)\in A} (x_{ij}^{k1} + x_{ij}^{k2}) \leq 1 \ \forall k \in D, i \in N \tag{10}$$

Each demand is routed on two subpaths: from the source node to the VNF node (equations (7)), then from the VNF node to the destination node (equations (8)). These two constraints impose that the routing of the demand passes through the VNF instance assigned to the demand itself, therefore ensuring that the assignment is consistent.

Simple path routing[6] is imposed by constraints (9) and (10): for each demand, at most one of the subpaths can pass through a node.

PR:

$$\sum_{j:(i,j)\in A} x_{ij}^{k} - \sum_{j:(j,i)\in A} x_{ji}^{k} = \begin{cases} 1 \text{ if } i = o_k \\ -1 \text{ if } i = t_k \\ 0 \text{ otherwise} \end{cases} \quad \forall k \in D,\ i \in N \tag{11}$$

[6]Isolated cycles with no service can be part of a feasible solution. Such cycles can be removed obtaining a cycle-free equivalent feasible solution.

$$z_i^k \leq \sum_{(j,i)\in A} x_{ji}^k \ \forall k \in D,\ i \in N \setminus \{o_k\} \tag{12}$$

$$\pi_j^k \geq \pi_i^k + x_{ij}^k - |N|\ (1 - x_{ij}^k)\ \forall k \in D,\ (i,j) \in A \tag{13}$$

Each demand is routed from its source to its destination with the classical flow balance constraints (11) and it is forced to pass through the VNF instance to which it is assigned by constraints (12), that impose that a demand k can be assigned to a service located on node i only if the routing path of the demand passes through the given node. The simple path and the elimination of isolated cycles are enforced using the TSP-like labeling variables π and constraints (13): continuous variables π_i^k represents the position of node i in the routing path of demand k.

Both formulations can be straightforwardly generalized to take into account multiple service types, even hosted on different nodes. However, when a partial (or no) order is asked, the SP formulation needs a new set of variables to partially decouple the path-splitting and service order allocation (see Appendix B). Further, the number of subpaths, and of the related variables, increases linearly with the increasing number of services in the chain. Instead, the PR formulation can be simply generalized to deal with any imposed order (full, partial, none). In fact, variables π can be used, together with additional constraints, to impose a given full or partial order of the services along the demand path (see [8] for details).

3.1. Relation between the two formulations

In this section, we highlight some properties of the feasible regions of the continuous relaxations of the two formulations. In what follows we will refer to the continuous relaxation of SP (PR) as SP_r (PR_r). The main property is the following:

Theorem 3.1. The SP_r produces a continuous relaxation bound that is always not worse than the one produced by PR_r.

Furthermore, it can be shown that for some instances the gap between the two relaxations is greater than zero (see Example 3.1).

The full proof of Theorem 3.1 is reported in Appendix A, however we believe that it is worth reporting here the main ideas (and properties) behind the proof, as they give some insights into the structure of the feasible regions of SP_r and PR_r.

Remark 3.1. The SP_r forbids demands to be served, even partially, by a VNF instance installed on an isolated cycle. Instead PR_r accepts solutions with an isolated cycle and a partial service installed on it. [7]

[7] As for Travelling Salesman Problem (TSP), isolated loop elimination constraints (13) are effective only in the integer formulation.

Therefore we can partition the feasible region of the SP_r into two subsets: in the first there are no isolated cycles while in the second one isolated cycles are present, but they do not host a service. Any solution of the second subset has an equivalent in the first one (as in classical flow problems).

Then it is necessary to prove that any solution belonging to the first subset can be transformed into a solution of PR_r. The mapping of variables x, y and z is quite straightforward. To map π variables, for each demand k, it is necessary to decompose the resulting solution flow into flow on simple paths (P_k: set of simple paths) and flow on cycles (C_k: set of cycles). As for the paths, if the value of routing variables x_{ij}^k is considered as the length of the corresponding arc (i, j) then, for any node i belonging only to simple paths, the π_i variable can be set as the longest path distance of such node i from the source node o_k. Such values satisfy constraints (13) by construction for all couples of nodes both belonging to the path. As for the cycles, we resort to the following remark:

Remark 3.2. Let us consider a feasible solution of SP_r (x^1, x^2, y, z), a demand k and the path-cycle decomposition $\{P_k, C_k\}$ of its routing. Let us suppose that a cycle $c \in C_k$ exists that shares at least one node with a simple path $p \in P_k$.
If a (partial) service is located on a node belonging to the cycle $c \in C_k$, but not to the path p, the routing variables inducing the path and the cycle are fractional and their value is less than or equal to $\frac{1}{2}$.[8]

We can now determine a value of π for the nodes belonging only to a cycle (nodes from m to ℓ in Figure 2).

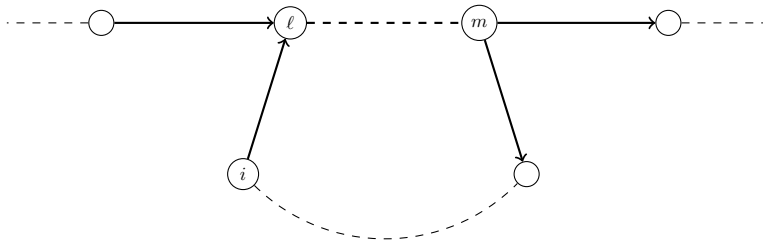


Figure 2: An example of a cycle sharing some nodes with a simple path

Let us consider the two nodes ℓ and m, corresponding to the smallest (π_ℓ^k) and largest (π_m^k) value of variables π on the cycle, respectively. If for all the nodes $i \in N_c$, we chose

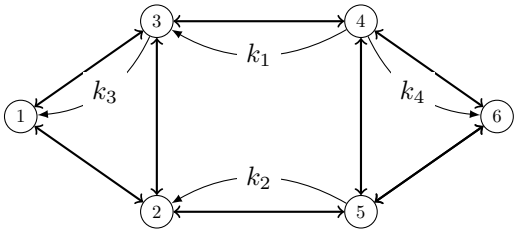
$$\pi_i^k = \frac{\pi_m^k + \pi_\ell^k}{2}$$

constraints (13) are always satisfied. The proof is based on the fact that variables π are bounded by $|N| - 1$ and that the (SP_r) routing variables for any arc (i, j) belonging to the cycle, but not a path, are bounded by $\frac{1}{2}$.

We now provide an instance in which the bound provided by SP_r is stricter than the one provided by PR_r.

[8]The proof is presented in Appendix A, see Property A1.

Example 3.1. Let us consider the symmetric graph in Figure 3 (in the following figures, each pair of symmetric arcs is represented by the corresponding edge). The services are uncapacitated, and the link capacity is 5. There are 4 demands, whose characteristics are listed in Figure 3b.



(a) Topology and demands distribution.

k	$o_k \to t_k$	d_k
1	$4 \to 3$	**5**
2	$5 \to 2$	**5**
3	$3 \to 1$	$\forall d_3 \in (0, 5]$
4	$4 \to 6$	$\forall d_4 \in (0, 5]$

(b) Demands.

Figure 3: The SP_r bound is strictly better than the one of PR_r: a numerical example

The SP_r optimal solution value is equal to 2, while the PR_r one is 1, for any value of the demands k_3 and k_4 in $(0, 5]$. In Table 3, we report an optimal solution for the two formulations: the location of service instances on nodes is reported in bold. As for the SP_r formulation, one service is located on node 4 and one on node 3. Demands k_1, k_2 and k_4 are served by the service located on node 4, while demand k_3 is served by the service located on node 3. As for the solution of PR_r, half service is installed on node 4 and the other half on node 5; each demand uses both services.

demand	SP_r paths	f_p	z_i^k	PR_r paths	f_p	z_i^k
k_1	$p_1^1 : \mathbf{4} \to 5 \to 2 \to 3$	1	1	$p_1^1 : 4 \to \mathbf{5} \to 2 \to 3$	0.5	0.5
				$p_2^1 \mathbf{4} \to 3$	0.5	0.5
k_2	$p_1^2 : 5 \to \mathbf{4} \to 3 \to 2$	1	1	$p_1^2 : 5 \to \mathbf{4} \to 3 \to 2$	0.5	0.5
	-			$p_1^2 : \mathbf{5} \to 2$	0.5	0.5
k_3	$p_1^3 : \mathbf{3} \to 1$	1	1	$p_1^3 : 3 \to 1$	1	-
	-			$p_2^3 : 5 \to \mathbf{4} \to \mathbf{5}$	0.5	0.5 0.5
k_4	$p_1^4 : \mathbf{4} \to 6$	1	1	$p_1^4 : \mathbf{4} \to 6$	1	0.5
	-			$p_2^4 : 5 \to 6 \to \mathbf{5}$	0.5	0.5
services	$y_3 = 1 \quad y_4 = 1$			$y_4 = 0.5 \quad y_5 = 0.5$		

