



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

Bernardetta Addis, Giuliana Carello, Francesca de Bettin, Meihui Gao

► To cite this version:

Bernardetta Addis, Giuliana Carello, Francesca de Bettin, Meihui Gao. On a Virtual Network Function Placement and Routing problem: properties and formulations. 2018. halshs-01643064v2

HAL Id: halshs-01643064

<https://shs.hal.science/halshs-01643064v2>

Preprint submitted on 23 Nov 2018

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

Bernardetta Addis¹

Giuliana Carello² and Francesca De Bettin² and Meihui Gao¹

¹Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France,
`bernardetta.addis,meihui.gao@loria.fr`

²Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano Milano, Italy, `giuliana.carello@polimi.it`

November 23, 2018

Abstract

In recent years, the increasing diffusion of applications, both on computers and mobile devices, has yielded to an increasing demand for network services. So far, the network services, such as firewalls or tunneling, were provided by expensive hardware appliances, which could not keep up with the ever increasing demand nor allow new services to be embedded at a reasonable cost. Network Functions Virtualization has been recently proposed to overcome such issues: hardware appliances are replaced with Virtual Network Functions running on generic servers. Indeed, thanks to the Network Functions Virtualization paradigm, it is possible to flexibly, dynamically and cost-effectively manage network services. A key problem in implementing the Network Functions Virtualization paradigm is the so called Virtual Network Functions chaining problem: Virtual Network Function instances must be located on some network nodes. Demands must be routed so as to guarantee that each demand passes through the functions it requires. In this work we consider a particular case of the VNF chaining problem where each demand requires a single service and must be routed on a simple path. All the demands require the same service. Links and service instances are capacitated. The goal is to minimize the number of VNF instances installed. We investigate the problem properties and we compare two formulations inspired by the two main modeling strategies proposed in the literature.

OR in telecommunications, Networks, Virtual Network Functions, ILP formulations

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, the 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 arising in implementing the NFV paradigm is the 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 of demands to VNF instances and the demand routing must satisfy quality requirements, such as delay, congestion, minimal resource utilization (CPU, energy, etc). We can schematically describe the VNF chaining problem as follows: we are given a telecommunication network, where the nodes can 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 must be served by a set of VNFs (services), which must be located on computing servers. For each demand, the VNFs must be traversed with a given (possibly partial) order or without any fixed order. Therefore a traffic demand that needs to access a VNF located in a given server must pass through the node that is directly connected to the computing server itself. Several constraints can be taken into account, such as the service order constraint, the VNF capacity constraint, and link capacity constraint, thus leading to several versions of the problem.

Different versions of the problem have been addressed in the telecommunication literature, which, however focuses mainly on the application and, to the best of our knowledge, properties have not been studied thoroughly (apart from the complexity of a particular version of the problem which has been addressed in [1]). 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 properties of a particular version of the problem and to compare two formulations derived from the main modeling strategies proposed in the literature. Further, additional inequalities to enrich such formulations are proposed. Formulations and valid inequalities are tested on a set of instances derived from the SNDLib ([2]).

We work under the (commonly used) hypothesis that the network link capacity is tighter than the capacity of the connection between computational servers and nodes. Therefore, as each server is connected directly to only one routing node, we use a compact representation where routing nodes and server nodes are collapsed, simply assigning the computational capacity of a server to the associated node. VNF instances are capacitated, as well as network connections. Further, each demand requires a single service, which is the same for all the demands, and must be routed on a simple path. The goal is to minimize the number of installed VNF instances.

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, 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 5. Section 6 concludes the paper.

2 Related work

NFV technology has received much attention from both industry and academia. A number of works on NFV have been published addressing challenging aspects spreading from the creation of NFV platforms [3],[4] to the optimization of VNF chaining with respect to, for instance, resource efficiency [5] or competitive goals [6]. We classify the literature on VNF optimization into two broad categories according to the modeling strategy. The first modeling strategy is based on the Virtual Network Embedding (VNE) paradigm [7]. In VNE a set of virtual networks is given that must be embedded into a physical one, mapping virtual nodes on physical nodes and virtual links on physical paths, while respecting capacity constraints. The second modeling strategy combines routing and location decisions.

Preliminary works on the optimization of VNF chains deployment tend to model the VNF chaining problem as a VNE problem. A virtual network is built for each demand representing the demand together with the chain of VNFs to serve it. The VNF instances location and demand routing are then defined according to the VNE mapping solution. In [8], authors introduce a VNE based MIP formulation. In [9] a VNE based ILP formulation is proposed along with a dynamic programming based heuristic to deal with large size instances. Similarly, [10] proposes a VNE based representation of the problem and focuses on the online version of the VNF chaining problem: three greedy algorithms and a tabu-search based heuristic method are proposed to deal with the VNF online mapping and scheduling. The problem of embedding VNF chains is addressed also in [11], where demands may be rejected and the subset of accepted demands must be maximized while limiting the number of service chains considered. In [5], the authors propose an ILP model based on the mapping of VNF chains on a physical network, although without naming it explicitly. Pre-

computed paths for mapping are used. The ILP model is used as a step in a heuristic procedure based on a dichotomic search on the number of located VNFs.

