On a Virtual Network Function Placement and Routing problem: properties and formulations

Bernardetta Addis, G. Carello, F de Bettin, Meihui Gao

▶ **To cite this version:**

Bernardetta Addis, G. Carello, F de Bettin, Meihui Gao. On a Virtual Network Function Placement and Routing problem: properties and formulations. 2017. halshs-01643064v1

HAL Id: halshs-01643064

https://shs.hal.science/halshs-01643064v1

Preprint submitted on 21 Nov 2017 (v1), last revised 23 Nov 2018 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On a Virtual Network Function Placement and Routing problem: properties and formulations

B. Addis, G. Carello, F. De Bettin, M. Gao

Abstract

Abstract

1 Introduction

The mass diffusion of telecommunication networks and the possibility of accessing email, social networking or cloud computing from mobile devices is exponentially increasing the demand for network services. Network services were up to now provided by proprietary network appliances, such as firewalls, proxies or WAN optimizers, but it has become difficult to integrate and operate such hardware based appliances or to add new functionalities to keep up with the ever increasing demand at a reasonable cost. To cope with this challenge, Network Functions Virtualization (NFV) paradigm has been recently proposed: hardware based appliances should be replaced by Virtual Network Functions (VNFs) running on generic servers, which will provide the required services in a cheaper and more flexible way.

A key problem to implement the NVF paradigm is known as VNF chaining problem: locating VNFs and routing demands so as to guarantee that each demand passes through a sequence (chain) of VNFs that provides the services it needs. The allocation and routing must satisfy certain 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 be connected to computing servers (and/or clouds). Each server is connected to one node, but in general a node can or cannot be connected to a server. Traffic demands must be routed on the network and be served by a set of network functions (services). For each demand, the network functions must be traversed with a given (possibly partial) order or without any fixed order. Network functions can be located on computing servers/cloud. Therefore a traffic demand that needs to access a network function located in a given server must pass through the node that is directly connected to the computing server. Several constraints can be taken into account, such as the service order constraint, the VNF capacity constraint, and link capacity constraint, thus leading to different versions of the problem.

Different versions of the problem have been addressed in the telecommunication literature, however the focus is mainly on the application and to the best of our knowledge the computational complexity have not been studied thoroughly. Although different mathematical programming formulations have been

proposed for the problem, they have not been compared. From an optimization point of view, 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, but the combination described by the VNF chaining problem represents a new challenge. The goal of the paper is to study the complexity properties of a particular version of the problem and to compare two formulations derived from the formulation strategies proposed in the literature. Further, additional inequalities to enrich such formulations are proposed and tested.

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/cloud is connected directly to only one routing node, we use a compact representation where nodes and server node are collapsed, simply assigning the computational capacity of a server to the associated node. In the resulting network, we have two type of nodes: the ones that can host a virtual function and the one that can not. VNF 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. The goal is to minimize the number of needed VNF.

This paper is organized as follows. In Section 2, we present the related work. In Section 3, we formally define the problem and investigate the problem properties. In Section 4, the problem complexity is reported. In Section 5, we introduce two problem formulations and describe the features of each of them and their relation. We present and analyze the results of computational experiments in Section 6. Section 7 concludes the paper.

2 Related work

NFV technology has gained significant attention from both industry and academia, and evidence of this is the significant number of works on NFV research addressing challenging aspects from the creation of NFV platform [15],[9] to the optimization of VNF chaining with respect to, for instance, resource efficiency [10] or competitive goals [1]. We review in this section some of the most outstanding research works related to the VNF chaining problem. We classify the existing strategies carried out to solve the VNF chaining problem into two broad categories depending on the underlying mathematical modeling approach: the Virtual Network Embedding (VNE) [6] based approaches and the combined facility location and network design [5] based approaches. In the following, we discuss each category starting from the preliminary studies to the recent efforts.

Preliminary works on the optimization of VNF chains deployment tend to solve the VNF chaining problem as a VNE problem, which treats service requests as virtual network requests to be mapped to the substrate network. In [8], authors introduce a MIP formulation for embedding VNFs and virtual links into the underlying network infrastructure. Further, authors in [3] provides an VNE based ILP formulation along with a dynamic programming based heuristic to deal with large instances of problem. Similarly, authors in [13] focus on the online version of VNF chaining problem. In [13], three greedy solutions and

a tabu-search based heuristic method are proposed to deal with VNF online mapping and scheduling. The problem of embedding VNF chains can also be found in [11], which attends to the problem of admitting and embedding a maximum number of service chains. In [10] the authors, even without namely it explicitly, propose an ILP model based on the mapping of VNF chains on a physical network. Precomputed paths for mapping are used. The ILP model is used as a basic step in a heuristic procedure based on a dichotomic search on the number of located VNFs.

In spite of their potentialities, the investigations mentioned above are all tailored to the fully determined order of VNFs and cannot be applied when there exist just a partial (or any) order of VNFs in the service chain. Moreover, from the optimization perspective, the VNF chaining problem differs in some aspects from the general VNE problem. In VNE, virtual network nodes and virtual network links need to be mapped to the substrate physical network. While in VNF chaining, the VNFs are located at potential physical nodes, and the service demands must be assigned to their required VNF instances and routed to pass thought the located VNFs. If we consider this description, VNE is closer to a mapping problem. It is true, that VNF chaining can be viewed as a "special case" of VNE where the virtual networks to be embedded reduce to linear graphs and both extreme nodes of any virtual graph have a fixed location. As we already pointed out, this parallel is possible only if the order of VNFs in the chain is fixed, otherwise the VNE model cannot be used for representing the VNF chaining problem. Therefore, in our opinion, the VNF chaining problem is closer to a combination of a facility location and a network design problems. We will show in Section 3, where we present results on computational complexity, how these two aspects influence the properties of the problem.

This second modeling perspective is adopted by a handful of papers in the networking literature. Authors in [12] define a model formalizing the VNF chaining problem using a context-free language and propose to use Mixed Integer Quadratically Constrained Program (MIQCP) for finding the optimal placement of VNFs and chaining them together. In [4], the specific Deep Packet Inspection (DPI) VNF placement problem is targeted and an adaptation of the multicommodity model is proposed to model the problem (a single type of services is asked). Small instances are solved with a standard ILP solver (glpk) and for larger instances a centrality-based greedy algorithm is proposed: at each step a new VNF (vDPI) is located in the node that has the highest centrality until all the traffic flows are served or all nodes have a vDPI. In [1], authors provide a MILP formulation considering facility location and demand routing for a generic number of services and no fixed order in the chain, taking into account different latency regimes and the traffic compression properties. Their work investigates the trade-off achievable between legacy traffic engineering (TE) ISP goals and novel combined TE-NFV goals. An extension of the work [7] presents a model that allows to take into account also ordered (or partially ordered) chains. A math-heuristic is presented to speed up the solution phase and a numerical comparison with a VNE model approach is proposed. A model similar to [1] is proposed in [2], where additional constraints are added to take into account the possibility to refuse the VNF assignment (and routing) to certain demands and the incompatibility of certain VNFs (and thus imposing their location in different nodes). A greedy algorithm based on the decomposition of the problem in two steps (routing and location) is proposed. To solve efficiently

the problem, authors propose a greedy algorithm (it allocates the flows one by one while instantiating VNFs on-the-fly) and a heuristic approach based on the LP-relaxation of the original ILP (based on rounding the optimal solution of the relaxed integer problem for each flow and deploying the VNFs with respect to the selected paths).

Due to the NP-hardness nature of the VNF chaining problem, most of the works focus on developing heuristic based methods to solve the problem within a reasonable computational time. Although the combined facility location and network design problem has been studied in optimization literature [5], the VNF chaining problem specificities are still not explored in the optimization literature. Compared to previous works, our work provides exact solutions to deal with the VNF chaining problem. To the best of our knowledge, except for our preliminary work, we are among the first to investigate in detail the computational complexity and properties of the VNF chaining problem. Last but not least, we consider the two different approaches presented in the literature (VNE and routing and location based) presenting two different formulations that are representative of each approach, we compare them theoretically and numerically. Furthermore, we develop valid inequalities and test them.

3 Problem description and properties

In this work we focus on a basic version of the VNF chaining problem where a single type of VNF is present and all server node can be equipped with a VNF. Links are capacitated and the cost of installing VNFs is minimized. Further, each demand must be served by the VNF and routed on a simple path. Let us denote the problem as *Network Function Virtualization Placement with Simple Path routing* (VNFP-SP).

The network is represented by a graph $G(N, A)$, where N represents the set of routing/server nodes and A represents the set of capacitated arcs (or links). Let u_{ij} denote the capacity of arc (i, j). An instance of the VNF can be installed on each node, has a fixed installation cost c and can serve a limited amount of demand q. The network demands is 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 required bandwidth d_k. The demand must be served (pass through) an instance of the VNF. Let recall that the nodes represent both routing nodes and servers, therefore, a demand can pass through a node without using a VNF installed on it[1]. 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.

As the cost is associated to services, and it is independent from the installation node, therefore, minimizing the total cost is equivalent to minimize the number of installed services.

3.1 General remarks

We assumed that a single VNF type is available. We now want to show that this assumption is not so restrictive. To this aim, let us considered a generalize problem where all the demands ask for a sequence of VNF (VNF chain). In the

[1]In this case the demand uses the routing node, but it does not use the server connected to it.

following, we prove that the problem where all the demands ask for the same sequence (same types of VNF and same order), in the case of VNFs uniform capacity and node capacity large enough to install an instance of each VNF type, is equivalent to the case with a single demand that we consider in this paper.

Property 3.1. Let us consider two problems $\mathcal{P}_1$ and $\mathcal{P}_2$ that are completely equal but for the cardinality of the required chain of services: in $\mathcal{P}_1$ each demand requires the same service, while in $\mathcal{P}_2$ each demand requires the same sequence of k services.

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 optimal sites selected by the solution of $\mathcal{P}_1$ and routing the demands according to the optimal solution of $\mathcal{P}_1$.

Proof. Let us consider the problem $\mathcal{P}_1$ composed of a set D of demands and a unique service type $\{a\}$. All the demands require the service type $\{a\}$. Let us suppose to find an optimal solution, that is to say a location solution for a service $\{a\}$ such that all the demands can be routed from their source node to their destination node passing through a service $\{a\}$, and respecting link and service capacity constraints.

Now, let us consider the problem $\mathcal{P}_2$ with the same data of $\mathcal{P}_1$ except the set of VNF types: T, s.t. $\{a\} \in T$. Let consider an optimal solution for $\mathcal{P}_1$. Installing an instance of each new service type $(T \setminus \{a\})$ on the same node where service $\{a\}$ is located, and assigning to them the same demands assigned to the instance of the service $\{a\}$, we obtain a feasible solution for $\mathcal{P}_2$. In fact, under these assumptions, adding a VNF type installation on a node already used, it does not affect neither the link capacity constraints or the routing demand constraints. Furthermore, as the VNF have the same capacity, if an installed service $\{a\}$ can serve the demands associated to it in a given node i, then the new added services $T \setminus \{a\}$ on node i, can serve the given demands as well.

As we observed, in the case of a single VNF type, the objective function can be reduced to minimize the number of installed services. We proved that any feasible solution of $P1$ can be extended to a feasible solution of $P2$ adding the additional services in the already used locations, therefore if the $P1$ solution is optimal, then it is optimal as well for $P2$.

■

We can observe that under the hypothesis that the set of required service types P is exactly the same, with or without order constraints, the two problem are equivalent [2]. So, we can get to the conclusion that the solution of two problem in which the same set P is required, independently from the order (no order, same order or different order), is the same.

Remark 3.1. Property 3.1 can be extended to the case where there exists a partition of demands that require disjoint set of VNFs. It is to say that if there exists a solution for the VNFP-SP where each disjoint subset of demands require

[2]We remind the reader that we assumed that for services allocated on the same sever it is always possible to schedule them in any order, and that in the application problem this order is determined by a suitable scheduling algorithm located at the server/cloud level

a different single VNF type, then it is a solution also for the problem in which each disjoint subset of demands require a disjoint subset of service types.

Proof. Similarly to the proof of the Property 3.1, let us consider two instances of the VNFP-SP.

The first one has a partitioned set of demand $D = D_1 \cup D_2$ where the first subset require the service type $\{a_1\}$ and the second one require the service type $\{a_2\}$.

The second instance is equal to the second one, but for the service types and the demand that request them. The first set of demand D_1 (D_2 respectively) requests a set of service type T_1 (T_2 respectively), such that $a_1 \in T_1$ ($a_2 \in T_2$ respectively) and such that $T_1 \cap T_2 = \emptyset$. Therefore, if a solution for the first instance is found, we can extend the solution to be feasible for the second problem: keep the same demand routing, the same installation and service assignment for a_1 and a_2, and installing services T_1 in the same location of a_1 and services T_2 in the same location as a_2.

■

3.2 Impact of biconnected components and articulation points

Given a graph that contains at least an *articulation point*, a lower bound on the number of VNF (and thus on the objective function) can be obtained. Let recall some definitions/properties necessary to prove our claim.

Definition 3.1. A graph is connected when there is a path between every pair of vertices.

Definition 3.2. A graph is biconnected if removing any node the graph remains connected.

Definition 3.3. Given a graph $G(N, A)$, a *biconnected component* (also known as a block or 2-connected component) is a maximal induced biconnected subgraph of G.

Biconnected components are connected to each other at shared vertexes called *cut vertexes* or *articulation points*.

Definition 3.4. A cut vertex or articulation point of a graph G is a vertex v such that $G \setminus v$ has more connected components than G.

Definition 3.5. A connected graph containing articulation points is said to be *separable*.

The following property can be stated.

Property 3.2. Let us consider a VNFP-SP defined on a separable graph G and for which for each biconnected component there exists at least one demand whose origin and destination nodes are in the biconnected component itself. Then, it is necessary to install a service inside each biconnected component. Furthermore, there exists an optimal solution where the service is located on the articulation points.

Proof. Let us suppose to install a service in node of a biconnected component that is not an articulation point. Then, a demand whose source node is in a

different biconnected component must pass through the connecting articulation point to reach the service. Since its destination node is in the same biconnected components of its source node, the demand must pass again through the same articulation point to complete its routing. But, passing twice through the same node, the demand routing violates the simple path constraint and thus it is not a feasible solution.

The only possible service locations to respect the simple path constraints are then inside the biconnected component where are located the demand source and destination. Since there is at least one demand for each biconnected component and the biconnected components have at most one node in common which is the articulation point, there exists an optimal solution where a service is installed on each articulation point.

■

Thanks to Property 3.2, if the number and structure of connected components is known, combining this information with the informations related to the source and destination of the demands, it is possible to have a lower bound of the number of VNFs to install, thus it is possible to determine a lower bound of the VNF-UCSP problem at hand. Furthermore, a partial solution can be constructed, installing services on articulation points.

4 Problem complexity

In this section, we investigate the computational complexity of VNFP-SP. We consider the complexity of the problem sub-cases obtained by relaxing one capacity constraint (service or link) at a time. We show that even the feasibility version of the problem with only one capacity constraint is $\mathcal{NP} - complete$. We then prove that the uncapacitated VNFP-SP is $\mathcal{NP} - complete$ on a general graph if costs are strictly positive, but that the feasibility version is polynomially solvable. For the uncapacitated case with not null cost, we prove that the problem is polynomially solvable for specific graph topologies (namely, full-mesh, ring and tree-like), In Table 1, a summary of the complexity results is presented.

Topology	Link cap feas	Service cap feas	Uncap opt	Uncap feas
General	$\mathcal{NP}-C$	$\mathcal{NP}-C$	$\mathcal{NP}-C$	$\mathcal{P}$
Full mesh	?	$\mathcal{NP}-C$	$\mathcal{P}$	$\mathcal{P}$
Ring	?	$\mathcal{NP}-C$	$\mathcal{P}$	$\mathcal{P}$
Tree-like	$\mathcal{NP}-C$	?	$\mathcal{P}$	$\mathcal{P}$

Table 1: Summary of complexity results

4.1 VNFP-SP Service Capacity Problem

We call *VNFP-SP Service Capacity Problem* the VNFP-SP with infinite arc capacity ($u_{ij} = \infty \quad \forall (i,j) \in A$).

Theorem 4.1. The decisional form of *VNFP-SP Service Capacity Problem* is $\mathcal{NP} - complete$ even if all the service costs are null.

Proof. We can prove it by reduction from the Bin Packing Problem (BPP).