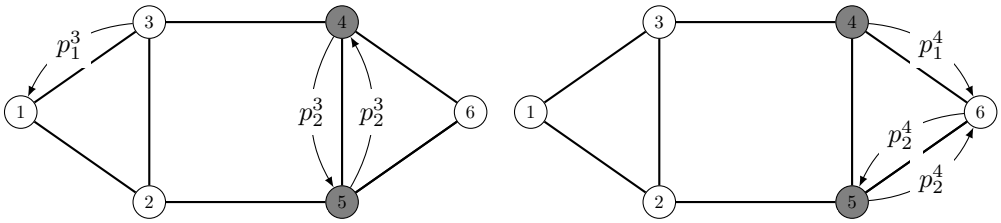
Table 3: Routing and assignment/location for SP_r and PR_r

This example allows us to make some observations on the structure of the feasible solutions of the two relaxations. Indeed, there exist some families of solutions that are feasible for PR_r, but not for SP_r. In this example, we can see two of them (see Figure 4):

- any solution where a source-destination path does not pass through any service node and an isolated loop hosts a (partial) service (as for demand k_3) (see Remark 3.1)

- any solution where the routing variables on a path (f_p in the table) have a different value from the assignment variables (path p_1 for demand k_3 and path p_1 for demand k_4).

Remark 3.3 highlights an additional property that is required by the SP_r formulation (but not PR_r), indeed, for SP, the routing and assignment variables have to be consistent also in the continuous relaxation. In this sense, the SP formulation shares some similarities with the min-cost-flow problem: each service node can be imagined as both sink and source of a demand and assignment variables z_i^k as the quantity of demand that is absorbed/produced by the node (depending if we are looking at the subpath entering or exiting the node). The main difference is that in our problem the sink/source nodes are decision variables and not given data.



(a) Routing of demand k_3 (b) Routing of demand k_4

Figure 4: Routing solution for demand k_3 and k_4 for the PR_r

Solutions with isolated loops hosting a partial service may be profitable when the capacity of a cut is small and some demands cannot reach the services installed on the other side of the cut itself: using a service on an isolated cycle is then the best option for PR_r. Looking again at the example described in Figure 3, we can observe that demands k_1 and k_2 saturate the cut $\{4,5,6\},\{1,2,3\}$ forcing SP_r to install a service in each side of the cut to serve also demands k_3 and k_4. Instead, PR_r does not open two services, thus providing a weaker bound. The bound provided by SP_r is stricter even if the amount of flow of demands k_1 and/or k_2 decreases and thus the cut is not fully saturated, as long as the residual capacity on the cut $\{4,5,6\},\{1,2,3\}$ is not enough to allow the full demand k_3 to pass through it and back (or symmetrically demand k_4 for the reversed cut). A feasible solution is then selected by PR_r as described in Figure 5, where the demand is half routed ($x = 0.5$) on the path and is completely served by partially using the two services located in node i and j.

Such solution is instead infeasible for SP_r, [9] resulting in the following general remark:

Remark 3.3. Let us consider a feasible solution for SP_r formulation where a fraction of a demand k is routed on a path p_k. Let us consider the services located along this path and the corresponding assignment variables z_i^k, and suppose that such services are not

[9] The larger the fraction of demand k_3 that can pass the cut, the smaller the gap between the two continuous relaxation bounds.

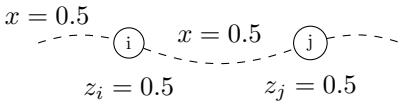


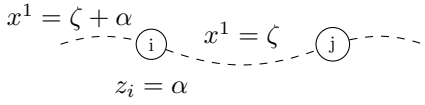
Figure 5: Example of infeasible flow acceptable for PR_r

used by demand k along other paths. Then:

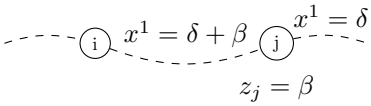
$$x_{lm}^{k1} \geq \sum_{i \in p_k : i = \text{succ}(lm)} z_i^k \tag{14}$$

where with succ(lm) we represent all the nodes that appear in the path p_k not before arc (l, m) (node m is considered belonging to succ(lm)).

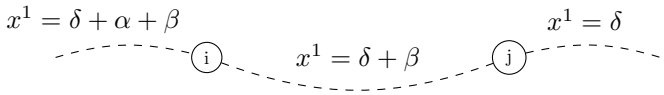
This property is a direct consequence of the modified flow balancing constraints (7)-(8). In Figure 6 a schematic illustration is presented.



(a) flow balance on node i



(b) flow balance on node j

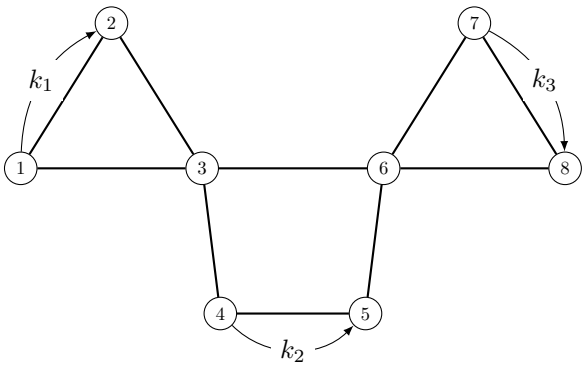


(c) resulting feasible values for x^1

Figure 6: Relation between routing and assignment variables for SP_r

3.1.1. Impact of the biconnected components

Network topology has also an impact on the quality of the bounds produced by SP_r and PR_r. In fact, SP_r can produce a tighter bound in presence of biconnected components, even in the uncapacitated case, thanks to the combination of simple path constraints and routing/assignment constraints.



(a) Topology and demands distribution.

k	$o_k \to t_k$	d_k
1	$1 \to 2$	1
2	$4 \to 5$	1
3	$7 \to 8$	1

(b) Demands.

Figure 7: The SP_r bound is strictly better than the one of PR_r: a numerical example with biconnected components

Let us consider the graph in Figure 7a, and a set of three demands described in Figure 7b. Service and link capacities are unbounded. The graph contains two articulation points (nodes 3 and 6) and 3 biconnected components:

$$BC_1 = \{1, 2, 3\}, \quad BC_2 = \{3, 4, 5, 6\}, \quad BC_3 = \{6, 7, 8\}.$$

The SP_r bound for this instance is $\frac{4}{3}$, the one obtained by the PR_r is the trivial bound (1).

In Table 4, solutions[10] for the SP_r and the PR_r formulations are reported. For each path, the node where the (partially) used service is located is reported in bold. In SP_r solution, partial services are installed on nodes $2, 5$ and 8, half for nodes 2 and 8, and one third for node 5. Demand k_1 (k_3, respectively) is served by the half service installed on node 2 belonging to its connected component BC_1 (node 8 in BC_3 respectively) and by half service installed on node 8 in connected component BC_3(node 2 in BC_1, respectively). Demand k_2 is split on three paths, and each of them is served by a (partial) service in a different connected component.

For the PR_r formulation, each demand is routed on the direct arc connecting its origin to its destination, thus satisfying the flow balance constraints (11). To reach the service, both demand k_2 and k_3 use two isolated cycles each in the connected component BC_1. It is worth noticing that a single cycle would not be enough to satisfy both the coherence constraints (12) and the TSP-like cycle elimination constraints (13) (only fractional solutions can have cycles).

It is worth noting that in this example the bound provided by Property 2.2 is 2, which improves upon both the SP_r and PR_r ones.