From the optimization perspective, however, the VNF chaining problem differs from the general VNE problem. In VNE, virtual network nodes and virtual network links need to be mapped on the physical network, while in the VNF chaining problem, the VNFs are located on physical nodes, and the service demands must be assigned to their required VNF instances and routed to pass through them. In a sense, VNF chaining can be seen as a special case of VNE where the virtual networks to be embedded reduce to linear graphs whose both extreme nodes have a fixed location. However, the VNE based representation is possible only if the order of VNFs in the chain is fixed and fully determined. As a consequence, the VNE complexity results cannot be just extended to the VNF chaining problem as they are. Indeed, [1] shows that the VNF chaining problem is easier than the VNE in some particular case and for some network topologies.

The second modeling perspective, adopted by a handful of papers in the networking literature, exploits the fact that the VNF chaining problem shares features with both facility location and network design problems. 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 [13], the specific Deep Packet Inspection (DPI) VNF placement problem, where a single type of services is asked, is targeted and modeled as an adaptation of the multicommodity flow problem model. 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 [6], authors provide a MILP formulation accounting for facility location and demand routing with a generic number of services and no fixed order in the chain, taking into account different latency regimes and traffic compression properties. Their work investigates the trade-off between a legacy traffic engineering (TE) related goal (namely maximum link utilization) and a combined TE-NFV goal (namely, the sequential optimization of the TE goal and of the VNF installing cost). An extension of the work [14] presents a model that takes 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 [6] is proposed in [15], where additional constraints are added to take into account the incompatibility of certain VNFs and thus imposing that they are located in different nodes. Furthermore, in this work, some demands can be rejected, therefore their routing and VNF assignment can be neglected. A greedy algorithm based on the decomposition of the problem into two steps (routing and location) is proposed. Flows are allocated one by one, and then the VNFs are located on the selected paths. Most of the works focus on developing heuristic methods to solve the problem within a reasonable computational time, and most

of them decompose the problem into two steps.

Although the combined facility location and network design problem has been studied in optimization literature [16], the VNF chaining problem specificities are still not explored in the optimization literature. To the best of our knowledge, we are among the first to investigate in detail the properties of the VNF chaining problem. We propose two different formulations that are representative of each modeling perspective presented in the literature (namely, VNE and routing and location based) and compare them theoretically and numerically.

3 Problem description and properties

In this work we focus on a VNF chaining problem where a single type of VNF is considered and any node can be equipped with a VNF. VNFs and links are capacitated and the number of installed VNF instances is minimized. Further, each demand must be served by an instance of the VNF and must be routed on a simple path. Let us denote the problem as *Virtual Network Function Placement and Routing with Simple Path* (VNF-PR_{SP}).

The network is represented by a graph $G(N, A)$, where N represents the set of nodes and A represents the set of capacitated arcs (or links). Let u denote the arc capacity. An instance of the VNF can be installed on each node in N and can serve a limited amount of demand q . The network demands are represented by the set D : each demand $k \in D$ is characterized by a source (origin) node o_k a destination (terminal) node t_k , a demand amount d_k . The demand must be served (pass through) an instance of the VNF (service)¹, but a demand can pass through a node without using a VNF installed on it². 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.

The problem is NP-complete even with only one type of capacity (either link or service capacity): in [17] the problem is proved to be difficult if nodes are capacitated (the same approach can be used for the service capacity case), while in [1] it is proved to be difficult if only arc capacity is considered.

3.1 General remarks

We assume that one VNF type is available, however this assumption is not so restrictive. Indeed, the problem where all the demands ask for the same sequence (same types of VNF and same order) is equivalent to the case with a single VNF type, if VNFs capacity is uniform and a node can host an instance of each VNF type.

Proposition 3.1. Let us consider two problems \mathcal{P}_1 and \mathcal{P}_2 that share the same features but the cardinality of the required chain of services: in \mathcal{P}_1 each demand

¹In the following we will use the two terms interchangeably.

²In the real application, the demand uses the routing node, but it does not use the server connected to it.

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

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

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

Now, let us consider the problem \mathcal{P}_2 . By installing an instance of each service type in T on the same nodes where instances of service a are located, and by keeping the same demand-service instance assignment as in the \mathcal{P}_1 solution, we obtain a feasible solution for \mathcal{P}_2 . In fact installing several VNF type instances on one node does not affect neither the link capacity constraints nor the routing constraints. Furthermore, as the VNF types have the same capacity, if a service $\{a\}$ instance installed in a given node i can serve all the demands assigned to it, then any service instance installed on node i can serve the same set of demands.

As any feasible solution of \mathcal{P}_1 can be extended to a feasible solution of \mathcal{P}_2 with the same number of service instances per service type, if the \mathcal{P}_1 solution location is optimal then it is optimal also for \mathcal{P}_2 . ■

We can observe that under the hypothesis that the set of required service types is exactly the same, the two problems are equivalent even if they account for different service orders ³.