Let us recall the decisional form of the BPP:

Definition 4.1. Consider a set of items I, each with an integer weight $w_i \geq 0$, and a set of bins of capacity C.
Given a threshold κ, there exists an assignment of all the items to the bin, such that at most κ bins are used?

Every instance of the BPP can be reduced to an instance of the *VNFP-SP Service Capacity Problem* with null service costs as follows:

- the graph $G(N, A)$ for the *VNFP-SP Service Capacity Problem* is a tripartite graph, as in Figure 1), with $N = I_1 \cup B \cup I_2$. The set I_1 (I_2 respectively) contains a node for each item $i \in I$ of the BPP. Let us call i_1 (i_2) the node corresponding to item i. The set B contains κ nodes, which are the only one that can host a VNF.
  The set of arcs A is constructed as follows: an uncapacitated arc exists connecting each node in I_1 to each node in B and each node in B to each node in I_2;
- for each item $i \in I$ a demand $k \in D$ is generated with origin i_1, destination i_2 and bandwidth equal to the item weight w_i;
- the service capacity is equal to the bin capacity C.

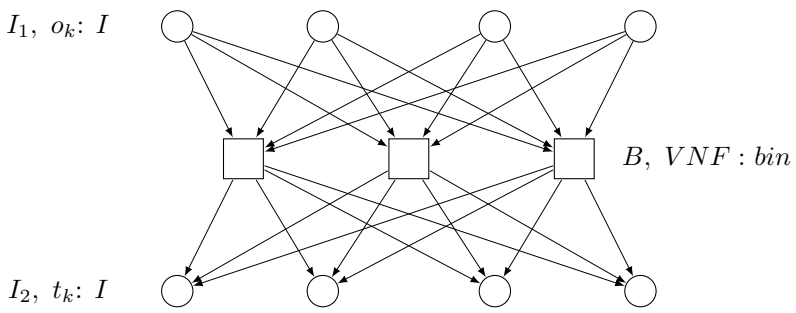


Figure 1: Reduction from Bin Packing Problem

By construction, the only possible paths, for any demand k, are of the form o_k-b-t_k, with $b \in B$, therefore, any path is a simple path. Furthermore, arcs are uncapacitated. Therefore the number of demands that can traverse any node b is limited by the service capacity (bin-capacity) only. In fact, as services are located only on nodes of B, if a service is routed on node b, then it must use the service located on b.

If there exists a feasible solution of the instance of *VNFP-SP Service Capacity Problem* then there exists a solution of the BPP that uses at most κ bins. Indeed, routing the demand $i_1 - i_2$ to path i_1-b-i_2 and therefore assigning the demand to the VNF $b \in B$ corresponds to assign the item i to the bin b. Any demand can be assigned to any node B, with the only limitation of the total capacity of the VNF. Similarly,in the BPP any item can be assigned to any bin

as long as the capacity is sufficient. Each open service corresponds to a used bin, thus the number of used bins is at most κ.

On the other hand, if the instance of *VNFP-SP Service Capacity Problem* is not feasible, it is not possible to assign all the demands to the κ services without violating their capacity, and this proves that more than κ bins are needed to host all the items.

Since the BPP is $\mathcal{NP} - complete$ and the *VNF Service Capacity Problem* without service costs belongs to $\mathcal{NP}$[3], then the VNFP-SP Service Capacity problem is $\mathcal{NP} - complete$ even if all the service installation costs are null.

■

4.2 VNF Link Capacity Problem

We call *VNFP-SP Link Capacity Problem* the VNFP-SP with infinite service capacity ($q = \infty$).

We prove that even considering the case of the *VNFP-SP Link Capacity Problem* where the service installation cost is zero ($c = 0$), the problem is $\mathcal{NP} - complete$.

Theorem 4.2. The decisional form of *VNFP-SP Link Capacity Problem* is $\mathcal{NP} - complete$ even if all the service costs are null.

Proof. We can prove it by reduction from the feasibility version of the Unsplittable Multicommodity Flow Problem (UMFP).

Let us recall the decisional form of the UMFP in the feasibility version:

Definition 4.2. Given a graph $G = (N, A)$ with a capacity for each arc and a set of unsplittable commodities, each with a source, a destination and an amount of flow, the problem asks for finding a path for each commodity connecting the source with the destination such that the overall flow on each arc does not exceed the arc capacity.

Every instance of the UMFP can be reduced to an instance of the *VNFP-SP Link Capacity Problem* without installation cost, as follows:

- the graph $G(N, A)$ for the VNF-SP Service Capacity problem has the same topology and arc capacity as the UMFP;
- the set of demands D is the set of commodities;
- the set of nodes that can host a service is the set of nodes that is origin of at least one demand.

If there exists a feasible solution of the VNFP-SP Link Capacity problem then there exists a routing for each demand of the multicommodity problem that does not violate the arc capacity constraints. First, we can observe that, being null all the service installation costs, we can install a service on each node that is origin of a demand, and assign to it all the demands originating from the node itself. Then, we route the demands from their origin to the destination on any path that does not violate the arc capacity. The obtained routing represents a feasible solution of the *Unsplittable Multicommodity Flow Problem.*

[3] we omit the proof for brevity

Since the *Unsplittable Multicommodity Flow Problem* is $\mathcal{NP}-complete$ even in its feasibility version and the *VNF Link Capacity Problem* without service installation costs belongs to $\mathcal{NP}$[4], then the *VNF Link Capacity Problem* is $\mathcal{NP}-complete$ even with null service installation costs.

■

4.3 Uncapacitated VNFP-SP

In the following we will prove that the fully uncapacitated (namely without link nor service capacity) *VNFP-SP* with non null service costs is NP-complete. Instead, when the service costs are null, the fully uncapacitated *VNFP-SP* is polynomial.

Theorem 4.3. The decisional form of the *Uncapacitated* VNFP-SP is $\mathcal{NP}-complete$.

Proof. We can prove it by reduction from the Set Covering (SC) problem.

Let us recall the decisional form of the SC:

Definition 4.3. Given a set of elements $\mathcal{U} = \{1, 2, ..., n\}$ (called the universe), a collection of subsets of $\mathcal{U}$, $\mathcal{S} = S_1, \ldots, S_m$, and a threshold κ, the set cover problem asks for finding a sub-collection of S that covers all the elements in $\mathcal{U}$ whose cardinality is smaller or equal to κ.

Every instance of the SC problem can be reduced to an instance of the *Uncapacitated VNFP-SP* as follows:

- the graph $G(N, A)$ of the *Uncapacitated VNFP-SP* is generated as a tripartite graph (see Figure 2), with $N = \{U_1, S, U_2\}$. The set U_1 (U_2 respectively) contains a node j_1 (j_2 respectively) for each element $j \in \mathcal{U}$. The set S contains a node for each each subset S_i of $\mathcal{S}$.
  Two arcs (j_1, S_i) and (S_i, j_2) exist if the element $j \in \mathcal{U}$ can be covered by the subset S_i. The arcs are uncapacitated;
- for each element $j \in \mathcal{U}$ a demand $k \in D$ is generated with source node $j_1 \in U_1$ and destination node $j_2 \in U_2$;
- only the nodes of set S can host a service (therefore each subset S_i of $\mathcal{S}$ is associated with a possible service). The cost of installing a service in any node in S is one;
- services are uncapacitated.

If there exists a solution of Uncapacitated VNFP-SP problem whose cost is at most κ then there exists a solution of the SC that uses at most κ subsets. A solution of the SC problem is obtained from the solution of the *Uncapacitated VNFP-SP* as follows. If the demand (j_1, j_2) is routed through the node s_i then the corresponding element j is covered by the subset S_i. A subset is in the cover if a service is installed on the corresponding node. Thus the costs of installed services equals the number of used subsets.

Since the *Set Covering Problem* is $\mathcal{NP}-complete$ and the *Uncapacitated VNFP-SP* belongs to $\mathcal{NP}$, then the *Uncapacitated VNFP-SP* is $\mathcal{NP}-complete$.

■

[4] we omit the proof for brevity

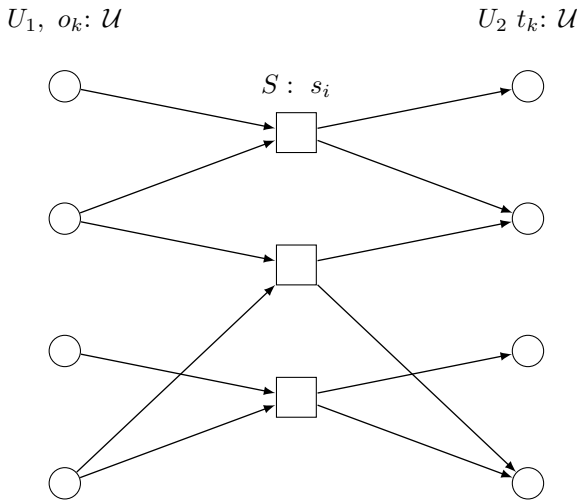


Figure 2: Reduction from Set Covering Problem

Differently from the case with link or service capacities, the Uncapacitated VNFP-SP problem with null installation cost is polynomial solvable.

Theorem 4.4. The *VNFP-SP without Capacities and without service installation costs* is polynomial.

Proof. A feasible solution, if exists, can be found as follows:

- as there is not cost for installing the services and the services have not capacity limit, each demand can have its own service. Therefore, a service can be installed on any node that is origin of a demand. It takes $|D|$ steps to install the services;
- as there is not arc capacity, each demand can be routed independently from the others. The chosen service placement does not induce any limitation on the possible routings. Therefore, any polynomial algorithm able to find a simple path from the origin to the destination of a demand can be used. The worst case is to explore every node and very arc, and the maximum number of arcs is $|V| \times (|V| - 1)$. Therefore this step has complexity $|D| \times |V|^2$

The overall complexity is thus $O(|D| \times |V|^2)$

■

4.4 VNFP-SP on special graph topologies

In the following, we will consider some graph topologies, namely full-mesh, ring and arborescence-like, and we will prove that for these topology the Uncapacitated VNFP-SP is polynomial solvable if costs are different from zero (without costs we already proved that is polynomial by the general case).

Definition 4.4. A fully meshed topology is a fully connected graph, i.e. a graph where there is a an arc between any pair of nodes.

Property 4.1. The Uncapacitated VNFP-SP is polynomially solvable on fully meshed graphs.

Proof. First, as there is no service capacity limit, it is sufficient to locate one service to satisfy all the demands, and this service can be located arbitrarily on any node i. In fact, as there is no link capacity, and the graph is full mesh, each demand k can be routed independently along the path $o_k - i - t_k$. Therefore a solution can be built in $O(|D|)$.

■

Remark 4.1. The VNFP-SP Service Capacity is $\mathcal{NP}$-complete on a full mesh topology even in the feasibility version.

Proof. We briefly sketch the proof. The set of nodes is build as follows:

- two nodes for each bin, the source and the destination of the demand associated with the bin
- k nodes that can host a service.

Each item is associated with a demand and thus the amount of traffic of the demand is equal to the weight of the item. Service capacity is equal to bin capacity. If a feasible solution exists for the VNFP-SP problem than each demand passes through one service thus each item is assigned to one bin and at most k bins are needed. ■

Definition 4.5. A (symmetric) ring topology is a graph in which each node is exactly connected with two other nodes, in such a way that a closed loop is formed.

As each node in the ring graph has exactly two adjacent neighbors, there are exactly two paths from one node to any other nodes (see Figure 3).

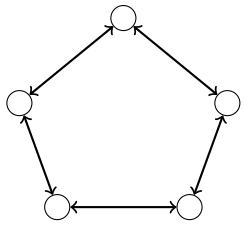


Figure 3: Ring graph

Property 4.2. The Uncapacitated VNFP-SP is polynomially solvable on a ring topology.

Proof. First of all we observed that any feasible solution with a single service is optimal. In the following, we show how to find such a solution in polynomial time. W.l.o.g., we first randomly choose one node to install a service instance and label it as node 1, then label all the nodes with an increasing integer in clockwise (or anticlockwise) direction starting from it (see Figure 4 for an example).

There are exactly two paths from any node to any other node, thus there are exactly two paths (clockwise path and anticlockwise path) for each demand.

In order to route the demand k passing by the service node, the following rule can be used: if the label of o_k is smaller than the label of t_k, then choose the anticlockwise path for this demand, otherwise choose the clockwise path. As setting a path takes at most $|N|$ steps, a solution can be computed in $O(|D| \times |N|)$.

■

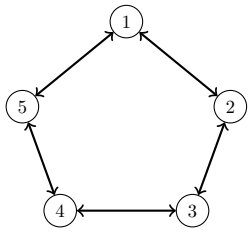


Figure 4: Node indexes on a ring topology

Remark 4.2. The VNFP-SP Service Capacity is $\mathcal{NP}$-complete on a ring topology even in the feasibility version.

Proof. We can prove it by reduction from the Bin Packing problem. Every instance of BPP can be reduced to VNFP-SP on a ring as follows:

- the set of nodes of the graph is divided into four subsets: B_1, B_2, I_1, I_2: $|I_1| = |I_2| = |I|$, namely for each item i there exists a pair of nodes i^s, i^t such that $i^s \in I_1, i^t \in I_2$, and $|B_1| = \left\lceil \frac{\kappa}{2} \right\rceil, |B_2| = \left\lfloor \frac{\kappa}{2} \right\rfloor$, namely there exists a node in $B_1 \cup B_2$ for each bin. Nodes are connected in a ring, as shown in Figure 5;
- each node in $B_1 \cup B_2$ can host a service with capacity C. Nodes in $I_1 \cup I_2$ cannot host services;
- for each item $i \in I$ there exists a demand k_i from i^s to i^t, with $d_k = w_i$.

There exists a feasible solution for the VNFP-SP Service Capacity problem if and only if there exists a solution for BPP that uses at most κ bins. ■

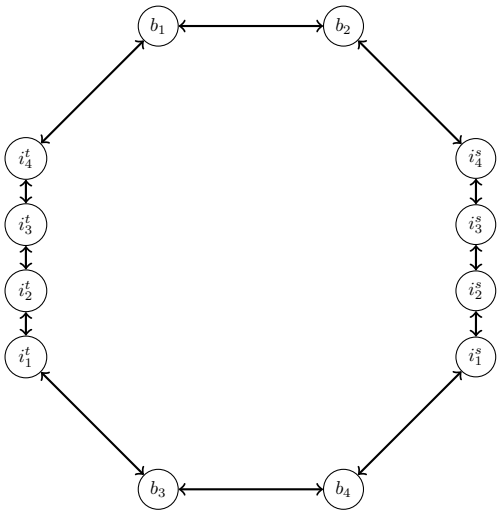


Figure 5: Ring graph for reduction from Bin Packing

Definition 4.6. In the following we will call *symmetric arborescence* a symmetric graph $T(V, A_T)$ with no cycles (as shown in Figure 6). We will call *leaf* a node if it is adjacent to only one node.

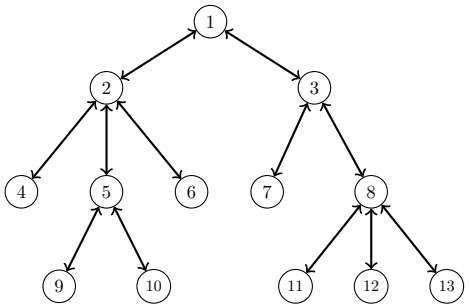


Figure 6: Symmetric arborescence

Remark 4.3. As there is no cycles in a symmetric arborescence, there is a unique simple path from a node to any other node. Thus, we can infer:

- the simple path between two adjacent nodes is the arc connecting them. For example in Figure 6, the path from node 5 to node 2 is arc $(5, 2)$.
- any path originating (ending, respectively) from (in, respectively) a leaf node must pass through the node adjacent to the leaf itself. For instance, the path from leaf node 9 to node 2 has to pass through the node 5.

Property 4.3. The Uncapacitated VNFP-SP is polynomially solvable on a symmetric arborescence.

Proof. According to Remark 4.3, we can further infer:

- if the source o_k and destination t_k of demand k are adjacent nodes, the routing path of demand k is the arc (o_k, t_k). To serve the demand, a service must be installed either on its source node o_k or on its destination node t_k.

- as the path originating from a leaf node i has to pass through its adjacent node j to reach other nodes, to serve any demand originating in i we can always install the service on node j and the cost of such solution is not larger than the cost of the solution where the service is installed on the leaf node. The same apply for a demand ending in i. As a consequence, a demand originating in leaf node i (respectively, ending at i) can be represented by a demand originating (respectively, ending) at the only node adjacent to i. In a sense, we collapse a leaf on its adjacent node.