[10]For both formulations several equivalent solutions exist, both for location and routing. For the sake of clarity, we chose to show a "compact" solution for both formulations, preferring symmetric, less fractional solutions, with short routing and a reduced number of isolated cycles.

demand	SP_r paths	f_p	z_i^k	PR_r paths	f_p	z_i^k
k_1	$p_1^1 : 1 \to \mathbf{2}$	0.5	0.5	$p^1 : \mathbf{1} \to 2$	1	1
	$p_2^1 : 1 \to 3 \to 6 \to \mathbf{8} \to 6 \to 3 \to 2$	0.5	0.5	-		
k_2	$p_1^2 : 4 \to \mathbf{5}$	1/3	1/3	$p_1^2 : 4 \to 5$	1	-
	$p_2^2 : 4 \to 3 \to \mathbf{2} \to 3 \to 6 \to 5$	1/3	1/3	$p_2^2 : \mathbf{1} \to 2 \to 1$	0.5	0.5
	$p_3^2 : 4 \to 3 \to 6 \to \mathbf{8} \to 6 \to 5$	1/3	1/3	$p_3^2 : \mathbf{1} \to 3 \to 1$	0.5	0.5
k_3	$p_1^3 : 7 \to \mathbf{8}$	0.5	0.5	$p_1^3 : 7 \to 8$	1	-
	$p_2^3 : 7 \to 6 \to 3 \to \mathbf{2} \to 3 \to 6 \to 8$	0.5	0.5	$p_2^3 : \mathbf{1} \to 2 \to 1$	0.5	0.5
	-			$p_2^3 : \mathbf{1} \to 3 \to 1$	0.5	0.5
services	$y_2 = 0.5 \quad y_5 = 1/3 \quad y_8 = 0.5$			$y_1 = 1$		

Table 4: Routing and assignment/location for the biconnected components case example for SP_r and PR_r

3.2. Enriching the formulation

The formulations can be enriched by adding two types of additional constraints. To derive the first one, we can observe that the total demand that can be served by a service located on the node i is limited not only by the service capacity, but also by the overall capacity of incident links (except for the demands served in the origin node i). The same reasoning can be applied to the outgoing links. Therefore, the maximal demand that can be served by a node can be calculated as follows:

$$\bar{q_i} = \min \left\{ q, \max \left\{ \sum_{(i,j) \in A} u + \sum_{k \in D : t_k = i} d_k, \sum_{(j,i) \in A} u + \sum_{k \in D : o_k = i} d_k \right\} \right\} \tag{15}$$

This allows us to obtain a strengthened version of constraint (4) (using both the service location variable y_i and the capacity $\bar{q_i}$):

$$\text{VI1:} \quad \sum_{k \in D} d_k \; z_i^k \;\leq\; \bar{q_i} \; y_i \; \forall i \in N \tag{16}$$

When the capacity of a single VNF instance is not large enough to serve all the demands, a bound on the number of needed VNF instances can be calculated (similar to bin-packing problems):

$$\text{VI2:} \quad \sum_{i \in N} y_i \;\geq\; \left\lceil \frac{\sum_{k \in D} d_k}{q} \right\rceil \; \forall i \in N \tag{17}$$

In the following, we will refer to inequality (16) as VI1 and to inequality (17) as VI2.

4. Computational Results

We performed computational tests on the SP and PR formulations to

- compare the performance of the two formulations and evaluate the impact of the VIs (Section 4.1)

- evaluate the scalability of the two formulations (Section 4.2)
- evaluate the impact of Property 2.1 and the impact of increasing the number of services (Section 4.3)
- evaluate the impact of the articulation point based preprocessing (Section 4.4)

To this purpose we have generated a test bed based on 24 instances from the SNDLib [16]. For each instance (topology and set of demands), we have generated different capacity profiles to analyze the impact of the VNF and the link capacity:

- as for the links, two levels of capacity have been generated: low and high. The high capacity is such that each link can host all the demands, thus leading to uncapacitated link instances. The low capacity is computed as the minimum capacity such that a feasible routing exists, neglecting the services;
- as for the services, three levels of capacity are considered: low, medium and high. The high capacity is computed to guarantee that all the demands can be served by a single VNF (service uncapacitated instances). Low VNF capacity is twice the total amount of the demands divided by the number of nodes, that is, we need to install a VNF in at least half of the nodes (for lower values many instances were not feasible due to network topologies). Medium capacity is the average between high and low.

In the following, we denote with *h, m* and *l* the *high, medium* and *low* capacity level respectively. For example instances marked with l_h are the ones where service capacity assumes the lowest value and the link the highest (therefore uncapacitated with respect to links). The features of the obtained 144 instances are summarized in Table 5, where each row refers to a network topology. Columns 3 to 5 report the network features from SNDLib (number of nodes, number of links and number of demands). Column 6 gives the sum of the demand amount based on the SNDLib values. Such a value corresponds to the high capacity value both for links and services. The last 3 columns give the remaining values of capacity of VNFs and links: the first 2 report the medium and low capacity values for the VNFs and the last 1 reports the low capacity value for the links. The first 16 network topologies, from di-yuan to norway, have less than 30 nodes (small-medium) and are used in the formulations comparison; 6 topologies, from india35 to germany50, have more than 30 nodes and less than 50 (large) and are used to assess formulation scalability; for the abilene, atlanta and dfn-bwin topologies instances have been generated with three types of services to assess the relevance of Property 2.1; the topologies france, ta2 and zib54 have articulation points (AP), thus they have been used to assess the impact of adding the articulation point based preprocessing.

Models are implemented with AMPL and instances are solved with IBM ILOG CPLEX 12.7.1.0 on an Intel Xeon, CPU E5-1620 v2 (4 cores), 3.7 GHz with 32 GB of RAM. A time limit of 3600s and a tree memory limit of 3000 MB are set, but for solving the large instances: when assessing the scalability, the time limit has been extended to two hours.

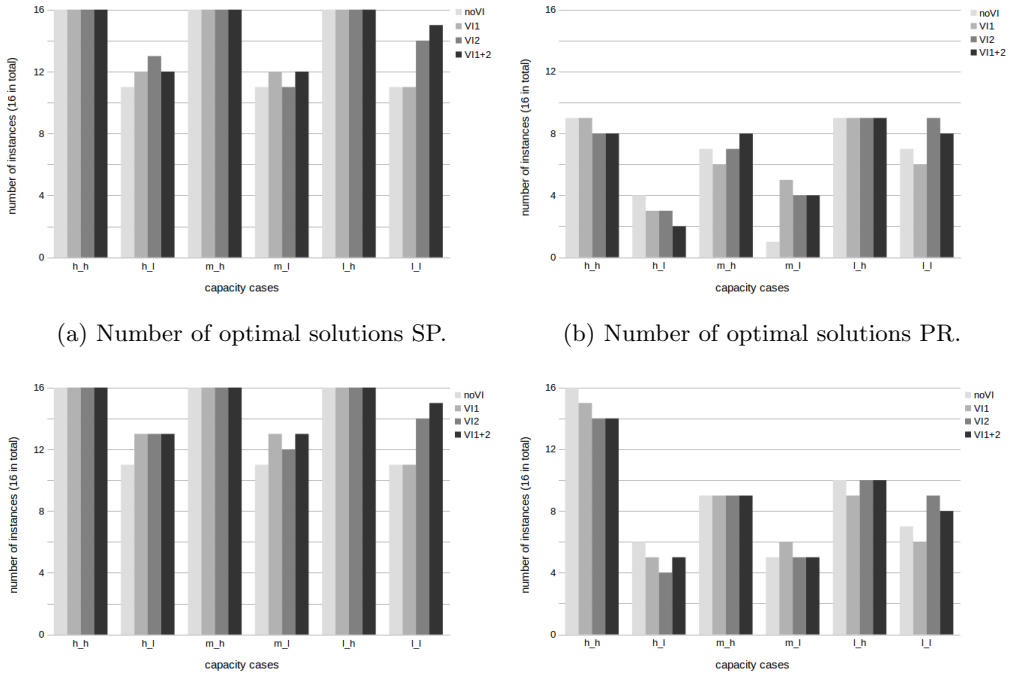
size	Network	Data from SNDLib $\|N\|$	$\|A\|$	$\|D\|$	$\sum_{k\in D} d_k$ (high cap)	Capacity Service medium	Service low	Link low
small-medium	di-yuan	11	42	22	53	31	9	5
	pdh	11	34	24	4621	2730	840	384
	polska	12	18	66	9943	5800	1657	995
	sun	27	51	67	476	255	35	53
	dfn-bwin	10	45	90	548388	329032	109677	55916
	nobel-us	14	21	91	5420	3097	774	486
	nobel-germany	17	26	121	660	368	77	74
	abilene	12	15	132	3000002	1750001	500000	829282
	atlanta	15	22	210	136726	77478	18230	19404
	newyork	16	49	240	1774	997	221	66
	france	25	45	300	99830	53908	7986	9413
	nobel-eu	28	41	378	1898	1016	135	214
	ta1	24	51	396	10127249	5485593	843937	819678
	geant	22	36	462	2999992	1636359	272726	359868
	janos-us	26	42	650	80000	43076	6153	7624
	norway	27	51	702	5348	2872	396	358
large	india35	35	80	595	3292	1740	188	121
	cost266	37	57	1332	679598	358166	36735	53562
	giul39	39	86	1471	7366	3871	377	363
	janos-us-ca	39	61	1482	2032274	1068246	104219	180471
	pioro40	40	89	780	115953	60875	5797	7609
	germany50	50	88	662	2365	1229	94	123
AP	zib54	54	80	1501	12230	6341	484	528
	ta2	65	108	1869	31419014	16192876	966738	1311190

Table 5: Instance details

4.1. Results on small-medium instances

Let us first compare the two formulations on the small-medium size instances with less than 30 nodes. In Figures 8a - 8c (and respectively Figures 8b - 8d), the number of optimal and integer feasible solutions found by the SP (respectively PR) formulation, with and without the additional valid inequalities, are reported.