3.2 Impact of biconnected components and articulation points

A lower bound on the number of VNF instances (and thus on the objective function) can be obtained if the graph contains at least one *articulation point*. Let us recall some definitions.

Definition 3.1. A graph is *biconnected* if removing any node the graph remains connected⁴.

Definition 3.2. Given a graph $G(N, A)$, a *biconnected component* is a maximal induced biconnected subgraph of G .

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

³We recall that it is always possible to schedule the services allocated on the same sever in any order, and that in the application problem this order is determined by a suitable scheduling algorithm performed at the server/cloud level

⁴i.e. there is a path between every pair of vertices

Definition 3.3. An *articulation point* of a graph G is a vertex v such that $G \setminus v$ has more connected components than G .

The following property can be stated.

Proposition 3.2. Let us consider a VNF-PR_{SP} problem defined on a graph G containing biconnected components and suppose that there exists at least one demand whose origin and destination nodes are both in the same biconnected component. Further, suppose that such biconnected component has only one articulation point. Let us refer to such biconnected components as *biconnected components with internal demands and single articulation point*.

Then, it is necessary to install a service inside each biconnected component with internal demand and single articulation point. Furthermore, if the instance is feasible, there exists an optimal solution where an instance of the service is located on each articulation point belonging to a biconnected component with internal demand and single articulation point.

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

Thanks to Property 3.2, it is possible to determine a lower bound of the number of VNFs to install combining the number and structure of the biconnected components with the source and destination of the demands. Furthermore, a partial solution can be built, installing services on articulation points. A pre-processing can be devised, which aims at

1. detecting the number of biconnected components with internal demand, thereby installing one service on each of their articulation points;
2. forbidding the assignment of a demand to the VNF which are out of the biconnected component the demand belongs to.

4 Mathematical formulation

As mentioned, in the telecommunication literature two main modeling perspectives have been proposed: the VNF Placement and Routing (VNF-PR) formulation [6] and the VNE formulation [9]. In this section we present two formulations inspired by such perspectives.

The first one is directly inspired by network design and facility location problems: a set of variables and constraints represent origin-destination routings and a set of variables and constraints represent the facility location part, connecting constraints are used to couple the two subproblems. It can be considered as the adapted version of the formulation presented in ([6]) to our VNF-PR_{SP} problem. The second one, the Split-Path (SP), is based on the decomposition of each demand path into several sub-paths, each associated with a service instance serving the demand. A similar model is presented in [13] (also here a single VNF type is considered). We can observe that the SP formulation is very close to the VNE formulations. Indeed, the VNF chaining problem can be viewed as a VNE problem where the networks to embed are linear graphs. Nevertheless, the SP formulation is leaner than the VNE based formulation, because it is tailored to the problem. Furthermore, the simple path condition is not guaranteed in the VNE formulations.

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

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

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

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

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

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

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

The objective function (1) minimizes the sum of the opened services (i.e., instances of the VNF). Constraints (2) impose that each demand is assigned to exactly one instance of the service. Inequalities (3) guarantee that if no

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

Table 1: Mathematical notation

VNF instance is installed on a node, then no demand can be assigned to it. Constraints (4) impose that each instance of the VNF can serve a maximum quantity q of demand.

The link capacity constraints are similar for the two models:

SP:

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

PR:

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

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

constraints.

Routing and assignment are modeled as follows:

SP:

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

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

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

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

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

Simple path routing⁵ is imposed by constraints (9) and (10): each demand can pass through a node at most once.

PR:

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

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

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

Each demand is routed from its source to its destination with the classical flow balance constraints (11) and it is forced to pass through a VNF

⁵Isolated cycles with no service can be part of a feasible solution. Such cycles can be removed obtaining a cycle-free equivalent feasible solution.

instance by constraints (12), that impose that a demand k can be assigned to a service located on node i only if the routing path of the demand passes through the given node. The simple path and the elimination of isolated cycles are enforced using the TSP-like labeling variables π and constraints (13): continuous variables π_i^k represents the position of node i in the routing path of demand k .

Both models can be generalized to take into account multiple service types, even hosted on different nodes. Nevertheless, SP model is able to manage the case where the order of services is fixed (as it explicitly split the path into sub-paths from source to service, service to next service, etc.) at the cost of increasing the number of routing variables and constraints, but it needs additional variables to manage the no ordered case (see A). This is due to the fact that the routing part and the location part are directly connected by the notion of sub-path, and new variables are needed to decouple them and generalize the model. Instead, PR model can be simply generalized to deal with any imposed order (full, partial, none). In fact, variables π can be used, together with additional constraints, to impose a given full or partial order of the services along the demand path (see [6]).

4.1 Relation between the two formulations

In this section we show that the SP formulation produces a continuous relaxation bound that is always not worse than the one produced by PR. The feasible region of SP can be partitioned into two subsets: in the first subset no isolated cycles exist while in the second one isolated cycles are present, but they cannot host a service (see Remark 4.1). Any solution of the second subset has an equivalent in the first one (see Proposition 4.2) and we prove that any solution of the first subset can be mapped into a solution of PR (see Theorem 4.1). Thus a solution of SP either is feasible also for PR or is equivalent to one that is. Instead, there exist solutions of PR that are not feasible for SP and for which the bound provided by SP is strictly better than the one provided by PR (see Proposition 4.3).