The solution procedure relies on iteratively generating updated graphs by collapsing leaves on their adjacent nodes. A feasible solution on a symmetric arborescence $T(V, A_T)$ can be found as follows:

1. for each demand k that is not already served on the initial graph $T(V, A^T)$, if its current source node i or/and its destination node j is a leaf node, then let $o_k = l$ or/and let $t_k = h$, where l is the node adjacent to i and h is the node adjacent to j. It takes $2 \times |D|$ steps to check all the unserved demands in the worst case.

2. then we shrink the arborescence $T(V, A_T)$ by removing all the leaves and the related arcs, and we get a new graph $T'(V', A'_T)$. It takes at most $2 \times |D|$ steps to remove the leaves.

3. we consider each demand k that is not served on the current graph $T'(V', A'_T)$: if its current source node o_k or destination node t_k has already a service then k is considered as a served demand. Otherwise, if its current source node and destination node are the same node, we install a service on this node, and demand k is regarded as served. This phase can guarantee that we serve all the demands with a simple path before the arborescence is totally cut off. It takes $3 \times |D|$ steps to perform this step.

By repeating steps 1-3 until all the demands are served, we can find the minimum number of services to be install to serve all the demands in the network. Moreover, the routing path of each demand is the set of adjacent nodes found in each iteration.

As an arborescence has at least two leaves (when it represents a path), in each iteration we cut at least two leaves, so that the number of performed iterations 1-3 is $\left\lfloor \frac{|V|}{2} \right\rfloor$ in the worst case.The overall complexity is thus $O(|V| \times |D|)$.

■

The procedure can be formalized as shown in Algorithm1, where the arborescence is denote as $T(V, A_T)$, where L is the set of leaves at each iteration and P_i is the adjacent node of leaf $i \in L$. The set of nodes that host a service is represented by S, while R_k records the routing path of demand $k \in D$.

Remark 4.4. In an arborescence, there is only one simple path for each demand and thus, if the feasible region is not empty, the feasible region contains only one solution w.r.t.the routing part, the very one generated by Algorithm 1.

Algorithm 1 RDCL (Rebuild-Demands and Cut-Leaves)

procedure VNFP-SP($T(V, A_T), D$)
  Input Data: $T(V, A_T), D$
  Initialization: $L \leftarrow \emptyset, P\{L\} \leftarrow 0, S \leftarrow \emptyset, R\{D\} \leftarrow \emptyset$
  while $D \neq \emptyset$ **do**
    let: $L \leftarrow$ set of leaves, $P_i \leftarrow$ adjacent node to $i, \forall i \in L$
    for all $k \in D$ **do**
      if $o_k \in L$ **then**
        let: $o_k \leftarrow P_{o_k}, R_k \leftarrow R_k \cup P_{o_k}$
      if $t_k \in L$ **then**
        let: $t_k \leftarrow P_{t_k}, R_k \leftarrow R_k \cup P_{t_k}$
    let: $V \leftarrow V \backslash L$, $A_T \leftarrow A_T \backslash setof\{i \in L\}(i, P_i)$
    for all $k \in D$ **do**
      if $o_k \in S$ or $t_k \in S$ **then**
        let: $D \leftarrow D \backslash k$
        continue
      if $o_k == t_k$ **then**
        let: $S \leftarrow S \cup o_k$
        let: $D \leftarrow D \backslash k$
  ▷ end while
  Output Data: $S, R\{D\}$

Thus, if we consider a link capacity, either there exists a feasible solution, and Algorithm 1 generates it, or the algorithm returns an unfeasible routing due to the link capacity, and then the considered instance is not feasible.

5 Mathematical formulation

As mentioned, in the telecommunication literature two main formulations strategies have been proposed: the VNF placement and routing formulation [1] and the VNE formulation [3]. In this section we present two formulations based on such two strategies. The first one is inspired by network design and facility location problems. The second one, the Split-Path (SP), is based on the decomposition of each demand path into several sub-paths, each associated with a NVF serving the demand[5].

is inspired by the fact that for be considered as a cleaned version of embedding, considering that for VNF chaining, the network to embed is a linear graph. In Table 2 the notation used for the problem is summarized. Furthermore, the variables used by the two models are reported. As some parameters and variables are common to both models and some are model dependent , in the last column we report in which model the parameter/variable is used.

In both models, binary variables y_i represents the location of an instance of the VNF on node i and binary variables z_i^k represents the assignment of demand k to a the instance of the VNF located on node i. The two models differ in the way the routing is represented. In both, arc binary variables are used, that

[5]VNE is related to SP as for VNF chaining the network to be embedded is indeed a linear graph.

Notation		Models
	Sets	
N	set of nodes	both
A	set of arcs	both
D	set of demands	both
	Network parameters	
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
	Binary variables	
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
	Model depend variables	
x_{ij}^k	1 if arc $(i,j) \in A$ is used by demand $k \in D$	VNFPR
x_{ij}^{k1}	1 if arc $(i,j) \in A$ is used by demand k on sub-path 1	SP
x_{ij}^{k2}	1 if arc $(i,j) \in A$ is used by demand k on sub-path 2	SP
	Integer variables	
π_i^k	position of node $i \in N$ in the path used by demand $k \in D$	VNFPR

Table 2: Mathematical notation

are equal to one if a given arc is used by a given demand. In VNF-PR, these variables are x_{ij}^k. In SP, the path is explicitly considered in the two sub-paths: the first from origin o_k to the node where the service used by the demand is located x_{ij}^{k1} and the second from the service node to the destination t_k x_{ij}^{k2}. Both models can be generalized to take into account multiple services, even hosted on different nodes. Nevertheless, SP model is able to manage straight forward the case where the order of services is fixed (as it explicitly split the path in sub-paths from source to service, service to next service, etc.), but cannot be straightforward extended to manage the partial or no order case. VNF-PR model can manage the case of no order by simple generalization of the model. In the case of partial or full order, some constraints must be added to impose the order, but it is not necessary to move to a not-polynomial variable representation.

As we want to enlighten the common points and difference between the two models, we decided to present them in parallel, starting from the common part.

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

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

$$z_i^k \leq y_i \qquad \forall k \in D, i \in N \tag{2}$$

$$\sum_{k \in D} d_k z_i^k \leq q \qquad \forall i \in N \tag{3}$$

The objective function minimizes the sum of the opened services (i.e., instances of the VNF). Eq. (1) imposes that each demand is assigned to exactly

one instance of the service. Eq. (2) guarantees that if there is no VNF instance is installed, then it cannot be assigned to any demand. Eq. (3) imposed that each instance of the VNF can serve a maximum quantity of demand given by its capacity q.

The link capacity constraints are similar for the two models:

SP:

$$\sum_{k \in D} d_k (x_{ij}^{k1} + x_{ij}^{k2}) \leq u \qquad \forall (i,j) \in A \tag{4}$$

VNF-PR:

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

We now present the constraints characterizing each formulation. The major difference is in the way the routing is managed, and, as a consequence, how the models 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 VNFPR formulation, routing is implied by the classical flow balance constraints and the connection between assignment and routing is implied by coherence constraints and isolated loop elimination constraints.

SP:

$$\sum_{j:(i,j)\in A} x_{ij}^{k1} - \sum_{j:(j,i)\in A} x_{ji}^{k1} = \begin{cases} 1 - z_i^k & \text{if } i = o_k \\ -z_i^k & \text{otherwise} \end{cases} \quad \forall k \in D,\ i \in N, \tag{6}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{k2} - \sum_{j:(j,i)\in A} x_{ji}^{k2} = \begin{cases} z_i^k - 1 & \text{if } i = t_k \\ z_i^k & \text{otherwise} \end{cases} \quad \forall k \in D,\ i \in N \tag{7}$$

$$\sum_{j:(j,i)\in A} (x_{ji}^{k1} + x_{ji}^{k2}) \leq 1 \ \forall k \in D, i \in N \tag{8}$$

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

Each demand is routed in two sub-paths: from the source node to the VNF node (Eq. (6)), then from the VNF node to the destination node (Eq. (7)). These two constraints impose that the routing of the demand pass through the VNF instance assigned to the demand itself, therefore ensure that the assignment is consistent.

The price to pay to obtain a consistent routing is to know in advance how the path it will be split, therefore, in case of multiple VNF types that can

be allocated on different nodes, it will ask for a fixed order in the visit of thee VNF types. If the order is partial or no order must be imposed, then an extended formulation is necessary. The formation of (isolated or not) cycles is prevented by Eq. (8) and (9). Each demand can pass through a node at most once.

VNF-PR:

$$\sum_{j:(i,j)\in A} x_{ij}^k \; - \sum_{j:(j,i)\in A} x_{ji}^k = \begin{cases} 1 \text{ if } i = o_k \\ -1 \text{ if } i = t_k \\ 0 \text{ otherwise} \end{cases} \quad \forall k \in D,\ i \in N \tag{10}$$

$$z_i^k \;\leq \sum_{(j,i)\in A} x_{ji}^k \; \forall k \in D,\ i \in N \tag{11}$$

$$\pi_j^k \;\geq\; \pi_i^k + x_{ij}^k - |N|\ (1 - x_{ij}^k)\ \forall k \in D,\ (i,j) \in A \tag{12}$$

Each demand is routed from its source to its destination with the classical flow balance constraints (10) and it is forced to pass through VNF using constraint (11), that impose that a demand k can be assigned to a the service located on node i only if the routing pass through the given node. The simple path assumption and the elimination of isolated cycles containing the service are enforced using the TSP-like labeling variables π and constraints (12). Integer variables π_i^k represent the position of node i in the routing path of demand k. VNF-PR formulation can be generalized to the case where exist multiple service that can be located in different nodes with any imposed order (full, partial, none). In fact, variables π (with variable z) can be used to impose a given full or partial order of the services along the demand path. If no order is present, no new additional constraints must be added.

5.1 Relation between the two formulation relaxations

In this section we show that the SP formulation produces a continuous relaxation bound that is always equal to or stricter than the one produced by the VNFPR continuous relaxation. Furthermore, we provide an instance were the bound is stricter.

Let us consider a solution of the continuous relaxation of SP (x^1, x^2, y, z). The routing induced by such solution is a feasible routing for the underling continuous multicommodity flow problem obtained removing location and assignment variables. Thus, the component of the routing due to any demand $k\ inD$ can be decomposed in a set of simple paths P_k from its origin o_k to its destination t_k and a set of cycles C_k.We refer to a path $p \in P^k$ (or a cycle $c \in C^k$) as induced by the routing of demand k. We use the sets P_k and C_k to represent both nodes and arcs belonging to the simple paths and cycles.
Let call f_p the fraction of flow associated to a path $p \in P_k$ and f_c the fraction of flow associated to a cycle $c \in C_k$.
Flows on paths f_p can be determined as follows:

- find a simple path p that has at least one arc not shared with any other path or cycle
- $f_p = \min_{(i,j)\in p}\{x_{ij}^{k1} + x_{ij}^{k2}\}$
- remove the flow associate to p from the current routing solution (and arcs with no more flow from the graph) obtaining a residual routing solution
- repeat the procedure

When all simple paths are removed, only cycles, if exit, will contribute to the residual routing solution. The same procedure can be repeated starting for any cycle that has at least one arc no common to any other cycle.

In the following of this section, when it is not differently specified, for the sake of brevity, we refer to a solution of the continuous relaxation of SP, as a feasible solution.

Remark 5.1. Given a feasible solution (x^1, x^2, y, z) and a demand k. Let consider the set of nodes of the cycle c that are not shared with any simple path: $N_c = \{i \in c \setminus P_k\}$. Due to flow balance constraints (6)-(7), we have three possible cases for all nodes belonging to N_c:

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

Property 5.1. Let consider a feasible solution (x^1, x^2, y, z), a demand k and the decomposition $\{P^k, C^k\}$ of its routing. Let suppose that exists a cycle $c \in C^k$ that shares at least one node with a simple path $p \in P^k$.
If a (partial) service is located on one of the nodes 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 $\frac{1}{2}$.

Proof. By remark 5.1 and flow balance constraints (6)-(7), the flow incident to any node l belonging to the cycle is induce by both semi-path variables and $x_l^{k1} = x_l^{k2}$. Therefore, the isolated cycle elimination constraint (8)-(9) implies that $x_l^{k1} = x_l^{k2} \leq \frac{1}{2}$. ■

Definition 5.1. Given a solution of SP, a demand k and the decomposition of its routing $\{P_k, C_k\}$, we call a cycle $c \in C_k$ that do not host any (partial) service on any node that belong only to cycles, i.e. $z_i^k = 0 \quad \forall i \in N_c = \{c \cap P_k\}$ a *service-free cycle*.

Definition 5.2. A solution is called a *minimal-routing solution* if for all demands k the induced cycles by their routing, if they exist, are service-free cycles.

Property 5.2. Any minimal-routing solution can be transformed in an equivalent solution (in term of costs, location and assignment) removing the service-free cycles from the routing.

Proof. Let consider a solution x^1, x^2, y, z of the continuous relaxation and consider w.l.g. that there exist an unique demand $k \in D$ such that its routing contains a single service-free cycle $c \in C^k$[6]. If we remove from the solution the flow due to the cycle f_c,the solution is still a feasible routing solution: any node belonging to the cycle has one entering and one exiting arc of the cycle. As no (partial) service is installed on the nodes belonging only to the cycle c, the obtained routing is coherent with the assignment variables, and therefore with the location variables. Thus, the obtained solution is still feasible and has the same cost of the original one. ■

Theorem 5.1. Any minimal-routing solution of the continuous relaxation of SP can be mapped in an equivalent solution of VNFPR in terms of routing, location and assignment and therefore cost.

Proof. We can start observing that some variables and constraints are the same for both formulations:

- opening variables y_i and assignment variables z_i^k
- objective function (that involves only opening variables y)
- constraints (1), (2) and (3)

Therefore a direct mapping of variables y and z will satisfy these common constraints.

Routing variables are different for the to formulation, but we can make an easy mapping between them:

$$x_{ij}^k = x_{ij}^{k1} + x_{ij}^{k2} \tag{13}$$

As simple path constraints on routing variables on SP (8) and (9) imply that

$$x_{ij}^{k1} + x_{ij}^{k2} \leq 1$$

this correspondence guarantees that x_{ij}^k is always in $0, 1$.

With this mapping, link capacity constraints of SP (4) implies the link capacity constraint of VNFPR (5).

Let consider routing constraints for the SP formulation (6)-(7), we now prove that they imply the routing constraints for the VNFPR formulation (10). We have three cases:

$$i = o_k, \quad i = t_k, \quad i \neq o_k, t_k$$

Let consider the case $i = o_k$, then for the SP formulation we have:

(a) $\sum_{j:(i,j)\in A} x_{ij}^{k1} - \sum_{j:(j,i)\in A} x_{ji}^{k1} = 1 - z_i^k \quad \forall k \in D$

(b) $\sum_{j:(i,j)\in A} x_{ij}^{k2} - \sum_{j:(j,i)\in A} x_{ji}^{k2} = z_i^k \quad \forall k \in D$

[6]if there exists more than one cycle and/or more than one demand, the procedure can be repeated for each of them.

Summing them, we get:

$$\sum_{j:(i,j)\in A} (x_{ij}^{k1} + x_{ij}^{k2}) - \sum_{j:(j,i)\in A} (x_{ji}^{k1} + x_{ij}^{k2}) \;=\; 1 \quad \forall k \in D$$

and using (13):

$$\sum_{j:(i,j)\in A} x_{ij}^{k} - \sum_{j:(j,i)\in A} x_{ji}^{k} \;=\; 1 \quad \forall k \in D$$

that is exactly constraint (10) in the case $i = o_k$. The other two cases, can be obtained with a similar procedure, so, for the sake of brevity, we avoid to present the detailed proof.

Now we show that SP routing constraints for the semi-path (6) imply routing-location connecting constraints of VNFPR (11).
Let consider the case $i \neq o_k$:

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

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 (13), and we verify constraint (11):

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

that is .