Results show that SP outperforms PR in all the capacity cases, whatever additional inequalities are applied. Indeed, in the high link capacity cases, SP can solve to optimality all the instances with any additional inequalities choice, while PR can solve at most 25 out of 48 instances (the best performance is obtained when no additional inequality is used or both of them are added). In the low link capacity cases, SP can solve more than 30 instances while PR can solve 12 instances with no additional inequalities, 16 instances using only inequality VI2 and 14 in the other two cases. SP seems to find the optimal solution almost every time it can find a feasible one: indeed, it is not able to prove the optimality of the feasible solution found only in very few instances with h_l and m_l capacity settings. When SP cannot prove optimality, the gap is still reasonable



(a) Number of optimal solutions SP.

(b) Number of optimal solutions PR.

(c) Number of feasible solutions SP.

(d) Number of feasible solutions PR.

Figure 8: Number of optimal and integer feasible solutions found

(at most around 27% for janos_us instance with h_l and m_l capacities). The number of instances where a feasible but not optimal solution is found is very small also for PR. It seems therefore that finding a feasible solution is in a sense as challenging as proving its optimality.

SP outperforms PR also as far as the computational times are concerned. As the PR formulation cannot find a feasible solution for several instances of this group (except for the h_h capacity case), we restrict our comparison only to those instances where the PR formulation is able to find a feasible solution for a capacity case different from h_h with at least one VI setting, namely instances from di-yuan to france (up to 300 demands).

The comparison is shown in Table 6 for the high link capacity cases and Table 7 for the low link capacity cases. In each table instances are grouped based on the service capacity. For each instance we report, for both formulations, the computational time needed by the fastest setting (Best), namely the choice of additional valid inequalities that requires the minimum computational time to solve the instance, and the computational time needed by the most time consuming setting (Worst). TL-IS indicates that the approach is not able to solve the instance to optimality, but it can provide an integer feasible solution in the given time limit, whereas TL indicates that the approach cannot even find a feasible solution. To show that the SP formulation outperforms the PR one

Capacity	Instance	Best SP	Worst SP	Best PR	Worst PR	Best(PR) vs Worst(SP)
	di-yuan	0.10	0.13	1.59	23.86	**11.07**
	pdh	0.09	0.20	3.11	15.24	**14.38**
	polska	0.20	0.29	13.22	29.34	**44.79**
	sun	0.43	0.85	213.41	TL-IS	**251.03**
	dfn-bwin	0.37	0.56	56.39	58.40	**99.64**
h_h	nobel-us	0.52	0.62	40.24	683.65	**63.92**
	nobel-germany	0.39	0.86	52.92	82.63	**60.68**
	abilene	0.41	0.48	6.53	8.92	**12.71**
	atlanta	0.90	2.37	1167.43	3430.02	**490.95**
	newyork	1.49	4.28	TL-IS	TL-IS	-
	france	1.67	2.42	TL-IS	TL-IS	-
	di-yuan	0.09	0.14	1.95	10.17	**13.40**
	pdh	0.09	0.32	2.60	16.53	**7.16**
	polska	0.19	2.24	9.59	21.81	**3.29**
	sun	0.91	8.88	465.07	TL-IS	**51.35**
	dfn-bwin	0.34	2.19	1360.67	TL-IS	**619.06**
m_h	nobel-us	0.60	2.73	10.69	2354.00	**2.92**
	nobel-germany	0.69	7.29	178.74	TL-IS	**23.52**
	abilene	0.60	1.68	3.74	19.26	**1.23**
	atlanta	1.49	12.01	341.28	TL	**27.41**
	newyork	4.18	20.47	TL-IS	TL	-
	france	2.20	2.81	TL	TL	-
	di-yuan	0.15	0.27	1.25	2.56	**3.59**
	pdh	0.11	0.25	0.74	68.72	**1.94**
	polska	0.23	1.15	0.38	9.10	-2.03
	sun	3.18	5.61	28.87	343.29	**4.15**
	dfn-bwin	0.48	1.26	3.39	192.09	**1.70**
l_h	nobel-us	0.52	2.18	1.35	9.02	-0.62
	nobel-germany	0.73	5.41	11.95	92.18	**1.21**
	abilene	0.50	2.14	0.82	6.35	-1.59
	atlanta	1.33	13.53	6.09	182.98	-1.22
	newyork	1.92	27.60	TL	TL	-
	france	6.23	335.67	TL-IS	TL	-

Table 6: Time comparison between SP and PR formulations for the high link capacity cases. Only best (and worst) times with respect to the different possible VIs choices are reported. Times are expressed in seconds. The time limit is set to 3600s.

(regardless of the setting), we report, in the last column, the difference between the best computational time of PR and the worst computational time of SP normalized by the smallest computational time ($\frac{CPUtime_{BestPR} - CPUtime_{WorstSP}}{\min(CPUtime_{BestPR}, CPUtime_{WorstSP})}$): a '-' denotes the instances in which one or both the formulations cannot be solved to optimality. We highlight with bold font the cases where even the worst setting of SP outperforms PR.

In the high and medium service capacity case, the worst performance of SP is always

Capacity	Instance	Best SP	Worst SP	Best PR	Worst PR	Best(PR) vs Worst(SP)
	di-yuan	0.76	3.65	168.28	1506.64	**45.07**
	pdh	2.79	53.59	296.77	TL-IS	**4.54**
	polska	435.22	TL	353.85	TL-IS	-
	sun	63.72	574.84	TL	TL	-
	dfn-bwin	2.94	7.85	TL-IS	TL	-
h_l	nobel-us	478.86	3065.77	TL	TL	-
	nobel-germany	9.84	13.28	TL-IS	TL	-
	abilene	2.37	11.23	491.14	3266.12	**42.72**
	atlanta	1065.05	1471.85	TL	TL	-
	newyork	TL	TL	TL	TL	-
	france	TL	TL	TL	TL	-
	di-yuan	0.35	1.28	11.60	TL-IS	**8.05**
	pdh	1.33	9.18	26.89	TL-IS	**1.93**
	polska	98.69	TL	1214.45	TL	-
	sun	24.11	170.48	TL	TL	-
	dfn-bwin	1.21	11.34	458.44	TL-IS	**39.41**
m_l	nobel-us	116.59	1978.38	TL	TL	-
	nobel-germany	6.73	15.20	TL-IS	TL	-
	abilene	1.81	18.54	161.16	1548.00	**7.69**
	atlanta	663.40	1842.73	TL	TL	-
	newyork	TL	TL	TL	TL	-
	france	TL	TL	TL	TL	-
	di-yuan	0.27	0.61	0.42	3.31	-0.45
	pdh	0.38	1.32	3.87	166.79	**1.95**
	polska	193.06	TL	55.09	711.47	-
	sun	8.59	103.78	TL	TL	-
	dfn-bwin	0.76	9.95	15.91	1277.22	**0.60**
l_l	nobel-us	8.50	551.17	124.12	1725.29	-3.44
	nobel-germany	1.97	26.68	50.85	TL	**0.91**
	abilene	2.12	34.41	2.43	66.62	-13.17
	atlanta	11.98	1798.66	769.79	TL	-1.34
	newyork	1409.36	TL	TL	TL	-
	france	TL	TL	TL	TL	-

Table 7: Time comparison between SP and PR formulations for the low link capacity cases. Only best (and worst) times with respect to the different possible VIs choices are reported. Times are expressed in seconds. The time limit is set to 3600s.

better than the best performance of PR. The difference is significant in many instances with high link capacity. Low link capacity instances seem to be more challenging both for SP and PR: indeed the increase of the SP computational time passing from the high link capacity instances to low link capacity instances is larger than the one of PR. However, the worst SP setting remains always faster than the best PR one, which cannot solve to optimality a significant number of instances. When it comes to the low service capacity,

the PR fastest VI choice improves upon the worst SP one in some instances. However, the difference is smaller than for the high service capacity cases and SP is always faster (except for the instance polska with l_l capacities) than PR, if the best computational times are compared. As the SP formulation in general outperforms the PR one regardless of the VI settings, we present a detailed analysis of the impact of VIs only for SP.

4.1.1. Detailed analysis of the impact of VIs addition for the SP formulation

The impact of adding valid inequalities on the number of optimal and feasible solutions is reported in Figure 8. Results show that the addition of inequalities improves the number of optimal or feasible solutions found by SP (or leaves them unchanged).

The impact of adding valid inequalities to the SP formulation on the computational times is summarized in Figure 9, where for each considered VIs choice, the number of instances where such choice is the fastest is reported, while Tables 8 and 9 report the details of the computational times for the different VIs choice: the third column (Best SP) reports the best computational time among the different VIs choices, while the last 4 columns report, for each VIs choice, the difference between its computational time and the best one normalized w.r.t. the best (calculated as $\frac{CPUtime_i - CPUtime_{Best}}{CPUtime_{Best}}$).