For any demand k , the flow in a feasible continuous solution can be divided into flow on simple paths and flow on cycles. Let us denote with P_k the set of simple paths and with C_k the set of cycles. We can distinguish between two type of cycles (see Figure 1): cycles sharing some nodes with a simple path (Figure 1a) and isolated cycles (Figure 1b).

Remark 4.1. SP forbids demands to be served, even partially, by a VNF instance installed on an isolated cycle⁶. For example, let us consider an isolated cycle of two nodes A and B such that the demand is partially served by a node

⁶Instead accepts solutions with an isolated cycle and a partial service installed on it, but routing variable values (on the cycle) cannot be greater than $\frac{|N|}{m+|N|}$ (where m is the length of the cycle) due to the isolated loop elimination constraints (13).

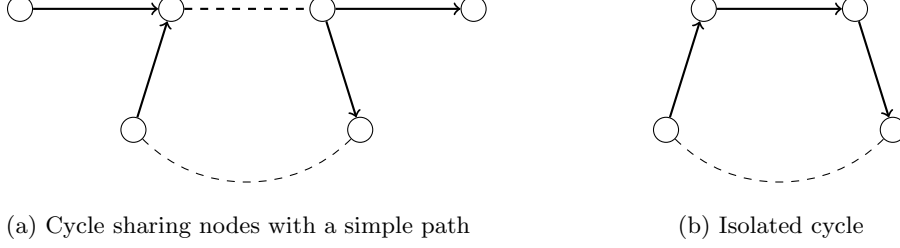


Figure 1: Examples of cycle sharing nodes and of isolated cycle

A ($z_A^k = \alpha, \alpha < 1$). Summing up flow balancing constraints (7) leads to an inconsistency $0 = -\alpha$ (analogously for subpath 2).

As a consequence, only three cases occur:

Remark 4.2. Consider a feasible solution of the continuous relaxation of SP (x^1, x^2, y, z) , a demand k and a cycle induced by such solution. Let us denote with N_c the set of nodes of the cycle c that are not shared with any simple path: $N_c = \{i \in c : \forall p \in P_k \quad i \notin p\}$. Due to flow balance constraints (7)-(8), we have only three possible cases for all nodes belonging to N_c :

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

Thus we can observe that:

Proposition 4.1. Let us consider a feasible solution (x^1, x^2, y, z) , a demand k and the path-cycle decomposition $\{P_k, C_k\}$ of its routing. Let us suppose that a cycle $c \in C_k$ exists that shares at least one node with a simple path $p \in P_k$. If a (partial) service is located on a node belonging to the cycle $c \in C_k$, but not to the path p , the routing variables inducing the path and the cycle are fractional and their value is less than or equal to $\frac{1}{2}$, due to constraints (7)-(8) and (9)-(10).

Proposition 4.2. Any solution such that no service instance is installed on a node belonging to a cycle but not to a simple path can be transformed in an equivalent solution (in term of cost, location and assignment) removing the cycles without service instances installed and the corresponding flow from the routing.

Theorem 4.1. Any solution of the continuous relaxation of SP such that no service instance is installed on a node belonging to a cycle but not to a simple path can be mapped into an equivalent solution of PR in terms of routing, location and assignment and therefore cost.

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

$$x_{ij}^k = x_{ij}^{k1} + x_{ij}^{k2} \quad (14)$$

Thanks to constraints (9) and (10) the mapping guarantees also that $x_{ij}^k \in [0, 1]$. Further, link capacity constraints of SP (5) imply the link capacity constraint of PR (6). Routing constraints of the SP formulation imply the routing constraints for the PR formulation. In fact, by summing equations (7)-(8) and using the mapping (14), we obtain constraints (11).

Now we show that SP routing constraints for the semi-path (7) imply routing-location connecting constraints of PR (12).

Let us consider the case $i \in N, i \neq o_k$:

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

reordering terms, we obtain:

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

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

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

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

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

Then, using the routing variables mapping (14), we verify constraint (12):

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

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

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

where $(i, j) \in A_k$ if $x_{ij}^{k1} + x_{ij}^{k2} > 0$ and $\exists p \in P_k : (i, j) \in p$, i.e. the arc belongs at least to one simple path. A node \hat{i} belongs to N_k if there exists an arc $((i, j) \text{ or } (j, \hat{i}))$ in A_k . For any arc $(i, j) \in G^k$ we define the following cost $c_{ij} = x_{ij}^{k1} + x_{ij}^{k2} = x_{ij}^k$.

As G^k does not contain arcs that belong only to cycles in C^k , the obtained graph is acyclic, thus we can define π_j^k as the longest path from node o_k to node j :

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

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

The nodes that belong to a cycle and to a simple path (nodes from l to m in Figure 2) have already been assigned a suitable value π . We now need to determine a value of π for the nodes on the cycle N_C that have been removed.