The last step is to find suitable values for π variables. Let consider a demand k and its routing. We can construct an induced graph, as follows:

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

where $(i,j) \in A_k$ iff $x_{ij}^{k1} + x_{ij}^{k2} > 0$ and $\exists p \in P_k : (i,j) \in p$, i.e. the arc belong at least to a simple path. A node $\hat{\imath}$ belongs to N_k if exists an incident 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 do not contain arcs that belongs only to cycles in C^k, the obtained graph is acyclic, thus we can define π^k as the value of the longest path form node o_k to node j. That is, for any demand $k \in D$ we can set:

- $\pi_{o_k} = 0$
- $\pi^k_{\hat{j}} = \max_{i:(i,\hat{j})\in A^k}\{\pi^k_i + x^k_{ij}\}$

We now need to determine a value of π for the nodes we removed. Let us consider a cycle and the nodes belonging to the cycle that are in common with a simple path (nodes from l to m in Figure 7). They already have an assigned value π.

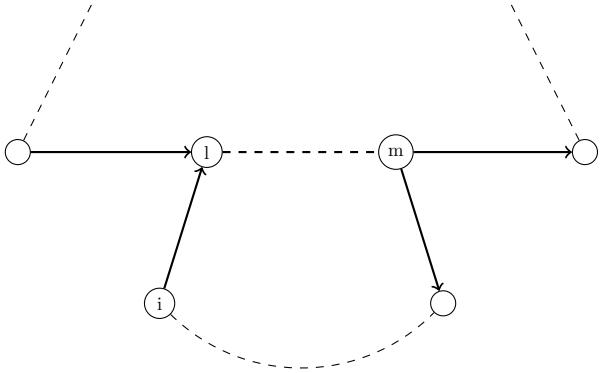


Figure 7: An example of attached cycle

Let consider the two extreme nodes l and m, corresponding to the smallest (π^k_l) and largest (π^k_m) value of variable π, , respectively. For all the nodes $i \in N_c$:

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

Now, we prove that our choice of π always satisfy constraints (12).
We can observe that $x_{ij} <= 1$ in any feasible solution, therefore the values of π are bounded by $|N| - 1$ (case of linear graph with origin and destination on the extremes nodes). For any arc (i, j) belonging to the induced acyclic graph, the proof is straightforward, by the definition of longest path.

$$\pi^k_{\hat{j}} = \max_{i:(i,\hat{j})\in A^k} \pi^k_i + x^k_{ij} \geq \pi^k_i + x^k_{ij} \quad \forall (i,j) \in A^k$$

Now we have to consider the arcs in $A \setminus A^k$, we have two cases:

- arcs belonging to the cycle but not a path.
  Due to property 5.1, we can infer that for any arc belonging to the cycle but not the path $x^k_{ij} \leq \frac{1}{2}$. Therefore, the term:

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

  is always negative with module at least $\frac{1}{2}$, under the assumption that $|N| \geq 2^7$. Therefore for any two nodes in the cycle, as they have the same value of π, the constraint is valid.

[7] condition necessary to have a cycle

- 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), we need to prove that:
  $$\pi_l \geq \pi_i + \alpha - |N|(1-\alpha) = \gamma$$
  where $\alpha \leq \frac{1}{2}$ is the value of the routing variable. Let rewrite the right-hand side term with the value of π for nodes belonging to the cycle:
  $$\gamma = \frac{\pi_m}{2} + \frac{\pi_l}{2} + \alpha - |N|(1-\alpha)$$
  If we call n the number of arcs in the path between node l and node m, and β the value of the associated routing variable (x_{ij}) on this path[8], we can deduce:
  $$\pi_m = \pi_l + \beta n$$
  then:
  $$\gamma = \frac{\beta n}{2} + \frac{\pi_l}{2} + \alpha - |N|(1-\alpha).$$
  If a cycle exists, $n \leq |N| - 1$ and using that $\beta \leq 1$:
  $$\gamma = \pi_l + \frac{(|N|-1)}{2} + \alpha - |N|(1-\alpha)$$
  rearranging terms:
  $$\gamma = \pi_l + (\alpha - \frac{1}{2})(|N|+1)$$
  and using $\alpha \leq \frac{1}{2}$:
  $$\pi_l \geq \gamma$$
  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.

■

We now present an instance for which the continuous relaxation bound provided by the SP formulation is stricter than the one provide by the VNFPR formulation.

We consider the graph in Figure 8, and a set of three demands:

$$k_1 : 1 \to 2; \quad k_2 : 4 \to 5; \quad k_3 : 7 \to 8$$

all demands have a required bandwidth $d_k = 1$. Service and link capacity are assumed large enough to host all demands.
We can observe that the graph contains two articulation points (nodes 3 and 6) and 3 biconnected components:

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

Each demand has its origin and destination in a different biconnected component: k_i in $BC_i \quad \forall i = 1, 2, 3$.

[8] due to flow balance constraints it is equal for all arcs

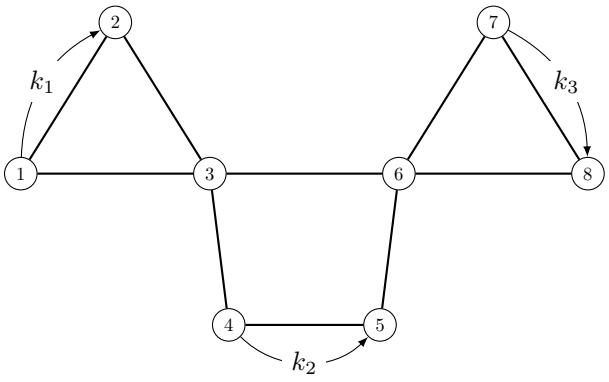


Figure 8: Example instance to compare continuous relaxation of the formulations

On this instance, the continuous relaxation bound obtained by the SP formulation is $\frac{4}{3}$ and the one obtained by the VNFPR formulation is 1. Let us look at a solution for both SP and VNFPR formulations[9].

For SP, a partial service is installed on nodes $2, 5, 8$. In Table 3, we report the solution with respect to location and assignment omitting zero variables. Each column represent a node. Fig. 9 shows the routing solution for the three demands.

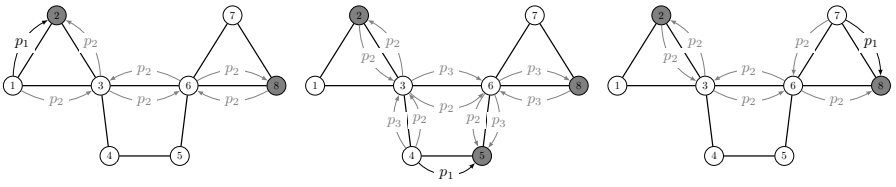
$y_2 = \frac{1}{2}$	$y_5 = \frac{1}{3}$	$y_8 = \frac{1}{2}$
$z_2^1 = z_8^1 = \frac{1}{2}$	$z_5^1 = z_5^2 = z_5^2 = \frac{1}{3}$	$z_8^1 = z_8^1 = \frac{1}{2}$

Table 3: Location and assignment in the continuous relaxation solution of SP formulation

In Table 4, the routing are reported with the associated fraction of assignment, the node where the used (fraction of service is located is in bold. Demand k_1 (k_3) is serve for half of the flow by a service in its connected component, node 2 (node 8) and for the other half by a service in another connected component, node 8 (node 3). Demand k^2 is split in three paths, each of them served by a service in one connected component.

For VNF a single service is installed on node 1.As previously we show the solution on Fig. 10 and report details on flows in Table 5. In this solution, each demand is routed in the direct arc connecting its origin to its destination, satisfying with this part of routing the flow balance constraints. To reach the service, both demand k_2 and k_3 use two isolated loops (disconnected by the principal path). We can observe that two loops are necessary, in fact, to satisfy the isolated loop elimination constraint, the fraction of flow in a loop must be smaller then a given threshold γ, depending on the cardinality of N (in this case

[9]for both formulation different equivalent solutions exists, both for location and routing. To the sake of clarity, we chose a "compact" solution for both formulations privileging symmetric and less fractional solutions, with short routings and a reduced number of isolated cycles



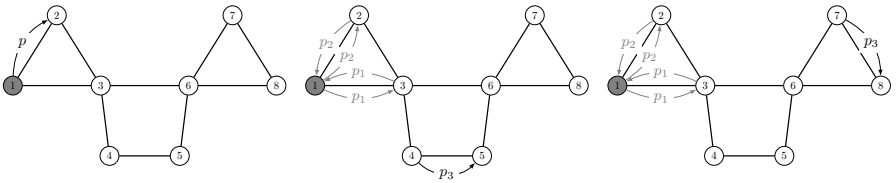
(a) $k_1 : \frac{1}{2}$ on each sub-path. (b) $k_2 : \frac{1}{3}$ on each sub-path. (c) $k_3 : \frac{1}{2}$ on each sub-path.

Figure 9: The routing solution of continuous relaxation of the SP formulation

demand	paths	fraction f_p
k_1	$p_1^1 : 1 \to \mathbf{2}$ $p_2^1 : 1 \to 3 \to 6 \to \mathbf{8} \to 6 \to 3 \to 2$	$\frac{1}{2}$
k_2	$p_1^2 : 4 \to \mathbf{5}$ $p_2^2 : 4 \to 3 \to \mathbf{2} \to 3 \to 6 \to 5$ $p_3^2 : 4 \to 3 \to 6 \to \mathbf{8} \to 6 \to 5$	$\frac{1}{3}$
k_3	$p_1^3 : 7 \to \mathbf{8}$ $p_2^3 : 7 \to 6 \to 3 \to \mathbf{2} \to 3 \to 6 \to 8$	$\frac{1}{2}$

Table 4: Routing solution of continuous relaxation of SP formulation for example instance

$\gamma \leq \frac{8}{9}$). Therefore, any other couple of cycles with a repartition of flows γ, 1_γ in feasible.



(a) k_1 (b) k_2 (c) k_3

Figure 10: The routing solution of VNFPR model

demand	paths	fraction f_p
k_1	$p^1 : \mathbf{1} \to 2$	1
k_2	$p_1^2 : \mathbf{1} \to 3 \to 1$ $p_2^2 : \mathbf{1} \to 2 \to 1$	$\frac{1}{2}$
	$p_3^2 : 4 \to 5$	1
k_3	$p_1^3 : \mathbf{1} \to 3 \to 1$ $p_2^3 : \mathbf{1} \to 2 \to 1$	$\frac{1}{2}$
	$p_3^3 : 7 \to 8$	1

Table 5: Routing solution of continuous relaxation of VNFPR formulation for example instance

5.2 Valid inequalities

In this section, we describe some valid inequalities and in the computational section we will present the effect of their addition to both formulations.

We can rewrite the service capacity constraint (3) taking into account that, as a demand had to reach the node where the service is located, the total demand that can be served by the service located on the node i is limited not only by the service capacity but also by overall incident link capacity of node i (at exception of the demands served in the origin node i). Symmetrically, the overall output link capacity impose a limit, because the demand after being served must reach its destination. We can calculate the maximal demand that can be served by a node as follow:

$$\bar{q_i} = \min(q, \max(\sum_{(i,j)\in A} u + \sum_{k\in D: t_k = i} d_k, \sum_{(j,i)\in A} u + \sum_{k\in D: o_k = i} d_k)) \tag{14}$$

Therefore, constraint (3) can be substituted by:

$$\sum_{k\in D} d_k \; z_i^k \;\leq\; \bar{q_i} \; \forall i \in N \tag{15}$$

Both constraint (3) and constraint(15) can be strengthen using the service location variable y, obtaining:

$$\sum_{k\in D} d_k \; z_i^k \;\leq\; q \; y_i \; \forall i \in N \tag{16}$$

and

$$\sum_{k\in D} d_k \; z_i^k \;\leq\; \bar{q_i} \; y_i \; \forall i \in N \tag{17}$$

respectively.

When the capacity of a single VNF instance is not large enough to serve all the demands, a bound on a number of VNF instances can be calculated and introduced in a VI:

$$\sum_{i\in N} y_i \;\geq\; \left\lceil \frac{\sum_{k\in D} d_k}{q} \right\rceil \; \forall i \in N \tag{18}$$

Additionally, VIs inspired by cover inequalities can be added. As preliminary tests showed that adding all cover inequalities was not effective, we decide to select only the *max-minimal cover*. We define the max-minimal cover C_1 with respect of service capacity as the minimal cover, i.e.

- $\sum_{k \in C_1} d_k > q$
- $C_1 \setminus k$ is not a cover for any $k \in C_1$

such that $|C_1|$ is maximal among all possible covers. We can find C_1 as follows: we order the set of demands in increasing order of bandwidth, and then select them until the total capacity is reached. Let us call $M_1 = |C_1| - 1$, then we introduce the following:

$$\sum_{k \in D} z_i^k \ \leq \ M_1 \ \forall i \in N \tag{19}$$

We can define in the same way a *max-minimal cover* C_2 for link utilization, changing the total demand condition as follows $\sum_{k \in C_2} d_k > u$. Thus, if we call $M_2 = |C_2| - 1$, we get the following VIs:

For VNF-PR:

$$\sum_{k \in D} x_{ij}^k \ \leq \ M_2 \quad \forall (i,j) \in A \tag{20}$$

For SP:

$$\sum_{k \in D} (x_{ij}^{k1} + x_{ij}^{k2}) \ \leq \ M_2 \quad \forall (i,j) \in A \tag{21}$$

6 Computational Results

We tested the two formulations and the proposed valid inequalities on 16 networks (with minimum 10 nodes and maximum 28 nodes) from the SNDLib ([14]). The topologies and the traffic demands (i.e., source, destination, required bandwidth) are taken directly from SNDLib. To analyze the impact of the VNF and of the link capacity, we generated different capacity profiles. Three levels of capacity are considered: high, low and medium, both for VNF and link capacity yielding 9 instances for each network. The high capacity is computed so as to guarantee that all the demands can be served by a single VNF or can be routed on a single link (that is, the problem is uncapacitated). Low VNF capacity is twice the total amount of 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). Low link capacity the smallest capacity needed to route all the demands independently of VNF location. Medium capacity is the average between high and low. Table 6 shows all the tested instances with their network size and the three levels of capacity profiles. In the following, we denote with *h, m and l* the *high, medium and low* capacity level respectively. Instances are solved with AMPL and IBM ILOG CPLEX 12.7.0.0 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 3000Mb are set. First we present the results of continuous relaxation (Section 6.1), then we report about the integer problem (Section 6.2) and finally we evaluate the impact of the connected components based preprocessing (Section 6.3).

Data from SNDLib					Capacity			
Network	$\|N\|$	$\|L\|$	$\|D\|$	$\sum_{k \in D} d_k$ (high cap)	Service		Link	
					medium	low	medium	low
abilene	12	15	132	3000002	1750001	500000	1914642	829282
atlanta	15	22	210	136726	77478	18230	78065	19404
dfn-bwin	10	45	90	548388	329032	109677	302152	55916
di-yuan	11	42	22	53	31	9	29	5
france	25	45	300	99830	53908	7986	54621	9413
geant	22	36	462	2999992	1636359	272726	1679930	359868
janos-us	26	42	650	80000	43076	6153	43812	7624
newyork	16	49	240	1774	997	221	920	66
nobel-eu	28	41	378	1898	1016	135	1056	214
nobel-germany	17	26	121	660	368	77	367	74
nobel-us	14	21	91	5420	3097	774	2953	486
norway	27	51	702	5348	2872	396	2853	358
pdh	11	34	24	4621	2730	840	2502	384
polska	12	18	66	9943	5800	1657	5469	995
sun	27	51	67	476	255	35	264	53
ta1	24	51	396	10127249	5485593	843937	5473463	819678

Table 6: Instances details

6.1 Continuous Relaxation Results

In this section, we compare VNFPR with SP results on linear relaxation. We also investigate the effect of the valid inequalities presented in Section 5.

In the following, we present aggregated results to highlight the impact of different capacities. Detailed results are reported in the Appendix 7.

In Figure 11 the percentage difference between the continuous relaxation (CR) of SP and VNFPR (i.e., $100\% * \frac{CR_{sp} - CR_{vnfpr}}{CR_{sp}}$) is reported. The instances with the same capacity levels are grouped and the average value on the 16 topologies is reported. Computational times are reported in Figure 12 and Table 7.

SP formulation always provides a CR that is better than or equal to the one provided by VNFPR. The difference is small when link capacity is high or medium, but in the case of low link capacity, the average difference between the CR values reaches almost 40%.