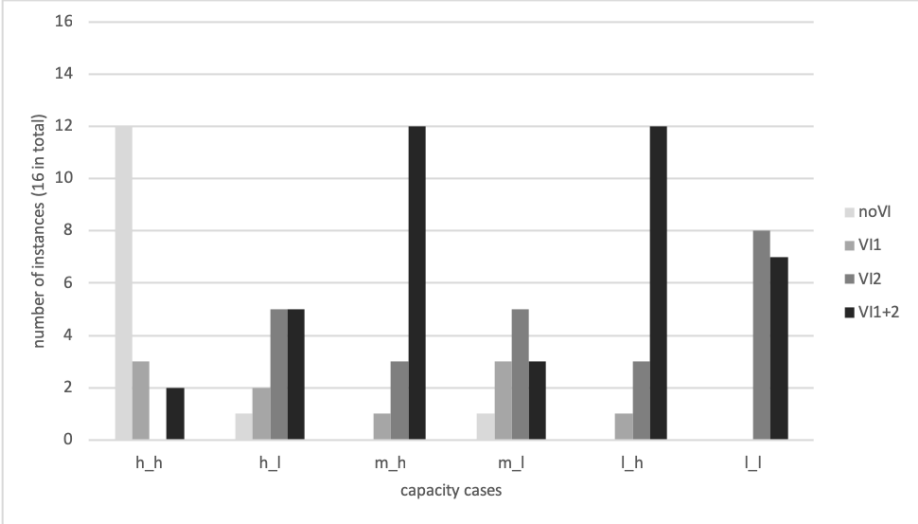


Figure 9: Speed of SP formulation with the different VIs choices reported as number of instances where each choice is the fastest (for each capacity case).

The addition of valid inequalities almost always reduces the computational time needed to find the optimal solutions. Indeed, the formulation without additional inequalities turns out to be the fastest almost only in instances with both high link and service capacities (h_h case). In general, the impact of VI1 is more evident when the link capacity is low and the service capacity is high. This is reasonable, because with such capacity setting the multicommodity nature of the problem emerges and the capacity of incident links is the one that limits the access to services. Symmetrically, when the service capacity is low, the VI2 shows its effectiveness more clearly. Adding both valid

inequalities seems to produce the best results: it is the fastest option in the majority of the instances.

4.2. Scalability with respect to the network size and number of demands

We have considered 6 network topologies, from cost266 to pioro40, that have from 35 to 50 nodes and from about 600 to about 1500 demands, to assess the scalability of the formulations with respect to the network size and number of demands. We increased the time limit to two hours. Both the additional inequalities have been added.

PR is not even able to provide an integer feasible solution for any of the considered instances, no matter the capacity settings. For some capacity settings, namely h_l and m_l, also SP is not able to provide any feasible solution within the time limit. In Table 10, the computational times of SP for the 6 considered instances are reported, for the capacity settings h_h, m_h, l_h, l_l. SP can solve all of them but one instance with m_h setting, for which a feasible solution is found (TL-IS), and two with l_l, for which not even a feasible solution can be found (TL).

Increasing the network size and the number of demands increases the computational times, but SP proves to degrade significantly less than PR. As observed for the small-medium size instances, high link capacity instances are easier than low link capacity ones. The computational times of SP for the high link capacity cases are acceptable, spreading from around 40 seconds up to about 24 minutes, with the only exception of two instances within the m_h capacity case: giul39, (where optimality cannot be proved) and janos-us-ca (where the computational time is a bit larger than one hour). When services and links have both low capacity, finding an optimal solution almost always requires around 50-55 minutes (germany50 is solved in less than 5 minutes).

In general, when SP can find a feasible integer solution it can also prove its optimality (a similar behavior has been observed for the small-medium size instances). In the m_h giul39 instance, the only one where a feasible, but not optimal, solution is provided, the gap from the best bound is significant.

4.3. Impact of Property 2.1 and scalability with respect to the number of services

In order to assess the impact of Property 2.1 we have generated, for three topologies, abilene, atlanta and dfn-bwin, instances with three types of VNF service instead of one. Table 11 reports a comparison of the computational time for the two cases (one VNF type and three VNF types). For each instance, the variation of the computational time (in percentage) due to the increase of the number of services is reported. It is computed as $\frac{CPUtime_3 - CPUtime_1}{CPUtime_1}$, where $CPUtime_\kappa$ is the computational time with κ services (a negative increase represents a reduction of the computational time when tackling the three service types case).

For the PR formulation passing from one service to three services seems to be always more challenging than for the SP formulation from a computational point of view. We denote with TL the cases where the PR formulation cannot find any solution for the three case services, a star denotes the cases where even in the one case service only a feasible solution can be found. On one instance (dfn-bwin with capacities m_l) passing

Capacity	Instance	Best SP	noVI	VI1	VI2	VI1+2
h_h	di-yuan	0.10	**0.00**	0.11	0.26	0.03
	pdh	0.09	**0.00**	0.55	1.35	0.51
	polska	0.20	0.09	0.01	0.44	**0.00**
	sun	0.43	**0.00**	0.23	0.99	0.98
	dfn-bwin	0.37	**0.00**	0.05	0.44	0.50
	nobel-us	0.52	0.06	**0.00**	0.12	0.20
	nobel-germany	0.39	**0.00**	0.21	1.21	1.09
	abilene	0.41	0.09	0.17	0.13	**0.00**
	atlanta	0.90	**0.00**	0.19	1.59	1.63
	newyork	1.49	**0.00**	0.03	1.84	1.87
	france	1.67	**0.00**	0.09	0.35	0.45
	nobel-eu	3.31	**0.00**	0.07	2.57	2.70
	ta1	3.64	0.01	**0.00**	0.28	0.29
	geant	2.31	**0.00**	0.16	1.13	1.24
	janos-us	13.65	**0.00**	**0.00**	0.72	0.82
	norway	9.75	**0.00**	0.01	3.70	4.03
m_h	di-yuan	0.09	0.14	0.38	0.54	**0.00**
	pdh	0.09	0.20	2.46	0.32	**0.00**
	polska	0.19	11.08	3.72	0.15	**0.00**
	sun	0.91	8.74	2.07	0.07	**0.00**
	dfn-bwin	0.34	5.54	0.65	1.40	**0.00**
	nobel-us	0.60	3.53	2.33	**0.00**	0.24
	nobel-germany	0.69	9.61	3.84	1.69	**0.00**
	abilene	0.60	1.79	0.48	0.15	**0.00**
	atlanta	1.49	7.05	1.17	0.22	**0.00**
	newyork	4.18	3.90	2.40	0.16	**0.00**
	france	2.20	0.28	**0.00**	0.06	0.11
	nobel-eu	7.46	77.51	7.87	**0.00**	0.37
	ta1	3.68	101.86	13.48	0.09	**0.00**
	geant	8.73	47.80	9.91	0.01	**0.00**
	janos-us	41.94	18.46	8.85	**0.00**	0.07
	norway	89.26	36.52	3.06	7.66	**0.00**
l_h	di-yuan	0.15	0.78	**0.00**	0.10	0.09
	pdh	0.11	1.30	0.60	0.02	**0.00**
	polska	0.23	4.07	3.93	0.16	**0.00**
	sun	3.18	0.37	0.77	**0.00**	0.33
	dfn-bwin	0.48	1.64	0.98	0.37	**0.00**
	nobel-us	0.52	3.18	1.77	0.39	**0.00**
	nobel-germany	0.73	6.41	0.87	0.93	**0.00**
	abilene	0.50	3.26	2.49	**0.00**	0.54
	atlanta	1.33	9.21	4.58	0.90	**0.00**
	newyork	1.92	13.39	4.47	0.33	**0.00**
	france	6.23	52.86	12.80	3.60	**0.00**
	nobel-eu	8.17	17.57	4.07	0.27	**0.00**
	ta1	6.44	51.31	6.66	0.47	**0.00**
	geant	5.10	36.28	10.27	1.15	**0.00**
	janos-us	18.23	16.58	10.81	0.03	**0.00**
	norway	23.42	20.99	14.88	**0.00**	0.16

Table 8: Variation of computational time with respect to best VIs choice - part 1 - high link capacity