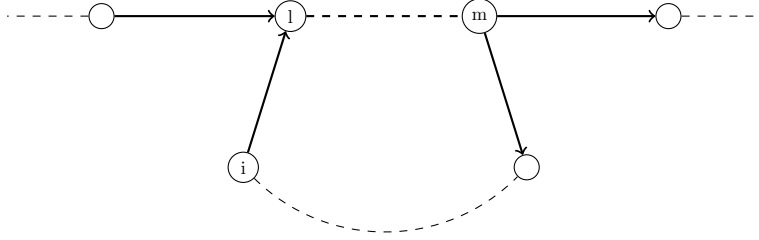


Figure 2: An example of a cycle sharing some nodes with a simple path

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

$$\pi_i^k = \frac{\pi_m^k + \pi_l^k}{2} \quad (15)$$

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

1. Arcs whose endpoints both belong to the cycle but not the path (thus both endpoints are labelled according to equation (15)).

Due to property 4.1, we can infer that for any arc belonging to the cycle but not the path, we have $x_{ij}^k \leq \frac{1}{2}$. Therefore, the term:

$$x_{ij}^k - |N| (1 - x_{ij}^k)$$

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

2. Arcs with one endpoint belonging to the cycle and the other one belonging both to the path and to the cycle, i.e. with one extreme in l or m .

Let consider the arc of the cycle entering node l : (i, l) (see Figure 2), we need to prove that:

$$\pi_l^k \geq \pi_i^k + x_{il}^k - |N|(1 - x_{il}^k), \quad (16)$$

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

$$\pi_m^k = \pi_l^k + \beta n$$

and we obtain:

$$\pi_i^k = \frac{\pi_m^k + \pi_l^k}{2} = \pi_l^k + \frac{\beta n}{2} \leq \pi_l^k + \frac{(|N| - 1)}{2}$$

Thus, denoting with γ the right-hand side of equation (16), we get:

$$\gamma \leq \pi_l^k + \frac{(|N| - 1)}{2} + x_{il}^k - |N|(1 - x_{il}^k)$$

and, rearranging the terms,:

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

and using $x_{il}^k \leq \frac{1}{2}$ (see Proposition 4.1):

$$\pi_l^k \geq \gamma$$

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

We now prove that the bound provided by PR is not better than the one of SP, by showing an instance in which the bound provided by SP is stricter than the one provided by PR. Further, we present three features that make a solution unfeasible for the SP continuous relaxation but are acceptable for the PR continuous relaxation.

Proposition 4.3. The bound provided by PR is not better than the one provided by SP.

Proof. Let us consider the symmetric graph in Figure 3a (in the following figures, each pair of the symmetric arcs is represented by the corresponding edge). The services are uncapacitated, and the link capacity is 5. There are 4 demands, whose characteristics are listed in Table 3b. The continuous relaxation provided by SP is equal to 2, while the continuous relaxation provided by PR is 1, for any value of the demands k_3 and k_4 in $(0, 5]$.

We report the details of the solution of SP formulation in Table 2 and Figure 4, while the details of the solution of PR in Figure 5 and Table 3: the assignment of service to nodes is reported in bold in the tables and in gray in

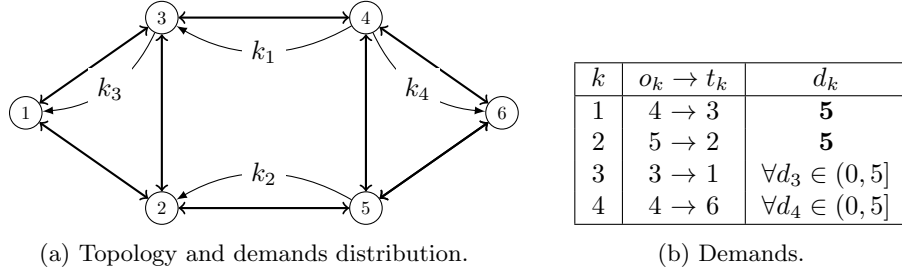


Figure 3: The SP relaxation bound is strictly better than the one of PR: a numerical example

demand	paths	f_p	node (i)	z_i^k
k_1	$p^1 : \mathbf{4} \rightarrow 5 \rightarrow 2 \rightarrow 3$	1	4	1
k_2	$p^2 : \mathbf{5} \rightarrow \mathbf{4} \rightarrow 3 \rightarrow 2$	1	4	1
k_3	$p^3 : \mathbf{3} \rightarrow 1$	1	3	1
k_4	$p^4 : \mathbf{4} \rightarrow 6$	1	4	1
Installed services				
$y_3 = 1 \quad y_4 = 1$				

Table 2: Routing and assignment/location in the continuous relaxation of SP

the figures. As for the SP formulation, one service is located on node 4 and one on node 3. Demands k_1 , k_2 and k_4 are served by the service located on node 4, while demand k_3 is served by the service located on node 3. As for the solution of PR, half service is installed on node 4 and the other half on node 5; each demand uses both services. ■

The difference between the two bounds is due to the fact that in the continuous relaxation of the PR formulation some routing solutions are feasible, while they are not for the SP formulation, because of the direct coupling of assignment and routing determined by constraints (7)-(8) present in SP. Indeed, the example above is a representative of a family of solutions that are feasible for PR but not for SP:

Remark 4.3. A solution where a source-destination path does not pass through any service node (see path p_1 for demand k_3 in 4.3) and an isolated loop hosts a (partial) service (see path p_2 for demand k_3 in 4.3) is feasible for PR but not for SP.