If we look into the detail of the CR obtained results, we can observe that when the service capacity is high, and the link capacity is high or medium (as shown in Tables 19-20), VNFPR and SP get the same solution in all instances but one and both formulations provide a solution equal to 1, that is a trivial bound (at least one service is necessary) valid for any instance. There is an exception: for the network topology *france*, which has an articulation point, SP

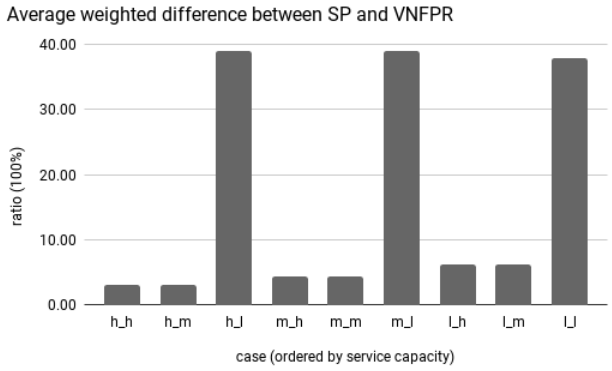


Figure 11: Averaged percentage difference between the CR value obtained by SP and VNFPR

computational time (s)	average		maximum	
	SP	VNFPR	SP	VNFPR
h_h	0.52	38.26	2.38	213.09
h_m	0.52	41.55	2.44	259.70
h_l	183.87	83.16	2272.21	640.95
m_h	4.35	25.45	34.71	184.58
m_m	8.76	25.38	99.26	174.66
m_l	130.36	84.11	1443.63	680.82
l_h	12.42	31.29	74.09	140.87
l_m	14.95	52.48	107.78	361.75
l_l	359.21	122.28	3599.29	936.80

Table 7: Average and maximum computational time of all the tested networks

obtains a larger value (2) than VNFPR. This seems to show that the theoretical bound on number of service implied by the presence of articulation points (see Section 3.2) is, in some sense, taken into account by the SP model, but not by the VNFPR.

Things significantly change when passing to low link capacity (shown in columns one and five of Table 21). In this case, SP finds better bounds in all instances but 4 (*dfn-bwin*,*di-yuan*, *geant*, and *pdh*), and these results can be further improved if we consider rounded values. On the contrary, the bound found by VNFPR remains always the trivial one for all the instances.

A similar behavior can be observed when the service capacity is medium (Table 22-24) or low (Table 25-27). The only difference is that SP improves upon VNFRP in two cases (i.e., network topology *abilene* and *france*) instead of one (i.e., network topology *france*) for the high and medium link capacity. It seems that SP is able to account for the impact of link capacity while VNFRP is not.

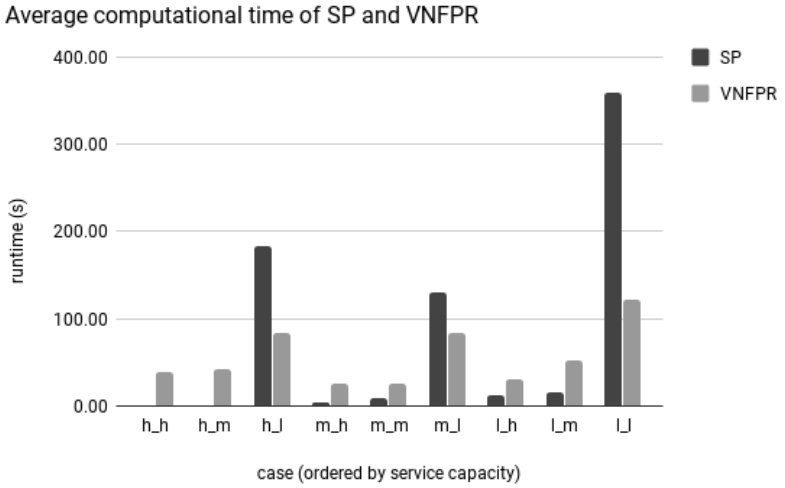


Figure 12: Average computational time of SP and VNFPR

As for the computational time, the two formulations seem to react differently to link and service capacity. As a rough general trend, we may say that they both increase their computational time when service or link capacity decreases, but the impact of link capacity is significant only when passing from high/medium to low capacity. Furthermore, the increasing of the computational time is significant for the SP formulation, where the increase of the average (and maximum) computational time is between one order of magnitude and almost 3 orders of magnitude (in the case of high service capacity, where the average computational times passes from 0.52s to 183.87s).

We run test on the valid inequalities introduced in Section 5. In Table 8 a summary of the VI is reported. In the last column, it is reported if the VI is simply added to the formulation or replaces another constraint[10] [11].

In Table 9, we report a summarized view of the results. We report the short name of the VI (first column), the effect of adding such inequality on the continuous relaxation bound (second column), and the computational impact (third column). We will discuss into details only the results of the most promising VIs.

It seems that the effectiveness of all the valid inequalities is similar for both models, in fact

1. F2, F6 and F7 get the same solution as the formulation without VIs in all instances;

2. F1, F3 and F4 improve upn the formulation without VIs in most of the instances.

[10]Inequalities F1, F2 and F3 are enhanced version of the VNF capacity constraint, so differently from other valid inequalities that are simply added to the formulation, they replace the VNF capacity constraint.

[11]Even if some VI are theoretically dominate by some others, we decided to perform computational tests to verify the numerical behavior of the corresponding formulation, also in terms of computational time.

Name	Test VIs	Test Model	Formulation Action
F1	Eq.16	both	Eq.16 replaces Eq.3
F2	Eq.15	both	Eq.15 replaces Eq.3
F3	Eq.17	both	Eq.17 replaces Eq.3
F4	Eq.18	both	add Eq.18
F6	Eq.19	both	add Eq.19
F7	Eq.20	VNFPR	add Eq.20
F7	Eq.21	SP	add Eq.21

Table 8: Notation of valid inequality test

Name	Bound	Runtime
F1	better in the majority of cases	increased in the majority of cases
F2	unchanged in all cases	unchanged or increased
F3	better in the majority of cases	increased in the majority of cases
F4	better in the majority of cases	decreased in the majority of cases
F6	unchanged in all cases	unchanged or increased
F7 (VNFPR)	unchanged in all cases	unchanged or increased
F7 (SP)	unchanged in all cases	unch./decr. in the maj. of cases

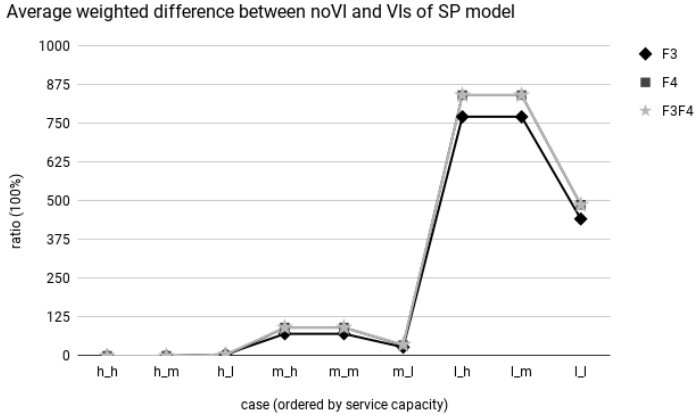
Table 9: General observation of valid inequality test on continuous relaxation

In the following, we discuss in detail the valid inequalities that seemed to be more effective (i.e., F3 and F4) and their coupled valid inequalities F3F4. We did not run other tests on F1 as it is dominated by F3.

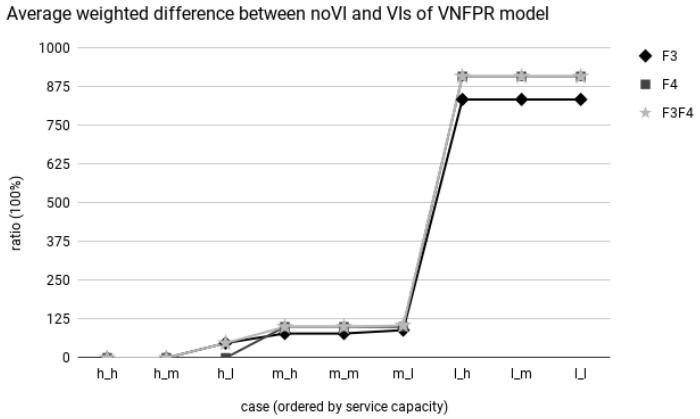
In Figure 13 is reported the percentage improvement obtained adding F3, F4 and F3F4 for both SP and VNFPR formulations, calculated as $100\% * \frac{CR_{form_{noVI}} - CR_{form_{VI}}}{CR_{form_{noVI}}}$. Figures are in logarithmic scale. The improvement depends on link and service capacities: in fact it is zero for the high service capacity and rises up to about 800% for the low service capacity. We remark that F4 alone and coupled with F3 (F3F4) seems to have a similar behavior. Furthermore, in the case where both service and link capacity are low, all VI seem less effective than in the other high service capacity instances for the SP formulation.

Computational times are reported in Figure 15 for the formulation without VIs, with F3 and F4, both alone and coupled. F4 seems to reduce the computational times in all cases, but the high service capacity and low link capacity with VNFPR formulation. The coupling of F3 and F4 is in general less computational expensive than the formulation without VI or with only F3, but with some exceptions.

We now compare the CR bound obtained by SP and VNFPR with the addition of VIs. Figure 14 reports the percentage difference between the two CR ($100\% * \frac{CR_{SP_{VI_i}} - CR_{VNFPR_{VI_i}}}{CR_{SP_{VI_i}}}$). We can observe that the difference is reduced w.r.t. the formulation without VI, in particular for the low service capacities, the CR bound obtained with the two models with added VI is always the same. For the medium service capacity and high/medium link capacity, adding F3 reduce significantly the difference, and adding F4 or F3F4 makes this different zero. F4 proves ineffective only for the h_l case. To conclude, F3, F4, and F3F4



(a) SP



(b) VNFPR

Figure 13: Percentage improvement in CR bound obtained using F3, F4 and F3F4

strengthen the initial formulation, since they improve the lower bound (with both SP and VNFPR). F3F4 provides the best bounds. Furthermore, it seems that these VIs help to reduce the gap between SP and VNFP model.

Figure 15 shows the average computational times. With SP formulation, F3 is in general more time consuming thant F4 and F3F4. With hVNFPR instead there is not a clear winner.

In most cases, F4 (with both SP and VNFPR) needs significantly smaller computational time than the formulation without VIs and with F3, especially when the service capacity is low. Indeed, it requires no more than 15s to solve low service capacity cases for both SP and VNFPR. When the service capacity is medium or high and the link capacity is medium or high, F4 with SP requires more time than the formulation without VIs, whereas F4 with VNFPR requires always less time than the initial formulation, except for case h_l.

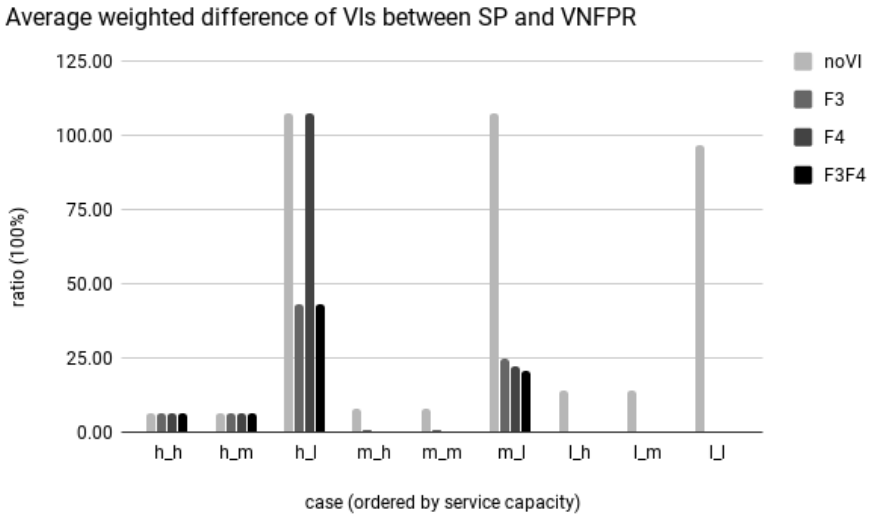


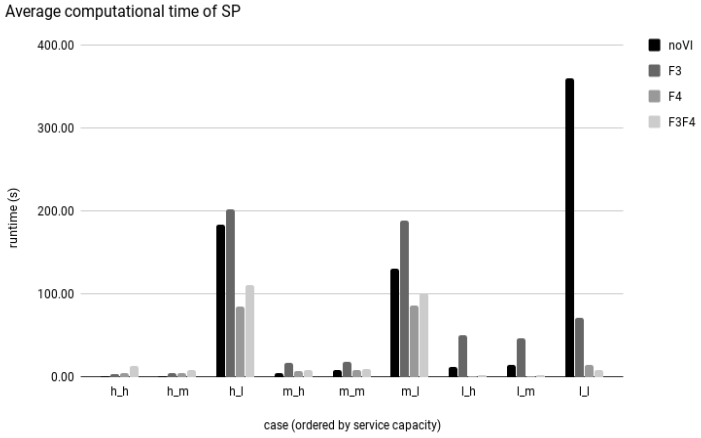
Figure 14: Average percentage difference between SP model and VNFPR model

As for the coupled valid inequalities F3F4, when the capacities are loose, they may need several seconds more than the formulation without VIs. But when the capacities become strict, they are the fastest.

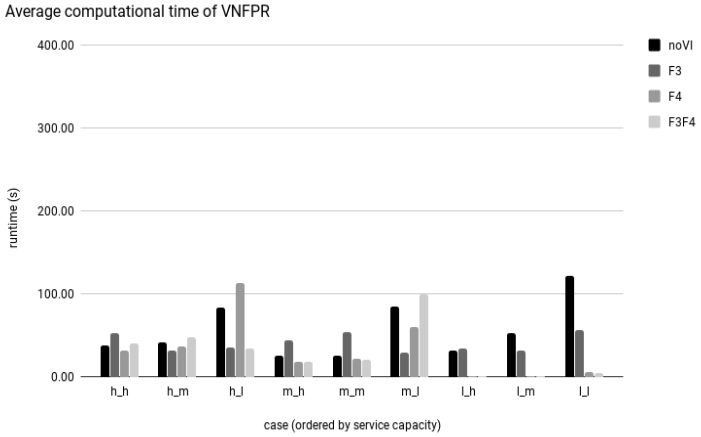
6.2 ILP results

Let us now compare the behavior of the two formulations when solving the integer problem. In Figure 16 (and 17, respectively) the number of instances for which SP (and VNFPR, respectively) finds the optimal and a feasible solution is reported. In general SP performs better than VNFPR. In fact, SP finds the optimal solution in all cases but those for which the link capacity is low. When link capacity is low it is not able to find the optimal solution only in 30% of the instances. Instead, the VNFPR formulation finds the optimal solution in 50% of the instances only for the h_h case. For all the other capacity cases, the percentage of instances solved to optimality is always below 50%. When the service capacity is low VNFPR solves to optimality 7 or 8 instances out of 16, but the number decreases when the service capacity is medium and for the h_m and h_l cases where at most 25% of the instances are solved to optimality.

As for the feasible solution, when SP can provide a feasible solution it is almost always the optimal one, but in one instance for the h_l case, whose optimality is not proved. On the other hand, it is not always able to provide a feasible solution when the link capacity is low: in fact, it cannot provide a feasible solution for 4 instances for the h_l case, and for 5 instances each for the m_l and l_l cases. The performance of VNFPR is significantly worse. For all the cases but h_h, l_h and l_m, it cannot provide a feasible solution in more than half of the instances. For the h_h one case it cannot provide any feasible solution in three instances out of 16, while for the l_h and l_m cases it provides a feasible solution in half of the instances. It seems that the low link capacity



(a) SP



(b) VNFPR

Figure 15: Computational time with and without VIs

makes the instances more challenging for both formulations.

Table 10 reports the minimum, average and maximum gap with respect to the dual bound provided by the formulation (upper table) and the best dual bound known (lower table). When the formulations fail in finding the optimal solution, the percentage gap is relevant: it always above 25% and may rise up to more than 300%. Results show also that the dual bound provided by VNFPR is worse than the one provided by SP.

As for the computational time (as shown in Figure 18), SP is much faster than VNFPR in finding optimal solutions: it requires always less than 10 minutes, while VNFPR may require up to 2000 seconds.