Capacity	Instance	Best SP	noVI	VI1	VI2	VI1+2
h_l	di-yuan	0.76	3.78	**0.00**	2.97	1.05
	pdh	2.79	18.19	1.61	2.42	**0.00**
	polska	435.22	TL	2.85	3.3	**0.00**
	sun	63.72	8.02	1.34	**0.00**	1.6
	dfn-bwin	2.94	1.67	0.67	0.33	**0.00**
	nobel-us	478.86	0.36	5.4	0.17	**0.00**
	nobel-germany	9.84	**0.00**	0.35	0.25	0.18
	abilene	2.37	0.24	0.01	3.75	**0.00**
	atlanta	1065.05	0.27	0.38	**0.00**	0.19
	newyork	TL	TL	TL	TL	TL
	france	TL	TL	TL	TL	TL
	nobel-eu	432.83	2.21	1.9	**0.00**	1.87
	ta1	187.86	0.85	2.48	**0.00**	2.33
	geant	142.74	13.45	**0.00**	2.27	13.48
	janos-us	2440.33	TL	TL	**0.00**	TL
	norway	TL	TL	TL	TL	TL
m_l	di-yuan	0.35	2.64	1.74	**0.00**	0.16
	pdh	1.33	5.92	**0.00**	0.34	1.79
	polska	98.69	**0.00**	32.16	TL	2.73
	sun	24.11	2.04	**0.00**	3.16	6.07
	dfn-bwin	1.21	8.34	0.7	**0.00**	0.2
	nobel-us	116.59	4.67	15.97	**0.00**	4.74
	nobel-germany	6.73	0.52	1.26	**0.00**	0.39
	abilene	1.81	2.95	9.25	0.83	**0.00**
	atlanta	663.4	1.05	1.51	1.78	**0.00**
	newyork	TL	TL	TL	TL	TL
	france	TL	TL	TL	TL	TL
	nobel-eu	322.68	6.58	**0.00**	0.91	1.51
	ta1	8.58	358.34	4.76	0.55	**0.00**
	geant	161.785	TL	2.63	**0.00**	16.54
	janos-us	3600	TL	TL	TL	TL
	norway	3600	TL	TL	TL	TL
l_l	di-yuan	0.27	1.13	0.51	**0.00**	1.23
	pdh	0.38	2.45	0.17	**0.00**	0.38
	polska	193.057	TL	TL	**0.00**	0.65
	sun	8.59	11.09	1.01	0.1	**0.00**
	dfn-bwin	0.76	12.09	1.54	0.29	**0.00**
	nobel-us	8.5	58.52	63.82	**0.00**	0.17
	nobel-germany	1.97	12.53	8.11	**0.00**	1.6
	abilene	2.12	10.3	15.21	0.25	**0.00**
	atlanta	11.98	5.27	149.14	6.09	**0.00**
	newyork	1409.36	TL	TL	TL	**0.00**
	france	TL	TL	TL	TL	TL
	nobel-eu	27.21	51.01	48.7	0.59	**0.00**
	ta1	12.13	110.14	68.34	0.14	**0.00**
	geant	29	40.87	85.27	**0.00**	0.15
	janos-us	81.5332	TL	TL	**0.00**	0.12
	norway	251.317	TL	TL	**0.00**	0.53

Table 9: Variation of computational time with respect to best VIs choice - part 2 - low link capacity

Instance	h_h	m_h	l_h	l_l
cost266	501.82	333.79	182.8	TL
germany50	151.97	694.57	70.39	265.93
giul39	975.67	TL-IS	415.86	2989.96
india35	119.65	65.14	39.62	3286.9
janos-us-ca	885.48	3718.73	610.05	TL
pioro40	302.33	1434.99	83.22	2950.12

Table 10: Computational times of SP formulation for large size instances (TL-IS means that the time limit has been reached and a feasible solution is found, while TL means that not even a feasible solution has been found within the time limit). PR cannot provide any feasible solution. The time limit is set to 7200s.

from one service to three services lead to the loss the optimality (TL-IS). Instances where even with one service no feasible solution can be found are denoted with a "-".

Results show that the property is far from irrelevant, as it allows us, in general, to dramatically reduce the computational time. Although for a few instances the computational times of SP formulation may reduce, in general, if the property is neglected the computational times increase up to two orders of magnitude as far as SP is concerned, and up to 5 orders of magnitude as far as PR is concerned. Further, if the property is not applied, the number of instances that PR can solve reduces significantly: 7 out of 15 instances (almost half of them) that PR can solve to optimality with one service type cannot be solved with three service types (TL and TL-IS) and for all but one no feasible solution can be found with three service types. For further three instances, PR cannot find a feasible solution when the number of service types increases.

It is worth noting that, even if the property is neglected, SP degrades significantly less for the increase of the number of services, although in the SP formulation the number of variables depends on the number of service types in each chain.

Formulation	Instance	h_h	h_l	m_h	m_l	l_h	l_l
SP	abilene	**82**	**138**	**80**	**415**	**38**	-52
	atlanta	**136**	**147**	**337**	-17	**56**	-89
	dfn-bwin	**226**	-1	**174**	**91**	**68**	**94**
PR	abilene	**7027**	**TL**	**6800**	**TL**	**12159**	**145558**
	atlanta	**TL**	-	**TL***	-	**TL**	**TL**
	dfn-bwin	**39**	**TL***	**TL-IS**	-	**4455**	**TL**

Table 11: Time increase (in percentage) due to the increase of the number of VNF types (from 1 to 3) for different capacity cases (in bold the cases where the overall computational time increases). TL-IS denotes the instances where the PR formulation can find only a feasible solution for the three cases services, TL denotes the instances where the PR formulation cannot find any feasible solution with the three cases services.

4.4. Articulation point based preprocessing

We tested the articulation point preprocessing on three network topologies of SNDLib which contain at least one articulation point: france, ta2 and zib54. In Table 12, the

computational time for the SP formulation with VI1+2 and the computational time variation (in percentage) that is obtained adding the preprocessing (taking into account also the time needed for the preprocessing) are reported for the capacity cases where an optimal solution can be found for all the three instances (results are not reported for the PR, as it fails in solving almost all the instances). We highlighted in bold the cases where adding the preprocessing procedure reduces the overall computational time.

The preprocessing is quite fast (few seconds) and its addition reduces almost always the computational time: thanks to the preprocessing SP solves to optimality, despite their big size, instances for which no feasible solution can be found without preprocessing (zib54 for the l_l capacity case). Nevertheless, in some capacity cases (all the h_l and m_l cases) no feasible solution can be found even using the preprocessing.

The preprocessing is not enough to allow PR to find a solution in most of the cases (the only exception being the france network, where the solution can be certified for the uncapacitated case).

To conclude, as it needs negligible computational time and is beneficial for some instances, although not for all, we believe that the articulation point based preprocessing is worth performing.

Instance	h_h		m_h		l_h	
	time (s)	time red.	time (s)	time red.	time (s)	time red.
france	1.67	**-32.0**	2.20	**-26.0**	6.23	153.0
ta2	261.31	**-73.0**	1095.84	**-93.0**	716.57	**-6.0**
zib54	136.78	**-37.0**	543.45	**-84.0**	306.45	**-36.0**

Table 12: For each instance and capacity case, the solution time (in seconds) of the SP formulation with VI1+2 and the time reduction (in percentage) obtained by using articulation point preprocessing are reported. The cases where the preprocessing allows us to obtain a time reduction of the overall optimization procedure (taking into account also the preprocessing time) are highlighted in bold.

5. Conclusions

Despite the large number of recent papers devoted to the Virtual Network Functions chaining problem, it is difficult to find a comparison among the proposed approaches, as many different versions of the problem have been considered and the proposed approaches are tailored to them. Therefore, our goal was to analyze the most promising formulation strategies on a common test bed. In order to do so, we worked on a version of the problem where each demand asks for the same service and must be routed on a simple path. We studied some problem properties and compare both theoretically and computationally two formulations: the first formulation (SP) is based on splitting each demand path into two subpaths, one connecting the source with the service, the second one connecting the service with the destination. The second one (PR) uses arc flow variables and forbids cycles exploiting node labels.

As for the problem properties, we proved that:

- the single service case is equivalent to the multiple services one, when the services have the same capacity and the sequence of services is the same for all the demands;

- an instance of the service must be installed on the articulation points that belong to biconnected components with internal demands.

As for the comparison between the two formulations, we proved that the continuous relaxation of SP always provides a bound not worse than the one of PR, which is confirmed by the computational results. SP outperforms PR as far as the scalability is concerned, as well, as it can solve instances with up to about 60 nodes and 1800 demands, if the link capacity is high. Instead, PR degrades significantly with the increasing instance size: it cannot deal with instances with about 30 nodes and 700 demands. Both the theoretical properties turn out to be effective also from the computational point of view, as they help in speeding up the computational time.

We are currently extending the formulations to address more general versions of the problem: the multi-service and multi-sequence case and the capacitated node one.

Furthermore, as computational results have shown that, in general, SP either is able to find an optimal solution or it is not even able to provide a feasible solution, the problem of finding a feasible solution is worth investigating. Therefore, we are currently developing ILP based heuristics that should benefit from the choice of SP formulation, which proved to be the more effective.

References

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

[2] B. Addis, G. Carello, and M. Gao, *On the complexity of a virtual network function placement and routing problem*, ENDM - Alio Euro 2018 special issue, 2018.