Such solutions may be profitable when the capacity of a cut is small and some demands cannot reach the services installed on the other side of the cut itself:

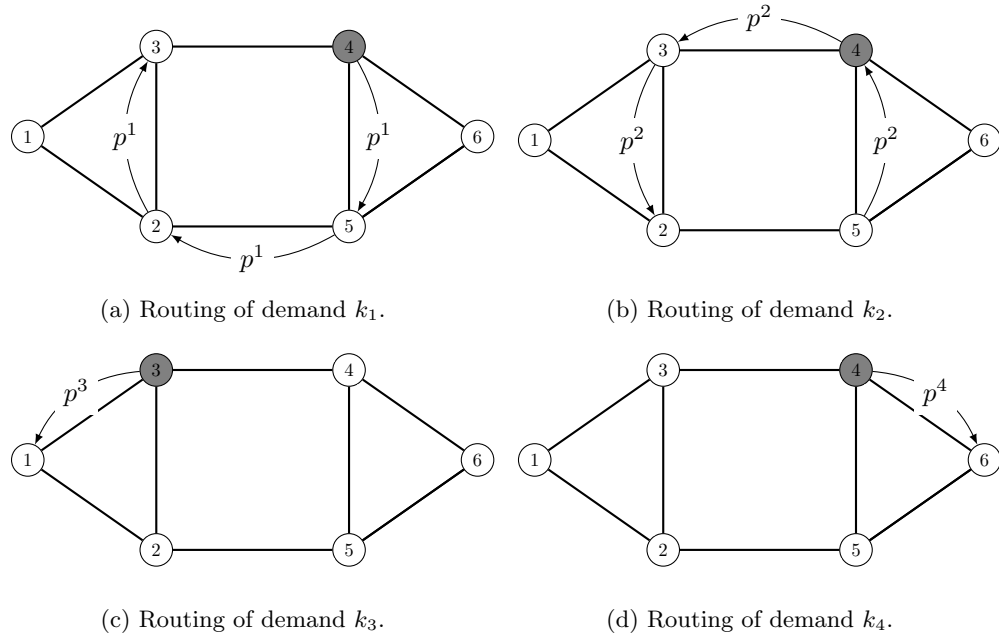


Figure 4: The routing provided by the continuous relaxation of SP

using a service on an isolated cycle is then the best option for PR. In the example described in Table 3b demands k_1 and k_2 saturate the cut $\{4, 5, 6\}, \{1, 2, 3\}$ forcing SP to install a service in each side of the cut to serve also demands k_3 and k_4 . Instead, PR does not open two services, thus providing a weaker bound. The bound provided by SP is stricter even if the amount of flow of demands k_1 and/or k_2 decreases and thus the cut is not fully saturated, as long as the residual capacity on the cut $\{4, 5, 6\}, \{1, 2, 3\}$ is not enough to allow the full demand k_3 to pass through it and back (or symmetrically demand k_4 for the reversed cut). A feasible solution is then selected by PR as described in Figure 6, where the demand is half routed ($x = 0.5$) on the path and is completely served by partially using the two services located in node i and j . Such solution is instead unfeasible for SP⁷, as proved by the following proposition.

Proposition 4.4. Let us consider a feasible solution for the continuous relaxation of SP formulation where a fraction of a demand k is routed on a path p_k . Let us consider the services located along this path and the corresponding assignment variables z_i^k , and suppose that such services are not used by demand k along

⁷The larger the fraction of demand k_3 that can pass the cut, the smaller the gap between the two continuous relaxation bounds.

demand	paths	f_p	node (i)	z_i^k
k_1	$p_1 : \mathbf{4} \rightarrow 3$	0.5	4	0.5
	$p_2 : 4 \rightarrow \mathbf{5} \rightarrow 2 \rightarrow 3$	0.5	5	0.5
k_2	$p_1 : \mathbf{5} \rightarrow 2$	0.5	5	0.5
	$p_2 : 5 \rightarrow \mathbf{4} \rightarrow 3 \rightarrow 2$	0.5	4	0.5
k_3	$p_1 : 3 \rightarrow 1$	1	-	
	$p_2 : 5 \rightarrow \mathbf{4} \rightarrow \mathbf{5}$	0.5	4	0.5
			5	0.5
k_4	$p_1 : \mathbf{4} \rightarrow 6$	1	4	0.5
	$p_2 : 5 \rightarrow 6 \rightarrow \mathbf{5}$	0.5	5	0.5
Installed services				
$y_4 = 0.5 \quad y_5 = 0.5$				

Table 3: Routing and assignment/location solution in the continuous relaxation of PR

other paths. Then:

$$x_{lm}^{k1} \geq \sum_{i \in p_k : i = \text{succ}(lm)} z_i^k \quad (17)$$

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

Proof. This property is a direct consequence of the modified flow balancing constraints (7)-(8). See Figure 7 for a schematic illustration of this property. ■

4.1.1 Impact of the biconnected components

Network topology has an impact on the quality of the bounds produced by the two formulations. In fact, the SP formulation can produce a tighter bound in presence of biconnected components, even in the uncapacitated case, due to the simple path assumption.