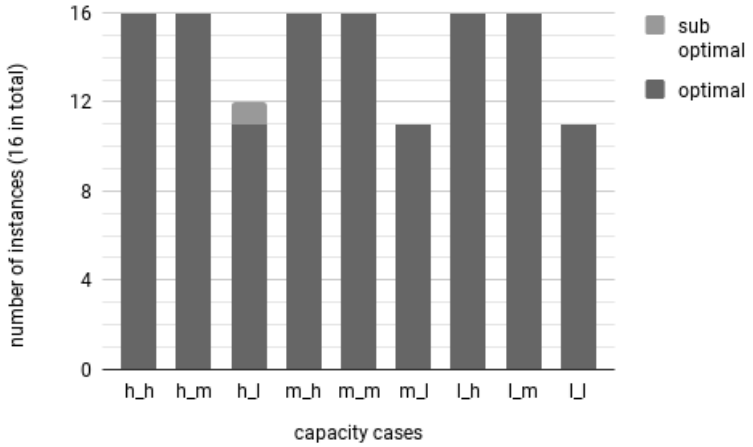


Figure 16: Number of optimal and feasible solutions found by SP

6.2.1 Impact of valid inequalities

We now analyze the impact of adding valid inequalities to the two formulations. In Figures 19 and 20 report the number of optimal and feasible solution found without and with additional inequalities. Adding VI to SP formulation does not increase significantly the number of optimal and feasible solutions found. The best performance is provided by adding F3 and F4: although it is able to found three optimal solution more for the l_l case, it also prevents to prove optimality in two instances for the m_l case. VNFPR does not benefit significantly from the VI as well.There is not a clear winner among the VIs and they may provide both an increase or a decrease of the number of optimal and feasible solutions found with respect to the formulation without VI. In the best case, F3F4 allows to find three more optimal solutions (case m_h). The benefit in terms of feasible solutions is less significant.

Average computational times required for finding the optimal solution are reported in Table 11. SP greatly benefits from the VIs (F4 and F3F4) especially when the service capacity is medium or low. In fact adding the VIs may reduce up to 10 times the required computational times, without increasing it too much when solving the high service capacity cases. However, there is not a clear winner between F4 and F3F4: F4 performs better on the medium service capacity case, while F3F4 on the low service capacity case. The outcome is somehow similar for the VNFPR formulation: although there is not a clear winner between a F4 and F3F4, and despite they may also cause an increase of the computational time, adding F4 or F3F4 helps in general to reduce the computational times. Comparing results reported in Tables 11 and 12 shows that the time needed to solve to optimality benefits more than the overall average time, which is anyway in general improved.

As far as the gap is concern (see Tables 13 and 14), adding VIs may both improve or worsen the gap, depending on the capacity case and on the added VIs.

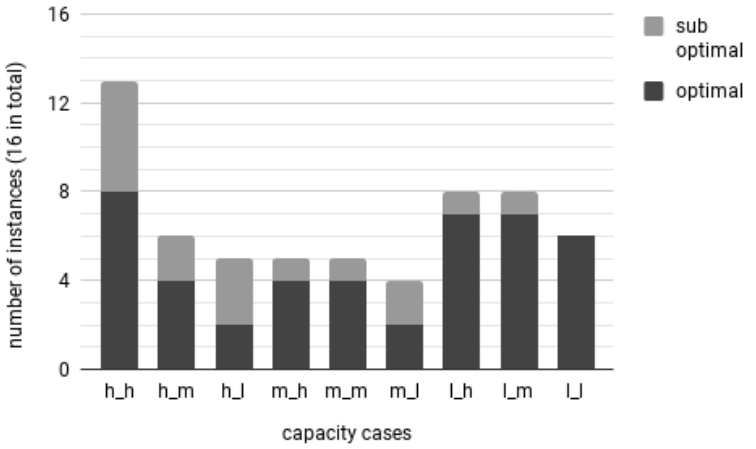


Figure 17: Number of optimal and feasible solutions found by VNFPR

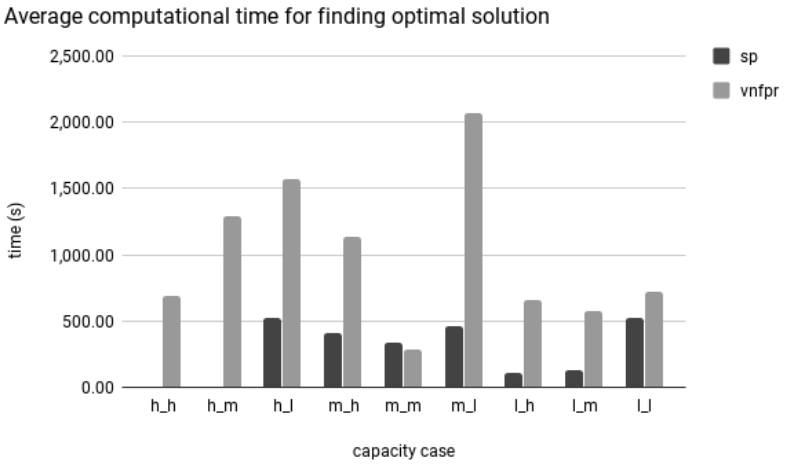
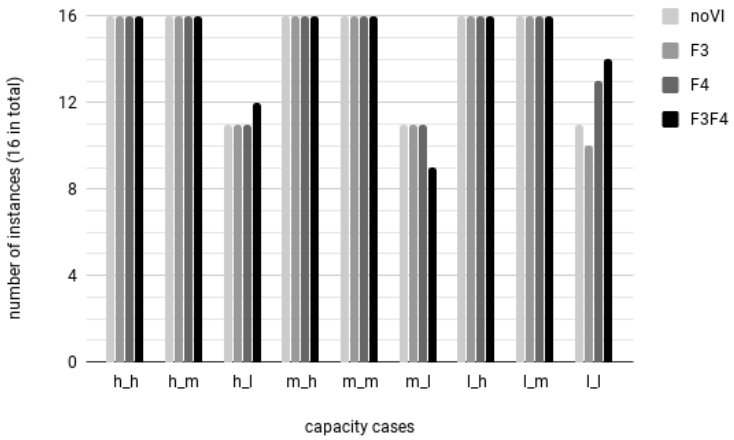
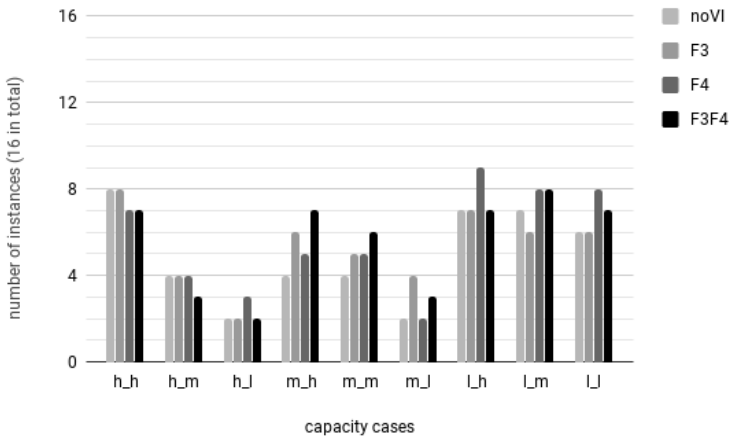


Figure 18: Average computational time for finding optimal integer solution of SP and VNFPR

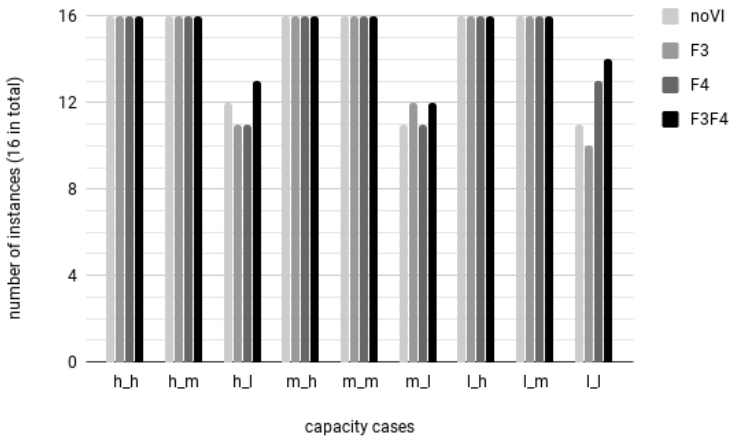


(a) SP - Ratio of optimal solutions

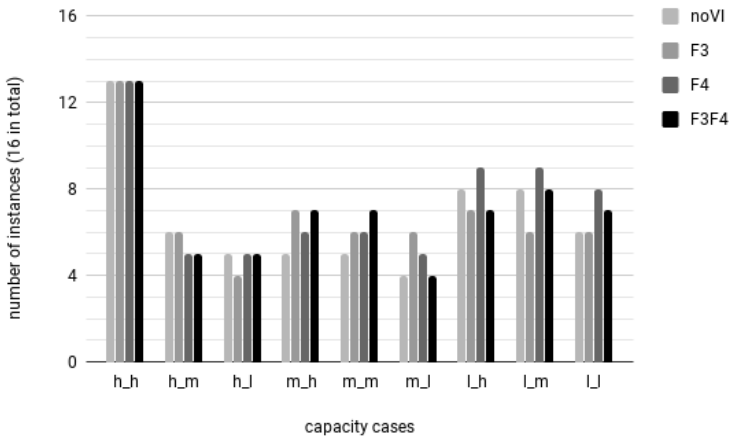


(b) VNFPR - Number of optimal solutions

Figure 19: Number of optimal integer solutions found by VIs of SP and VNFPR



(a) SP - Number of feasible solutions



(b) VNFPR - Number of feasible solutions

Figure 20: Number of integer solutions found by VIs of SP and VNFPR

Comparison with individual lower bound						
	SP			VNFPR		
case	min	average	max	min	average	max
h_h	-	-	-	100	360	500
h_m	-	-	-	100	200	300
h_l	25.9	25.9	25.9	25	263.37	600
m_h	-	-	-	260	260	260
m_m	-	-	-	200	200	200
m_l	-	-	-	75.47	197.74	320
l_h	-	-	-	60	60	60
l_m	-	-	-	60	60	60
l_l	-	-	-	-	-	-
Comparison with best lower bound						
	SP			VNFPR		
case	min	average	max	min	average	max
h_h	-	-	-	100	310	500
h_m	-	-	-	100	200	300
h_l	25.9	25.9	25.9	25	225	600
m_h	-	-	-	200	200	200
m_m	-	-	-	200	200	200
m_l	-	-	-	50	150	250
l_h	-	-	-	33.33	33.33	33.33
l_m	-	-	-	33.33	33.33	33.33
l_l	-	-	-	-	-	-

Table 10: Minimum, average and maximum relative optimality gap of feasible integer solutions found by SP and VNFPR

	SP				VNFPR			
cases	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
h_h	**7.9**	8.2	10.55	10.69	692.24	965.22	**289.44**	291.34
h_m	**3.46**	3.55	7.58	7.59	1287.55	686.16	320.62	**27.65**
h_l	**522.39**	548.88	769.83	611.67	1573.3	**82.91**	1458.04	189.46
m_h	407.18	154.61	**16.68**	42.91	1140.16	370.5	144.57	**100.17**
m_m	340.57	243.04	**18.11**	133.91	280.84	610.48	**44.12**	530.89
m_l	462.48	517.11	**295.65**	301.24	2069.98	**644.49**	1419.89	1497.1
l_h	113.19	52.63	10.69	**6.69**	663.12	103.31	**47.72**	231.76
l_m	127.97	84.12	8.31	**5.19**	580.07	328.97	114.83	**69.42**
l_l	522.38	481.63	244.65	**97.19**	717.97	**514.47**	570.68	764.68

Table 11: Average computational time (in seconds) for finding optimal integer solution by VIs of SP and VNFPR

	SP				VNFPR			
cases	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
h_h	**7.9**	8.2	10.55	10.69	2145.71	2,281.83	**1954.50**	1955.28
h_m	**3.46**	3.55	7.58	3.46	2925.51	2,777.00	**2515.45**	2664.33
h_l	**1306.85**	1502.05	1654.08	1358.37	3345.90	2977.21	3198.55	**2942.95**
m_h	407.18	154.61	**16.68**	42.91	2846.93	2243.88	2412.90	**1933.88**
m_m	340.57	243.04	**18.11**	133.91	2754.87	2523.80	**2430.72**	2,448.32
m_l	1399.68	1480.34	**1171.17**	1711.85	3393.01	**2698.46**	3254.05	3205.09
l_h	113.19	52.63	10.69	**6.69**	1630.64	1729.39	**1601.22**	2125.89
l_m	127.97	84.12	8.31	**5.19**	1787.97	**1773.13**	1856.18	1833.81
l_l	1483.83	1596.49	797.20	**534.64**	2149.36	2208.02	**2065.26**	2349.69

Table 12: Average computational time (in seconds) for for all instances by VIs of SP and VNFPR considering also cases where no solution is found before the timelimit (3600s)

Computed with individual lower bound								
	SP				VNFPR			
case	noVI	VI3	VI4	VI3VI4	noVI	VI3	VI4	VI3VI4
h_h	-	-	-	-	360	360	383.3	416.7
h_m	-	-	-	-	200	300	300	200
h_l	25.9	-	-	29.7	263.4	326.5	236.2	209.2
m_h	-	-	-	-	260	150	100	-
m_m	-	-	-	-	200	150	50	150
m_l	-	27.3	-	76.6	197.7	210.5	116.7	200
l_h	-	-	-	-	60	-	-	-
l_m	-	-	-	-	60	-	16.7	-
l_l	-	-	-	-	-	-	-	33.3
Computed with best lower bound between SP and VNFPR								
	SP				VNFPR			
case	noVI	VI3	VI4	VI3VI4	noVI	VI3	VI4	VI3IV4
h_h	-	-	-	-	310	300	300	333.3
h_m	-	-	-	-	200	300	300	200
h_l	25.9	-	-	29.7	225	300	212.5	183.3
m_h	-	-	-	-	200	150	100	-
m_m	-	-	-	-	200	150	50	150
m_l	-	27.3	-	76.6	150	137.5	83.3	200
l_h	-	-	-	-	33.3	-	-	-
l_m	-	-	-	-	33.3	-	16.7	-
l_l	-	-	-	-	-	-	-	33.3

Table 13: Average relative optimality gap of feasible integer solutions found by VIs

Computed with individual lower bound								
	SP				VNFPR			
case	noVI	VI3	VI4	VI3VI4	noVI	VI3	VI4	VI3VI4
h_h	-	-	-	-	500	500	900	900
h_m	-	-	-	-	300	500	300	300
h_l	25.9	-	-	29.7	600	500	400	500
m_h	-	-	-	-	260	150	100	-
m_m	-	-	-	-	200	150	50	150
m_l	-	27.3	-	76.6	320	260	200	200
l_h	-	-	-	-	60	-	-	-
l_m	-	-	-	-	60	-	16.7	-
l_l	-	-	-	-	-	-	-	33.3
Computed with best lower bound between SP and VNFPR								
	SP				VNFPR			
case	noVI	VI3	VI4	VI3VI4	noVI	VI3	VI4	VI3IV4
h_h	-	-	-	-	500	500	800	800
h_m	-	-	-	-	300	500	300	300
h_l	25.9	-	-	29.7	600	500	400	500
m_h	-	-	-	-	200	150	100	-
m_m	-	-	-	-	200	150	50	150
m_l	-	27.3	-	76.6	250	200	200	200
l_h	-	-	-	-	33.3	-	-	-
l_m	-	-	-	-	33.3	-	16.7	-
l_l	-	-	-	-	-	-	-	33.3

Table 14: Maximum relative optimality gap of feasible integer solutions found by VIs

Computed with individual lower bound								
	SP				VNFPR			
case	noVI	VI3	VI4	VI3VI4	noVI	VI3	VI4	VI3VI4
h_h	-	-	-	-	100	100	100	100
h_m	-	-	-	-	100	100	300	100
h_l	25.9	-	-	29.7	25	153	72.4	33.3
m_h	-	-	-	-	260	150	100	-
m_m	-	-	-	-	200	150	50	150
m_l	-	27.3	-	75.5	197.74	161	50	200
l_h	-	-	-	-	60	-	-	-
l_m	-	-	-	-	60	-	16.7	-
l_l	-	-	-	-	-	-	-	33.3
Computed with best lower bound between SP and VNFPR								
	SP				VNFPR			
case	noVI	VI3	VI4	VI3VI4	noVI	VI3	VI4	VI3IV4
h_h	-	-	-	-	100	100	100	100
h_m	-	-	-	-	100	100	300	100
h_l	25.9	-	-	29.7	25	100	25	0
m_h	-	-	-	-	200	150	100	-
m_m	-	-	-	-	200	150	50	150
m_l	-	27.3	-	76.6	50	75	0	200
l_h	-	-	-	-	33.3	-	-	-
l_m	-	-	-	-	33.3	-	16.7	-
l_l	-	-	-	-	-	-	-	33.3

Table 15: Minimum relative optimality gap of feasible integer solutions found by VIs

6.3 Biconnected components and articulation points results

We present in this section the results of tests on property of biconnected components and articulation points. The idea is to improve the lower bound and reduce the computational time by applying the property of biconnected components and articulation points (i.e., Property 3.2).