[3] 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 **71** (2018), 97–106.

[4] M.F. Bari, S.R. Chowdhury, R. Ahmed, and R. Boutaba, *On orchestrating virtual network functions*, 2015 11th International Conference on Network and Service Management (CNSM), Nov 2015, pp. 50–56.

[5] M. Bouet, J. Leguay, and V. Conan, *Cost-based placement of vDPI functions in NFV infrastructures*, Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), April 2015, pp. 1–9.

[6] I. Contreras and E. Fernández, *General network design: A unified view of combined location and network design problems*, Eur. J. Oper. Res. **219** (2012), 680 – 697.

[7] M. Gao, Models and Methods for Network Function Virtualization (NFV) Architectures, PhD thesis, Université de Lorraine, 2019.

[8] M. Gao, B. Addis, M. Bouet, and S. Secci, *Optimal orchestration of virtual network functions*, Comput. Networks **142** (2018), 108 – 127.

[9] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, *A novel approach to virtual networks embedding for SDN management and orchestration*, 2014 IEEE Network Operations and Management Symposium (NOMS), May 2014, pp. 1–7.

[10] J. Hopcroft and R. Tarjan, *Algorithm 447: Efficient algorithms for graph manipulation*, Commun. ACM **16** (June 1973), 372–378.

[11] J. Hwang, K.K. Ramakrishnan, and T. Wood, *NetVM: High performance and flexible networking using virtualization on commodity platforms*, IEEE Trans. Network Service Manage. **12** (March 2015), 34–47.

[12] 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 International Symposium on Integrated Network Management (IM), May 2015, pp. 98–106.

[13] M.C. Luizelli, W.L. da Costa Cordeiro, L.S. Buriol, and L.P. Gaspary, *A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining*, Comput. Commun. **102** (2017), 67 – 77.

[14] T. Lukovszki and S. Schmid, *Online admission control and embedding of service chains*, Post-Proceedings of the 22nd International Colloquium on Structural Information and Communication Complexity - Volume 9439, Springer-Verlag, New York, NY, USA, 2015, SIROCCO 2015 pp. 104–118.

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

[16] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, *SNDlib 1.0 – Survivable Network Design library*, Networks **55** (May 2010), 276–286.

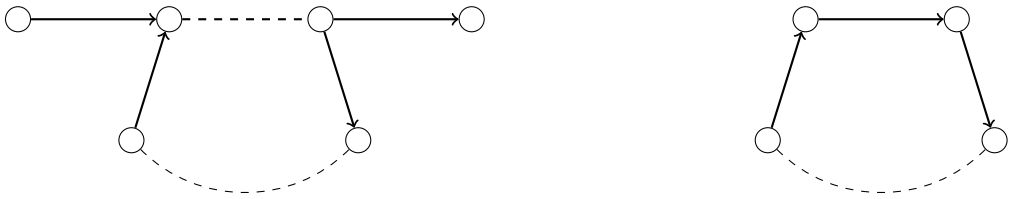
[17] N.W. Paper, *Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges Call for Action. http://portal.etsi.org/NFV/NFV_White_Paper.pdf*, Oct. 2012.

[18] J. Soares, M. Dias, J. Carapinha, B. Parreira, and S. Sargento, *Cloud4NFV: A platform for virtual network functions*, 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), Oct 2014, pp. 288–293.

Appendix A. Additional proofs and properties

In what follows, we report the full proofs that we sketched in Section 3.1. This material is intended as a complement to the paper and not as stand-alone, therefore we refer to the notation, equations and properties presented before. Nevertheless, for the sake of clarity, we repeat some definitions and explanations.

Let us considered an instance of the VNF-PR$_{\text{SP}}$ problem. For any demand k, the flow in a feasible continuous solution can be divided into flow on simple paths and flow on cycles. Let us denote with P_k the set of simple paths and with C_k the set of cycles. We can distinguish between two type of cycles (see Figure A.10): cycles sharing some nodes with a simple path (Figure A.10a) and isolated cycles (Figure A.10b).



(a) Cycle sharing nodes with a simple path

(b) Isolated cycle

Figure A.10: Examples of cycle sharing nodes and of isolated cycle

As we already observed (see Remark 3.1), the SP formulation forbids demands to be served, even partially, by a VNF instance installed on an isolated cycle[11]. For example, let us consider an isolated cycle of two nodes A and B such that the demand is partially served by a node A ($z_A^k = \alpha, \alpha < 1$). Summing up flow balancing constraints (7) leads to an inconsistency $0 = -\alpha$ (analogously for subpath 2).
As a consequence, only three cases occur, as detailed in the following remark.

[11] Instead PR accepts solutions with an isolated cycle and a partial service installed on it, but routing variable values (on the cycle) cannot be greater than $\frac{|N|}{m+|N|}$ (where m is the length of the cycle) due to the isolated loop elimination constraints (13).

Remark A1. Consider a feasible solution of the continuous relaxation of SP (x^1, x^2, y, z), a demand k and a cycle c induced by such solution. Let us denote with N_c the set of nodes of the cycle c that are not shared with any simple path: $N_c = \{i \in C \setminus P_k\}$ where C denotes the set of the nodes belonging to cycle c. Due to flow balance constraints (7)-(8), we have only three possible cases for all nodes belonging to N_c:

1. if a (partial) service instance is located on any node belonging to N_c, the cycle is induced by variables of both semi-paths, and $x_i^{k1} = x_i^{k2}$
2. if no service is located on any node belonging to N_c, then
   2.1) the cycle is induced by only one type of semi-path variables
   2.2) the cycle can be decomposed into two super-imposed cycles, each of them generated by only one type of semi-path variables.

Thus we can observe that:

Property A1. (it corresponds to Remark 3.2).
Let us consider a feasible solution (x^1, x^2, y, z), a demand k and the path-cycle decomposition $\{P_k, C_k\}$ of its routing. Let us suppose that a cycle $c \in C_k$ exists that shares at least one node with a simple path $p \in P_k$.
If a (partial) service is located on a node belonging to a cycle $c \in C_k$, but not to the path $(\{i \in c \setminus P_k\})$, the routing variables inducing the path and the cycle are fractional and their value is less than or equal to $\frac{1}{2}$.

The proof follows directly from constraints (7)-(8) and (9)-(10).

Property A2. Any solution such that no service instance is installed on a node belonging to a cycle but not to a simple path can be transformed into an equivalent solution (in term of cost, location and assignment) removing the cycles without service instances installed and the corresponding flow from the routing.

The proof follows from flow balancing constraints as in classical flow problems.

Theorem A1. Any solution of the continuous relaxation of SP such that no service instance is installed on a node belonging to a cycle but not to a simple path can be mapped into an equivalent solution of PR in terms of routing, location and assignment and therefore cost.

Proof. Opening variables y_i and assignment variables z_i^k, the objective function and constraints (2), (3) and (4) are common to both formulations. SP routing variables can be easily mapped into PR ones as follows:

$$x_{ij}^k = x_{ij}^{k1} + x_{ij}^{k2} \tag{A.1}$$

Thanks to constraints (9) and (10) the mapping guarantees also that $x_{ij}^k \in [0, 1]$. Further, link capacity constraints of SP (5) imply the link capacity constraint of PR (6). Routing constraints of the SP formulation imply the routing constraints for the PR formulation. In fact, by summing equations (7)-(8) and using the mapping (A.1), we obtain constraints (11).

Now we show that SP routing constraints for the semi-path (7) imply routing-location connecting constraints of PR (12).
Let us consider the case $i \in N, i \neq o_k$:

$$\sum_{j:(i,j)\in A} x_{ij}^{k1} - \sum_{j:(j,i)\in A} x_{ji}^{k1} \;=\; -z_i^k$$

reordering terms, we obtain:

$$\sum_{j:(i,j)\in A} x_{ij}^{k1} + \; z_i^k \;=\; \sum_{j:(j,i)\in A} x_{ji}^{k1}$$

As $\sum_{j:(i,j)\in A} x_{ij}^{k1} \geq 0$, we can derive:

$$z_i^k \;\leq \sum_{j:(j,i)\in A} x_{ji}^{k1}$$

Adding $\sum_{j:(i,j)\in A} x_{ij}^{k2}$ at the right side, as this term is always not negative, the inequality still holds:

$$z_i^k \leq \sum_{j:(j,i)\in A} (x_{ji}^{k1} + x_{ij}^{k2})$$

Then, using the routing variables mapping (A.1), we verify constraint (12):

$$z_i^k \;\leq \sum_{j:(j,i)\in A} x_{ji}^{k}.$$

Finally, suitable values for π variables must be derived. Let us consider a demand k and its routing. We can build an induced graph as follows:

$$G^k(N_k, A_k)$$

where $(i,j) \in A_k$ if $x_{ij}^{k1} + x_{ij}^{k2} > 0$ and $\exists p \in P_k : (i,j) \in p$, i.e., the arc belongs to at least a simple path. A node $\hat{\imath}$ belongs to N_k if there exists an arc $((\hat{\imath}, j)$ or $(j, \hat{\imath}))$ in A_k. For any arc $(i,j) \in G^k$ we define the following cost $c_{ij} = x_{ij}^{k1} + x_{ij}^{k2} = x_{ij}^k$.
As G^k does not contain arcs that belong only to cycles in C^k, the obtained graph is acyclic, thus we can define π_j^k as the longest path from node o_k to node j:

- $\pi_{o_k}^k = 0$
- $\pi_j^k = \max_{i:(i,\hat{\jmath})\in A^k}\{\pi_i^k + x_{ij}^k\}$

Such values obviously satisfy constraints (13) for the nodes belonging to the path.