Let us consider the graph in Figure 8a, and a set of three demands described in Table 8b. Service and link capacities are unbounded. The graph contains two articulation points (nodes 3 and 6) and 3 biconnected components:

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

The continuous relaxation bound obtained by the SP formulation for this instance is $\frac{4}{3}$, the one obtained by the PR formulation is the trivial bound 1.

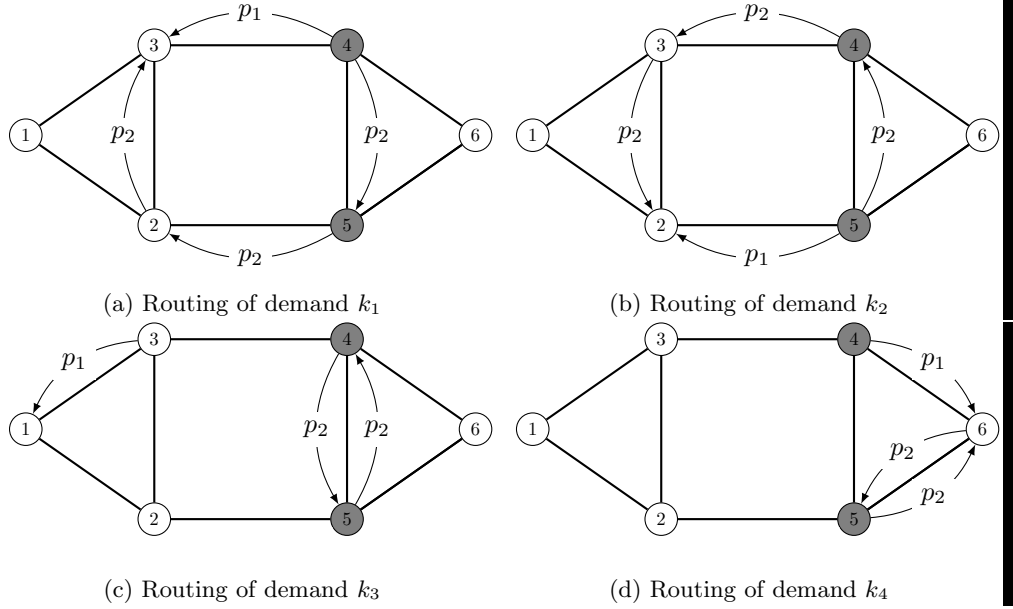


Figure 5: The routing solution of continuous relaxation of the PR

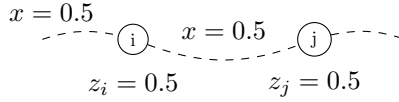


Figure 6: Example of unfeasible flow acceptable for PR

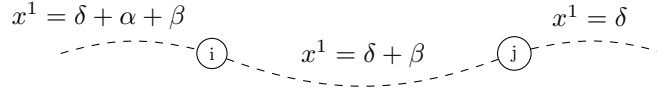
In Table 4, a solution⁸ for the SP formulation is reported. For each path, the node where the (partially) used service is located is reported in bold. Partial services are installed on nodes 2, 5 and 8. Half for nodes 2 and 8 and one third for node 5. Demand k_1 (k_3 , respectively) is served by the half service installed on node 2 belonging to its connected component BC_1 (node 8 in BC_3 respectively) and by half service installed on node 8 in connected component BC_3 (node 2 in BC_1 , respectively). Demand k_2 is split on three paths, and each of them is served by a (partial) service in a different connected component.

⁸For both formulations several equivalent solutions exist, both for location and routing. For the sake of clarity, we chose to show a “compact” solution for both formulations privileging symmetric, less fractional solutions, with short routings and a reduced number of isolated cycles.



(a) flow balance on node i

(b) flow balance on node j



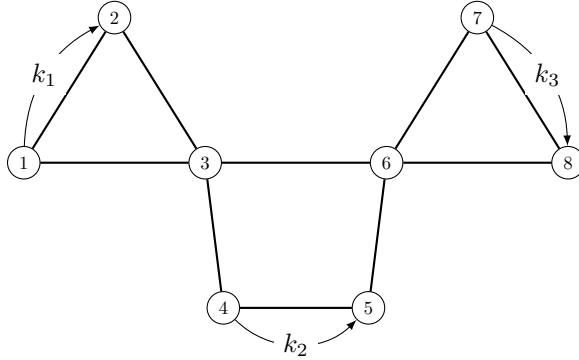
(c) resulting feasible values for x^1

Figure 7: Relation between routing and assignment variables in the SP formulation