As have been discussed in Section 3.2, if a given separable graph contains at least an *articulation point*, a lower bound on the number of VNF may be obtained. More precisely, according to Property 3.2, if for each biconnected component, there exists at least one demand whose origin and destination nodes are in the biconnected component itself, then when the number and structure of biconnected components are known, combining this information with the informations related to the source and destination of the demands, it is consequently possible to determine a minimum number of services to install. Moreover, there exists an optimal solution where the service is located on the articulation points.

In order to apply this property to improve the problem resolution, we propose to run a preprocessing before running the problem models, which aims at

1. detecting the number and structure of biconnected components, thereby installing one service on each articulation point
2. eliminating the assignment of demand to the VNF which is out of the components that demand belongs to, to help find the optimal routing solution for each demand.

To examine the performance of our proposal, we run linear relaxation and integer linear programming tests on the models with preprocessing. We perform tests on 3 network instances of SNDLib (i.e., *france, ta2 and zib54*), which contain at least one articulation point.

Table 16 shows the lower bounds found by the linear relaxation after applying the preprocessing procedure. For SP model, compared to the initial model, the preprocessing procedure helps to improve the lower bound in the low link capacity cases. But not enough to improve with respect of the rounded up lower bound. While for VNFPR model, the preprocessing contributes greatly to the computation of lower bounds. The initial VNFPR model finds always the trivial bound (i.e., 1). After adding the preprocessing, VNFPR model finds better bounds but never better than the ones found by the SP model with preprocessing.

As for the computational time (as shown in Table 17), we observe that applying the preprocessing will greatly reduce the runtime for both SP and VNFPR models. For example, in h_h case, VNFPR with preprocessing uses just 12 seconds to find the lower bound of value 2, while the initial model spends up to two orders of magnitude higher computational times to find the lower bound of value 1.

Table 18 shows the results of integer linear programming test on the preprocessing procedure under SP formulation, for all instances where as feasible

solution was found in the time limit. VNFPR only finds feasible solution for instance *france* in h_h case, therefore we do not report the results of VNFPR model.

As shown in Table 18, the number of optimal solution found by SP model with preprocessing procedure is the same as the initial SP model. Moreover, it appears that when the initial model cannot find any feasible solutions, the model with preprocessing procedure cannot find any feasible solutions neither. But with respect of computational time, applying the preprocessing can reduce the computational time. Applying the VIs F3F4 allows to solve at optimality instances that cannot be solved using the initial formulation, with or without the preprocessing. Even when we consider the formulation with F3F4, the preprocessing reduces the computational time in almost all the instances. Furthermore, in a single instance zib54-l_l, the combination of F3F4 and preprocessing allows to find the optimal solution.

		SP		VNFPR	
Case	Instance	NoPrep	Prep	NoPrep	Prep
h_h	france	**2**	**2**	1	**2**
	ta2	**2**	**2**	none	**2**
	zib54	**2**	**2**	1	**2**
h_m	france	**2**	**2**	1	**2**
	ta2	**2**	**2**	none	**2**
	zib54	**2**	**2**	1	**2**
h_l	france	**2.94**	**2.94**	1	2
	ta2	none	none	none	none
	zib54	none	none	none	none
m_h	france	**2**	**2**	1	**2**
	ta2	**2**	**2**	none	**2**
	zib54	**2**	**2**	none	**2**
m_m	france	**2**	**2**	1	**2**
	ta2	**2**	**2**	none	**2**
	zib54	**2**	**2**	none	**2**
m_l	france	**2.94**	**2.94**	1	2
	ta2	none	none	none	none
	zib54	none	none	none	none
l_h	france	2.53	**2.85**	1	**2.85**
	ta2	none	**2.94**	none	none
	zib54	2.68	**2.92**	none	none
l_m	france	2.53	**2.85**	1	**2.85**
	ta2	2.75	**2.94**	none	none
	zib54	2.68	**2.92**	none	none
l_l	france	3.36	**3.8**	1	2.85
	ta2	none	none	none	none
	zib54	none	none	none	none

Table 16: Results of bounds of continuous relaxation test on preprocessing under both SP and VNFPR models. *Prep* refers to the model with preprocessing, *NoPrep* refers to the model without preprocessing. *none* indicates no solutions found within time limit of 3600s.

		SP		VNFPR	
Case	Instance	NoPrep	Prep	NoPrep	Prep
h_h	france	0.53	**0.2**	7.23	0.42
	ta2	15.27	**14.49**	TL	25.9
	zib54	7.71	**7.55**	2764.02	12.13
h_m	france	0.57	**0.2**5	11.68	0.46
	ta2	13.61	**13.51**	TL	24.67
	zib54	7.34	**7.17**	3119.98	13.05
h_l	france	4.45	1.88	46.71	**1.73**
	ta2	TL	TL	TL	TL
	zib54	TL	TL	TL	TL
m_h	france	2.3	**0.44**	10.28	0.82
	ta2	75.08	**21.24**	TL	32.24
	zib54	23.55	**8.05**	TL	13.81
m_m	france	1.72	**0.41**	10.16	0.64
	ta2	60.83	**17.59**	TL	26.99
	zib54	22.86	**8.07**	TL	14.28
m_l	france	6.1	**1.99**	40.86	2.33
	ta2	TL	TL	TL	TL
	zib54	TL	TL	TL	TL
l_h	france	**5.12**	6.55	14.87	8.85
	ta2	TL	**1200.03**	TL	TL
	zib54	**937.45**	1136.88	TL	TL
l_m	france	9.92	**5.89**	24.05	12.3
	ta2	2225.98	**1077.96**	TL	TL
	zib54	1180.01	**650.15**	TL	TL
l_l	france	68.25	71.42	48.84	**38.66**
	ta2	TL	TL	TL	TL
	zib54	TL	TL	TL	TL

Table 17: Computational time (s) of continuous relaxation test on preprocessing under SP and VNFPR model. *Prep* refers to the model with preprocessing, *NoPrep* refers to the model without preprocessing. *TL* indicates that the computational time reaches the time limit of 3600s.

		Integer Solution				Computational time			
		noVI		F3F4		noVI		F3F4	
Case	Instance	NoPrep	Prep	NoPrep	Prep	NoPrep	Prep	NoPrep	Prep
h_h	france	**2**	**2**	**2**	**2**	**1.57**	1.59	2.39	1.73
	ta2	**2**	**2**	**2**	**2**	188.95	**67.77**	263.28	68.86
	zib54	**2**	**2**	**2**	**2**	90.69	**87.37**	147.25	91.27
h_m	france	**2**	**2**	**2**	**2**	2.19	**1.44**	1.88	1.50
	ta2	**2**	**2**	**2**	**2**	184.13	**66.23**	180.13	68.38
	zib54	**2**	**2**	**2**	**2**	138.30	**41.76**	83.34	42.67
m_h	france	**2**	**2**	**2**	**2**	2.58	1.77	2.32	**1.68**
	ta2	**2**	**2**	**2**	**2**	184.13	**66.23**	180.13	68.38
	zib54	**2**	**2**	**2**	**2**	149.73	**85.24**	537.85	86.53
m_m	france	**2**	**2**	**2**	**2**	2.71	**1.59**	2.61	1.82
	ta2	**2**	**2**	**2**	**2**	197.49	**80.46**	1125.35	83.40
	zib54	**2**	**2**	**2**	**2**	89.33	44.87	563.79	**43.02**
l_h	france	**13**	**13**	**13**	**13**	334.21	29.48	**5.85**	14.67
	ta2	none	none	**33**	**33**	TL	TL	720.00	**671.91**
	zib54	none	none	**26**	**26**	TL	TL	306.96	**194.70**
l_m	france	**13**	**13**	**13**	**13**	139.67	64.20	**6.71**	21.58
	ta2	none	none	**33**	**33**	TL	TL	1113.04	**722.41**
	zib54	none	none	**26**	**26**	TL	TL	**220.61**	228.41
l_l	france	none	none	none	none	TL	TL	TL	TL
	ta2	none	none	none	none	TL	TL	TL	TL
	zib54	none	none	none	**26**	TL	TL	TL	**2253.56**

Table 18: Results of preprocessing test under both initial SP formulation (namely *noVI*) and SP formulation with valid inequalities F3 and F4 (namely *F3F4*). *Prep* refers to the model with preprocessing, *NoPrep* refers to the model without preprocessing. *TL* indicates that the computational time reaches the time limit of 3600s.

7 Conclusions

This paper addresses a Virtual Network Function Placement and Routing problem, where each demand can be routed only through a simple path. The paper discusses complexity properties of different versions of the problem, that differ with respect to capacity limits or network topology. Two formulations are compared which are inspired on the two main modelling strategies proposed in the literature. The first modelling strategy (SP) is based on the splitting of the path into two sub-paths, one connecting the source with the reference function, the second one connecting the reference function with the destination. The second one (VNFPR) uses arc flow variables and forbids cycles exploiting node labels. Valid inequalities are added and their impact evaluated. Results show that stand alone SP improves upon VNFPR, which in turn benefits more from the VIs. It is important however to point out that SP can be applied thanks to some assumptions on the problem features, such as the fixed and given order of the network functions. When such assumption do not hold, namely each demand is to be served by a set of functions, but the order according to which it must cross the function is not given, the number of sub-paths needed by SP increases exponentially, while VNFPR does not experience such a drawback. Thus, as future research, we suggest to generalize the problem and to investigate dynamic addition of sub-paths in SP-like formulations.

References

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

[2] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin. Virtual function placement for service chaining with partial orders and anti-affinity rules. *CoRR*, abs/1705.10554, 2017.

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

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

[5] Ivan Contreras and Elena Fernndez. General network design: A unified view of combined location and network design problems. *European Journal of Operational Research*, 219(3):680 – 697, 2012. Feature Clusters.

[6] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, 15(4):1888–1906, Fourth 2013.

[7] M. Gao, B. Addis, D., M Bouet, and S. Secci. Optimal orchestration of virtual network functions. *Available on HAL, Submitted for publication*, 2017.

[8] 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. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–7, May 2014.

[9] J. Hwang, K. K. Ramakrishnan, and T. Wood. Netvm: High performance and flexible networking using virtualization on commodity platforms. *IEEE Transactions on Network and Service Management*, 12(1):34–47, March 2015.

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

[11] Tamás Lukovszki and Stefan Schmid. Online admission control and embedding of service chains. In *Post-Proceedings of the 22Nd International Colloquium on Structural Information and Communication Complexity - Volume 9439*, SIROCCO 2015, pages 104–118, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

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

[13] 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)*, pages 1–9, April 2015.

[14] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. Sndlib 1.0—survivable network design library. *Netw.*, 55(3):276–286, May 2010.

[15] J. Soares, M. Dias, J. Carapinha, B. Parreira, and S. Sargento. Cloud4nfv: A platform for virtual network functions. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 288–293, Oct 2014.

Appendix

In the appendix we report tables with detailed results for both the continuous relaxation and the ILP with and without the addition of VIs.

7.1 Continuous relaxation results

Continuous relaxation results are summarized in Tables 19-27. Each table presents results for a set of instances with a given service and link capacity couple.

High service capacity and high link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
atlanta	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
dfn-bwin	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
di-yuan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
france	2.00	2.00	2.00	2.00	1.00	1.00	1.00	1.00
geant	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
janos-us	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
newyork	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
nobel-eu	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
nobel-germany	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
nobel-us	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
norway	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
pdh	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
polska	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
sun	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
ta1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 19: Continuous relaxation bounds of F3, F4 and F3F4 (case h_h)

High service capacity and medium link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
atlanta	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
dfn-bwin	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
di-yuan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
france	2.00	2.00	2.00	2.00	1.00	1.00	1.00	1.00
geant	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
janos-us	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
newyork	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
nobel-eu	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
nobel-germany	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
nobel-us	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
norway	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
pdh	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
polska	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
sun	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
ta1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 20: Continuous relaxation bounds of F3, F4 and F3F4 (case h_m)

High service capacity and low link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	2.00	2.00	2.00	2.00	1.00	1.00	1.00	1.00
atlanta	2.22	2.22	2.22	2.22	1.00	1.42	1.00	1.42
dfn-bwin	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
di-yuan	1.00	1.02	1.00	1.02	1.00	1.02	1.00	1.02
france	2.94	2.94	2.94	2.94	1.00	1.29	1.00	1.29
geant	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
janos-us	3.65	3.65	3.65	3.65	1.00	1.97	1.00	1.97
newyork	1.72	2.41	1.72	2.41	1.00	2.37	1.00	2.37
nobel-eu	1.99	1.99	1.99	1.99	1.00	1.59	1.00	1.59
nobel-germany	2.31	2.31	2.31	2.31	1.00	1.20	1.00	1.20
nobel-us	2.20	2.29	2.20	2.29	1.00	2.11	1.00	2.11
norway	4.49	4.49	4.49	4.49	1.00	2.21	1.00	2.21
pdh	1.00	1.14	1.00	1.14	1.00	1.14	1.00	1.14
polska	3.24	3.24	3.24	3.24	1.00	1.90	1.00	1.90
sun	1.32	1.52	1.32	1.52	1.00	1.42	1.00	1.42
ta1	1.09	1.11	1.09	1.11	1.00	1.00	1.00	1.00

Table 21: Continuous relaxation bounds of F3, F4 and F3F4 (case h_l)

Medium service capacity and high link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	1.24	1.71	2.00	2.00	1.00	1.71	2.00	2.00
atlanta	1.00	1.76	2.00	2.00	1.00	1.76	2.00	2.00
dfn-bwin	1.00	1.67	2.00	2.00	1.00	1.67	2.00	2.00
di-yuan	1.00	1.71	2.00	2.00	1.00	1.71	2.00	2.00
france	2.00	2.00	2.00	2.00	1.00	1.85	2.00	2.00
geant	1.00	1.83	2.00	2.00	1.00	1.83	2.00	2.00
janos-us	1.00	1.86	2.00	2.00	1.00	1.86	2.00	2.00
newyork	1.00	1.78	2.00	2.00	1.00	1.78	2.00	2.00
nobel-eu	1.00	1.87	2.00	2.00	1.00	1.87	2.00	2.00
nobel-germany	1.00	1.79	2.00	2.00	1.00	1.79	2.00	2.00
nobel-us	1.00	1.75	2.00	2.00	1.00	1.75	2.00	2.00
norway	1.00	1.86	2.00	2.00	1.00	1.86	2.00	2.00
pdh	1.00	1.69	2.00	2.00	1.00	1.69	2.00	2.00
polska	1.00	1.71	2.00	2.00	1.00	1.71	2.00	2.00
sun	1.00	1.87	2.00	2.00	1.00	1.87	2.00	2.00
ta1	1.00	1.85	2.00	2.00	1.00	1.85	2.00	2.00

Table 22: Continuous relaxation bounds of F3, F4 and F3F4 (case m_h)

Medium service capacity and medium link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	1.24	1.71	2.00	2.00	1.00	1.71	2.00	2.00
atlanta	1.00	1.76	2.00	2.00	1.00	1.76	2.00	2.00
dfn-bwin	1.00	1.67	2.00	2.00	1.00	1.67	2.00	2.00
di-yuan	1.00	1.71	2.00	2.00	1.00	1.71	2.00	2.00
france	2.00	2.00	2.00	2.00	1.00	1.85	2.00	2.00
geant	1.00	1.83	2.00	2.00	1.00	1.83	2.00	2.00
janos-us	1.00	1.86	2.00	2.00	1.00	1.86	2.00	2.00
newyork	1.00	1.78	2.00	2.00	1.00	1.78	2.00	2.00
nobel-eu	1.00	1.87	2.00	2.00	1.00	1.87	2.00	2.00
nobel-germany	1.00	1.79	2.00	2.00	1.00	1.79	2.00	2.00
nobel-us	1.00	1.75	2.00	2.00	1.00	1.75	2.00	2.00
norway	1.00	1.86	2.00	2.00	1.00	1.86	2.00	2.00
pdh	1.00	1.69	2.00	2.00	1.00	1.69	2.00	2.00
polska	1.00	1.71	2.00	2.00	1.00	1.71	2.00	2.00
sun	1.00	1.87	2.00	2.00	1.00	1.87	2.00	2.00
ta1	1.00	1.85	2.00	2.00	1.00	1.85	2.00	2.00