The nodes that belong to a cycle and to a simple path (nodes from ℓ to m in Figure A.11) have already been assigned a suitable value π. We now need to determine a value of π for the nodes on the cycle N_C that have been removed.

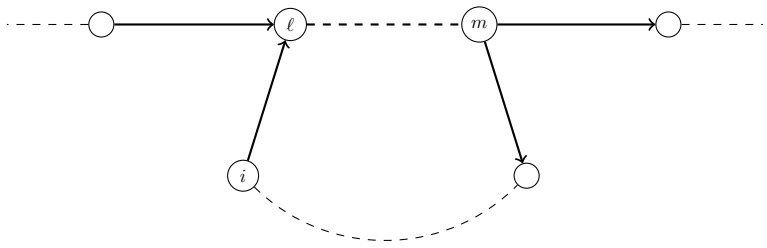


Figure A.11: An example of a cycle sharing some nodes with a simple path

Let us consider the two nodes ℓ and m, corresponding to the smallest (π^k_ℓ) and largest (π^k_m) value of variables π on the cycle N_C, respectively. For all the nodes $i \in N_c$, we impose:

$$\pi^k_i = \frac{\pi^k_m + \pi^k_\ell}{2}$$

Now, we prove that such a value of π always satisfy constraints (13).
We can observe that $x_{ij} \leq 1$ in any feasible solution, therefore the values of π are bounded by $|N| - 1$. As for the arcs in $A \setminus A^k$, we have two cases.

1. Arcs belonging to the cycle but not a path.
   Due to Property A1, we can infer that for any arc belonging to the cycle but not the path, we have $x^k_{ij} \leq \frac{1}{2}$. Therefore, the term:

   $$x^k_{ij} - |N| \ (1 - x^k_{ij})$$

   is always non positive as $|N| \geq 2$. Therefore for any two nodes in the cycle, as they have the same value of π, the constraint is valid.

2. Arcs belonging to the cycle and incident both in the path and the cycle, i.e., with one extreme in ℓ or m.
   Let consider the arc of the cycle entering node ℓ: (i, ℓ) (see Figure 2), we need to prove that:

   $$\pi^k_\ell \geq \pi^k_i + x^k_{i\ell} - |N|(1 - x^k_{i\ell}), \tag{A.2}$$

   Let us denote with n the number of arcs in the path between node ℓ and node m ($n \leq |N| - 1$), and with $\beta \leq 1$ the value of the associated routing variable on the path, then we have:

   $$\pi^k_m = \pi^k_\ell + \beta n$$

   and we obtain:

   $$\pi^k_i = \frac{\pi^k_m + \pi^k_\ell}{2} = \pi^k_\ell + \frac{\beta n}{2} \leq \pi^k_\ell + \frac{(|N| - 1)}{2}$$

   Thus, denoting with γ the right-hand side of equation (A.2), we get:

   $$\gamma \leq \pi^k_\ell + \frac{(|N| - 1)}{2} + x^k_{i\ell} - |N|(1 - x^k_{i\ell})$$

and, rearranging the terms:

$$\gamma \leq \pi_\ell^k + \left(x_{i\ell}^k - \frac{1}{2}\right)(|N|+1)$$

and using $x_{i\ell}^k \leq \frac{1}{2}$ (see Property A1):

$$\pi_\ell^k \geq \gamma$$

A similar argument can be used to prove that the constraint is valid for the link belonging to the cycle and exiting the other extreme node m.

■

Appendix B. Extending the SP formulation for the multiple services case

If we want to consider multiple services and that each demand can ask for a different number of services (among the available ones), we need to introduce the following notation:

- F: set of VNFs types
- n_k: number of services asked by demand k
- VNF_f^k: indicator parameters, equal to 1 if demand k asks for service of type f

Furthermore, if the chain order is given, we need to define:

- $f^k(s) : \{1, ..., n_k\} \rightarrow F$: integer indicator map, type of service at position s for demand k, 0 if the service is not requested

Decision variables must be extended accordingly:

- $y_{if} \in \{0, 1\}$ if service type $f \in F$ is located on node $i \in N$
- $z_{if}^k \in \{0, 1\}$ if demand $k \in D$ uses service type $f \in F$ on node $i \in N$
- $x_{ij}^{ks} \in \{0, 1\}$ if arc $(i, j) \in A$ is used by demand $k \in D$, subpath $s \in \{1, ..., n_k + 1\}$

The resulting SP extended formulation is:

$$\min \sum_{i \in N} \sum_{f \in F} y_{if}$$

$$\sum_{i \in N} z^k_{if} \;=\; 1 \qquad \forall k \in D,\ f \in F : VNF^k_f = 1$$

$$z^k_{if} \;\leq\; y_{if} \qquad \forall k \in D,\ i \in N,\ f \in F$$

$$\sum_{k \in D} \sum_{s \in 1 \ldots n_k+1} d_k\ x^{ks}_{ij} \;\leq\; u \qquad \forall (i,j) \in A$$

$$\sum_{k \in D : VNF^k_f = 1} d_k\ z^k_{if} \;\leq\; q_f \qquad \forall i \in N,\ f \in F$$

$$\sum_{j:(i,j) \in A} x^{k,s}_{ij} - \sum_{j:(j,i) \in A} x^{k,s}_{ji} \;= z^k_{i,f^k(s-1)} \;-\; z^k_{i,f^k(s)} \qquad \forall k \in D,\ i \in N,\ s \in 2..n_k$$

$$\sum_{j:(i,j) \in A} x^{k,1}_{ij} - \sum_{j:(j,i) \in A} x^{k,1}_{ji} = \begin{cases} 1 - z^k_{i,f^k(1)} & \text{if } i = o_k \\ -z^k_{i,f^k(1)} & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N$$

$$\sum_{j:(i,j) \in A} x^{k,n_k+1}_{ij} - \sum_{j:(j,i) \in A} x^{k,n_k+1}_{ji} \;=\; \begin{cases} z^k_{i,f^k(n_k)} - 1 & \text{if } i = t_k \\ z^k_{i,f^k(n_k)} & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N$$

$$\sum_{s \in 1 \ldots n_k+1} \sum_{l:(j,i) \in A} x^{ks}_{ji} \leq 1 \qquad \forall k \in D, i \in N$$

$$\sum_{s \in 1 \ldots n_k+1} \sum_{j:(i,j) \in A} x^{ks}_{ij} \leq 1 \qquad \forall k \in D, i \in N$$

To extend the formulation for the case with unordered services, it is necessary to decouple the subpaths description and the services assignment to nodes (originally both represented by variable z). We keep variables z to represent the location of services, while a new variable w will be used to describe the flow balance for subpaths:

- $w^{ks}_i \in \{0,1\}$ if demand $k \in D$ uses the s-th service on node $i \in N$

With respect to already presented constraints, only the flow balancing constraints are affected by this change:

$$\sum_{j:(i,j) \in A} x^{k,s}_{ij} - \sum_{j:(j,i) \in A} x^{k,s}_{ji} \;= w^{k,s-1}_i \;-\; w^{k,s}_i \qquad \forall k \in D,\ i \in N,\ s \in 2..n_k$$

$$\sum_{j:(i,j) \in A} x^{k,1}_{ij} - \sum_{j:(j,i) \in A} x^{k,1}_{ji} = \begin{cases} 1 - w^{k1}_i & \text{if } i = o_k \\ -w^{k1}_i & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N$$

$$\sum_{j:(i,j) \in A} x^{k,n_k+1}_{ij} - \sum_{j:(j,i) \in A} x^{k,n_k+1}_{ji} \;=\; \begin{cases} w^{k,n_k}_i - 1 & \text{if } i = t_k \\ w^{k,n_k}_i & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N$$

Additional consistency constraints must be added to link z and w variables:

$$\sum_{f \in F : VNF^k_f = 1} z^k_{if} \;=\; \sum_{s \in 1 \ldots n_k+1} w^{k,s}_i \qquad \forall k \in D,\ i \in N$$