Table 23: Continuous relaxation bounds of F3, F4 and F3F4 (case m_m)

Medium service capacity and low link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	2.00	2.00	2.00	2.00	1.00	1.71	2.00	2.00
atlanta	2.22	2.22	2.22	2.22	1.00	1.76	2.00	2.00
dfn-bwin	1.00	1.67	2.00	2.00	1.00	1.67	2.00	2.00
di-yuan	1.00	1.71	2.00	2.00	1.00	1.71	2.00	2.00
france	2.94	2.94	2.94	2.94	1.00	1.85	2.00	2.00
geant	1.00	1.83	2.00	2.00	1.00	1.83	2.00	2.00
janos-us	3.65	3.65	3.65	3.65	1.00	2.02	2.00	2.02
newyork	1.72	2.41	2.00	2.41	1.00	2.37	2.00	2.37
nobel-eu	1.99	1.99	2.00	2.00	1.00	1.87	2.00	2.00
nobel-germany	2.31	2.31	2.31	2.31	1.00	1.79	2.00	2.00
nobel-us	2.20	2.29	2.20	2.29	1.00	2.11	2.00	2.11
norway	4.49	4.49	4.49	4.49	1.00	2.21	2.00	2.21
pdh	1.00	1.69	2.00	2.00	1.00	1.69	2.00	2.00
polska	3.24	3.24	3.24	3.24	1.00	1.97	2.00	2.00
sun	1.32	1.88	2.00	2.00	1.00	1.87	2.00	2.00
ta1	1.09	1.85	2.00	2.00	1.00	1.85	2.00	2.00

Table 24: Continuous relaxation bounds of F3, F4 and F3F4 (case m_l)

Low service capacity and high link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	1.67	6.00	7.00	7.00	1.00	6.00	7.00	7.00
atlanta	1.00	7.50	8.00	8.00	1.00	7.50	8.00	8.00
dfn-bwin	1.00	5.00	6.00	6.00	1.00	5.00	6.00	6.00
di-yuan	1.00	5.89	6.00	6.00	1.00	5.89	6.00	6.00
france	2.53	12.50	13.00	13.00	1.00	12.50	13.00	13.00
geant	1.00	11.00	12.00	12.00	1.00	11.00	12.00	12.00
janos-us	1.00	13.00	14.00	14.00	1.00	13.00	14.00	14.00
newyork	1.00	8.03	9.00	9.00	1.00	8.03	9.00	9.00
nobel-eu	1.00	14.06	15.00	15.00	1.00	14.06	15.00	15.00
nobel-germany	1.00	8.57	9.00	9.00	1.00	8.57	9.00	9.00
nobel-us	1.00	7.00	8.00	8.00	1.00	7.00	8.00	8.00
norway	1.00	13.51	14.00	14.00	1.00	13.51	14.00	14.00
pdh	1.00	5.50	6.00	6.00	1.00	5.50	6.00	6.00
polska	1.00	6.00	7.00	7.00	1.00	6.00	7.00	7.00
sun	1.00	13.60	14.00	14.00	1.00	13.60	14.00	14.00
ta1	1.00	12.00	13.00	13.00	1.00	12.00	13.00	13.00

Table 25: Continuous relaxation bounds of F3, F4 and F3F4 (case l_h)

Low service capacity and medium link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	1.67	6.00	7.00	7.00	1.00	6.00	7.00	7.00
atlanta	1.00	7.50	8.00	8.00	1.00	7.50	8.00	8.00
dfn-bwin	1.00	5.00	6.00	6.00	1.00	5.00	6.00	6.00
di-yuan	1.00	5.89	6.00	6.00	1.00	5.89	6.00	6.00
france	2.53	12.50	13.00	13.00	1.00	12.50	13.00	13.00
geant	1.00	11.00	12.00	12.00	1.00	11.00	12.00	12.00
janos-us	1.00	13.00	14.00	14.00	1.00	13.00	14.00	14.00
newyork	1.00	8.03	9.00	9.00	1.00	8.03	9.00	9.00
nobel-eu	1.00	14.06	15.00	15.00	1.00	14.06	15.00	15.00
nobel-germany	1.00	8.57	9.00	9.00	1.00	8.57	9.00	9.00
nobel-us	1.00	7.00	8.00	8.00	1.00	7.00	8.00	8.00
norway	1.00	13.51	14.00	14.00	1.00	13.51	14.00	14.00
pdh	1.00	5.50	6.00	6.00	1.00	5.50	6.00	6.00
polska	1.00	6.00	7.00	7.00	1.00	6.00	7.00	7.00
sun	1.00	13.60	14.00	14.00	1.00	13.60	14.00	14.00
ta1	1.00	12.00	13.00	13.00	1.00	12.00	13.00	13.00

Table 26: Continuous relaxation bounds of F3, F4 and F3F4 (case l_m)

Low service capacity and low link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	2.09	6.00	7.00	7.00	1.00	6.00	7.00	7.00
atlanta	2.33	7.50	8.00	8.00	1.00	7.50	8.00	8.00
dfn-bwin	1.00	5.00	6.00	6.00	1.00	5.00	6.00	6.00
di-yuan	1.00	5.89	6.00	6.00	1.00	5.89	6.00	6.00
france	3.36	12.50	13.00	13.00	1.00	12.50	13.00	13.00
geant	1.00	11.00	12.00	12.00	1.00	11.00	12.00	12.00
janos-us	3.71	13.00	14.00	14.00	1.00	13.00	14.00	14.00
newyork	1.72	8.03	9.00	9.00	1.00	8.03	9.00	9.00
nobel-eu	1.99	14.06	15.00	15.00	1.00	14.06	15.00	15.00
nobel-germany	2.31	8.57	9.00	9.00	1.00	8.57	9.00	9.00
nobel-us	2.20	7.00	8.00	8.00	1.00	7.00	8.00	8.00
norway	none	13.51	14.00	14.00	1.00	13.51	14.00	14.00
pdh	1.00	5.50	6.00	6.00	1.00	5.50	6.00	6.00
polska	3.24	6.00	7.00	7.00	1.00	6.00	7.00	7.00
sun	1.32	13.60	14.00	14.00	1.00	13.60	14.00	14.00
ta1	1.24	12.00	13.00	13.00	1.00	12.00	13.00	13.00

Table 27: Continuous relaxation bounds of F3, F4 and F3F4 (case l_l)

7.2 Detailed results of ILP

ILP results are summarized in Tables 28-36. Each table presents results for a set of instances with a given service and link capacity couple. The cases were not integer solution could be found in the timelimit are indicated by *none*. The cases were a formulation produce better solution than the other one (with the same additional VIs) are highlighted in bold.

High service capacity and high link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	1	1	1	1	1	1	1	1
atlanta	1	1	1	1	1	1	1	1
dfn-bwin	1	1	1	1	1	1	1	1
di-yuan	1	1	1	1	1	1	1	1
france	**2**	**2**	**2**	**2**	5	6	10	10
geant	**1**	**1**	**1**	**1**	6	6	2	3
janos-us	**1**	**1**	**1**	**1**	5	none	none	none
newyork	**1**	**1**	**1**	**1**	5	6	9	9
nobel-eu	**1**	**1**	**1**	**1**	none	none	none	none
nobel-germany	1	1	1	1	1	1	2	2
nobel-us	1	1	1	1	1	1	1	1
norway	**1**	**1**	**1**	**1**	none	none	none	none
pdh	1	1	1	1	1	1	1	1
polska	1	1	1	1	1	1	1	1
sun	**1**	**1**	**1**	**1**	5	3	3	3
ta1	**1**	**1**	**1**	**1**	2	2	3	4

Table 28: Integer solution of F3, F4 and F3F4 (case h_h)

High service capacity and medium link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	**1**	1	1	**1**	2	1	1	2
atlanta	**1**	**1**	**1**	**1**	none	none	none	none
dfn-bwin	**1**	**1**	**1**	**1**	4	6	4	4
di-yuan	1	1	1	1	1	1	1	1
france	**2**	**2**	**2**	**2**	none	none	none	none
geant	**1**	**1**	**1**	**1**	none	none	none	none
janos-us	**1**	**1**	**1**	**1**	none	none	none	none
newyork	**1**	**1**	**1**	**1**	none	none	none	none
nobel-eu	**1**	**1**	**1**	**1**	none	none	none	none
nobel-germany	**1**	**1**	**1**	**1**	none	none	none	none
nobel-us	1	1	**1**	**1**	1	1	none	none
norway	**1**	**1**	**1**	**1**	none	none	none	none
pdh	1	**1**	1	1	1	2	1	1
polska	1	1	1	1	1	1	1	1
sun	**1**	**1**	**1**	**1**	none	none	none	none
ta1	**1**	**1**	**1**	**1**	none	none	none	none

Table 29: Integer solution of F3, F4 and F3F4 (case h_m)

High service capacity and low link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	2	2	2	2	2	2	2	2
atlanta	**3**	**3**	**3**	**3**	none	none	none	none
dfn-bwin	**1**	**1**	**1**	**1**	7	5	6	6
di-yuan	**2**	2	2	2	3	2	2	2
france	none	none	none	none	none	none	none	none
geant	**1**	**1**	none	**1**	none	none	none	none
janos-us	**5**	none	**5**	**5**	none	none	none	none
newyork	none	none	none	none	none	none	none	none
nobel-eu	**3**	**3**	**3**	**3**	none	none	none	none
nobel-germany	**3**	**3**	**3**	**3**	none	none	none	none
nobel-us	**4**	**4**	**4**	**4**	none	none	none	none
norway	none	none	none	none	none	none	none	none
pdh	2	2	**2**	**2**	2	2	4	3
polska	none	none	none	4	5	5	none	4
sun	**2**	**2**	**2**	**2**	none	none	none	none
ta1	**2**	**2**	**2**	**2**	none	none	none	none

Table 30: Integer solution of F3, F4 and F3F4 (case h_l)

Medium service capacity and high link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	2	2	2	2	2	2	2	2
atlanta	2	2	2	2	none	none	none	none
dfn-bwin	**2**	**2**	**2**	2	6	5	4	2
di-yuan	2	2	2	2	2	2	2	2
france	**2**	**2**	**2**	**2**	none	none	none	none
geant	**2**	**2**	**2**	**2**	none	none	none	none
janos-us	**2**	**2**	**2**	**2**	none	none	none	none
newyork	**2**	**2**	**2**	**2**	none	none	none	none
nobel-eu	**2**	**2**	**2**	**2**	none	none	none	none
nobel-germany	2	2	2	2	none	2	none	2
nobel-us	2	2	2	2	none	2	2	2
norway	**2**	**2**	**2**	**2**	none	none	none	none
pdh	2	2	2	2	2	2	2	2
polska	2	2	2	2	2	2	2	2
sun	**2**	**2**	**2**	**2**	none	none	none	none
ta1	**2**	**2**	**2**	**2**	none	none	none	none

Table 31: Integer solution of F3, F4 and F3F4 (case m_h)

Medium service capacity and medium link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	2	2	2	2	2	2	2	2
atlanta	**2**	**2**	**2**	**2**	none	none	none	none
dfn-bwin	**2**	**2**	**2**	**2**	6	5	3	5
di-yuan	2	2	2	2	2	2	2	2
france	**2**	**2**	**2**	**2**	none	none	none	none
geant	**2**	**2**	**2**	**2**	none	none	none	none
janos-us	**2**	**2**	**2**	**2**	none	none	none	none
newyork	**2**	**2**	**2**	**2**	none	none	none	none
nobel-eu	**2**	**2**	**2**	**2**	none	none	none	none
nobel-germany	2	2	2	2	none	none	none	2
nobel-us	2	2	2	2	none	2	2	2
norway	**2**	**2**	**2**	**2**	none	none	none	none
pdh	2	2	2	2	2	2	2	2
polska	2	2	2	2	2	2	2	2
sun	**2**	**2**	**2**	**2**	none	none	none	none
ta1	**2**	**2**	**2**	**2**	none	none	none	none

Table 32: Integer solution of F3, F4 and F3F4 (case m_m)

Medium service capacity and low link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	2	2	2	2	2	2	3	2
atlanta	3	3	3	3	none	none	none	none
dfn-bwin	**2**	**2**	**2**	**2**	7	6	6	6
di-yuan	**2**	2	2	2	3	2	2	2
france	none	none	none	none	none	none	none	none
geant	none	2	2	5	none	none	none	none
janos-us	none	5	none	5	none	none	none	none
newyork	none	none	none	none	none	none	none	none
nobel-eu	3	3	3	3	none	none	none	none
nobel-germany	3	3	3	3	none	none	none	none
nobel-us	4	4	4	4	none	7	none	none
norway	none	none	none	none	none	none	none	none
pdh	2	2	2	2	2	2	2	2
polska	4	none	none	none	none	4	4	none
sun	2	2	2	2	none	none	none	none
ta1	2	2	2	2	none	none	none	none

Table 33: Integer solution of F3, F4 and F3F4 (case m_l)

Low service capacity and high link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	7	7	7	7	7	7	7	7
atlanta	8	8	8	8	8	none	8	8
dfn-bwin	**6**	6	6	6	8	6	6	6
di-yuan	6	6	6	6	6	6	6	6
france	13	13	13	13	none	none	none	none
geant	12	12	12	12	none	none	none	none
janos-us	14	14	14	14	none	none	none	none
newyork	9	9	9	9	none	none	none	none
nobel-eu	15	15	15	15	none	none	none	none
nobel-germany	9	9	9	9	none	9	9	none
nobel-us	8	8	8	8	8	8	8	8
norway	14	14	14	14	none	none	none	none
pdh	6	6	6	6	6	6	6	6
polska	7	7	7	7	7	7	7	7
sun	14	14	14	14	14	none	14	none
ta1	13	13	13	13	none	none	none	none

Table 34: Integer solution of F3, F4 and F3F4 (case l_h)

Low service capacity and medium link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	7	7	7	7	7	7	7	7
atlanta	8	8	8	8	8	none	8	8
dfn-bwin	**6**	6	**6**	6	8	6	7	6
di-yuan	6	6	6	6	6	6	6	6
france	13	13	13	13	none	none	none	none
geant	12	12	12	12	none	none	none	none
janos-us	14	14	14	14	none	none	none	none
newyork	9	9	9	9	none	none	none	none
nobel-eu	15	15	15	15	none	none	none	none
nobel-germany	9	9	9	9	9	none	9	9
nobel-us	8	8	8	8	8	8	8	8
norway	14	14	14	14	none	none	none	none
pdh	6	6	6	6	6	6	6	6
polska	7	7	7	7	7	7	7	7
sun	14	14	14	14	none	none	14	none
ta1	13	13	13	13	none	none	none	none

Table 35: Integer solution of F3, F4 and F3F4 (case l_m)

Low service capacity and low link capacity case								
	SP				VNFPR			
Instance	noVI	F3	F4	F3F4	noVI	F3	F4	F3F4
abilene	7	7	7	7	7	7	7	7
atlanta	**8**	none	**8**	8	none	none	none	8
dfn-bwin	6	6	6	6	6	6	6	8
di-yuan	6	6	6	6	6	6	6	6
france	none	none	none	none	none	none	none	none
geant	none	**12**	**12**	**12**	none	none	none	none
janos-us	none	none	**14**	**14**	none	none	none	none
newyork	none	none	none	**9**	none	none	non	none
nobel-eu	**15**	**15**	**15**	**15**	none	none	none	none
nobel-germany	9	9	9	9	9	9	9	9
nobel-us	**8**	**8**	8	**8**	none	none	8	none
norway	none	none	**14**	**14**	none	none	none	none
pdh	6	6	6	6	6	6	6	6
polska	7	none	none	none	7	**7**	**7**	**7**
sun	14	14	14	14	none	none	14	14
ta1	13	13	13	13	none	none	none	none

Table 36: Continuous relaxation bounds of F3, F4 and F3F4 (case l_l)