demand	paths	fraction f_p	node (i)	z_i^k
k_1	$p_1^1 : 1 \rightarrow \mathbf{2}$	0.5	2	0.5
	$p_2^1 : 1 \rightarrow 3 \rightarrow 6 \rightarrow \mathbf{8} \rightarrow 6 \rightarrow 3 \rightarrow 2$	0.5	8	0.5
k_2	$p_1^2 : 4 \rightarrow \mathbf{5}$	1/3	5	1/3
	$p_2^2 : 4 \rightarrow 3 \rightarrow \mathbf{2} \rightarrow 3 \rightarrow 6 \rightarrow 5$	1/3	2	1/3
	$p_3^2 : 4 \rightarrow 3 \rightarrow 6 \rightarrow \mathbf{8} \rightarrow 6 \rightarrow 5$	1/3	8	1/3
k_3	$p_1^3 : 7 \rightarrow \mathbf{8}$	0.5	8	0.5
	$p_2^3 : 7 \rightarrow 6 \rightarrow 3 \rightarrow \mathbf{2} \rightarrow 3 \rightarrow 6 \rightarrow 8$	0.5	2	0.5
Installed services				
$y_2 = 0.5 \quad y_5 = 1/3 \quad y_8 = 0.5$				

Table 4: Routing and assignment/location in the continuous relaxation of SP for the biconnected components case example

In Table 5 a solution for the PR formulation is reported. Each demand is routed on the direct arc connecting its origin to its destination, thus satisfying the flow balance constraints (11). To reach the service, both demand k_2 and k_3 use two isolated cycles each in the connected component BC_1 . It is worth to notice that a single cycle would not be enough to satisfy both the coherence constraints (12) and the TSP-like cycle elimination constraints (13) (only fractional solutions can have cycles).



(a) Topology and demands distribution.

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

(b) Demands.

Figure 8: The SP relaxation bound is strictly better than the one of PR: a numerical example with biconnected components

demand	paths	fraction f_p	node (i)	z_i^k
k_1	$p^1 : 1 \rightarrow 2$	1	1	1
k_2	$p_1^2 : 1 \rightarrow 3 \rightarrow 1$	0.5	1	0.5
	$p_2^2 : 1 \rightarrow 2 \rightarrow 1$	0.5	1	0.5
	$p_3^2 : 4 \rightarrow 5$	1	-	-
k_3	$p_1^3 : 1 \rightarrow 3 \rightarrow 1$	0.5	1	0.5
	$p_2^3 : 1 \rightarrow 2 \rightarrow 1$	0.5	1	0.5
	$p_3^3 : 7 \rightarrow 8$	1	-	-
Installed services				
$y_1 = 1$				

Table 5: Routing and assignment/location in the continuous relaxation of PR for the biconnected components case example

4.2 Valid inequalities

In this section, we describe some valid inequalities.

We can rewrite the service capacity constraint (4): 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 the overall capacity of links that are incoming in the node (except for the demands served in the origin node i). A similar remark holds for the outgoing links capacity. We can calculate the maximal demand that can be served by a node as follow:

$$\bar{q}_i = \min \left\{ q, \max \left\{ \sum_{(i,j) \in A} u + \sum_{k \in D: t_k = i} d_k, \sum_{(j,i) \in A} u + \sum_{k \in D: o_k = i} d_k \right\} \right\} \quad (18)$$

Therefore, constraint (4) can be replaced by:

$$\sum_{k \in D} d_k z_i^k \leq \bar{q}_i \quad \forall i \in N \quad (19)$$

Both constraints (4) and constraints (19) can be strengthened using the service location variable y , obtaining:

$$\sum_{k \in D} d_k z_i^k \leq q y_i \quad \forall i \in N \quad (20)$$

and

$$\sum_{k \in D} d_k z_i^k \leq \bar{q}_i y_i \quad \forall i \in N \quad (21)$$

respectively.

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

$$\sum_{i \in N} y_i \geq \left\lceil \frac{\sum_{k \in D} d_k}{q} \right\rceil \quad (22)$$

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 to the 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, namely the maximum number of demands that can be served by one service. 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:

$$\sum_{k \in D} z_i^k \leq M_1 \quad \forall i \in N \quad (23)$$

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 PR:

$$\sum_{k \in D} x_{ij}^k \leq M_2 \quad \forall (i, j) \in A \quad (24)$$

For SP:

$$\sum_{k \in D} (x_{ij}^{k1} + x_{ij}^{k2}) \leq M_2 \quad \forall (i, j) \in A \quad (25)$$

5 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 ([2]). The topologies and the traffic demands (i.e., source, destination, amount of flow) are taken directly from SNDLib. Different capacity profiles were generated to analyze the impact of the VNF and link capacity. As for the links, two levels of capacity have been generated: low and high. The high capacity is such that all the demands can be routed on a single link, thus leading to uncapacitated link instances. The low capacity is computed as the minimum capacity such that a feasible routing exists, neglecting the services. As for the services, three levels of capacity are considered: low, medium and high. The high capacity is computed so as to guarantee that all the demands can be served by a single VNF (i.e., the VNF capacity is equal to the sum of the demand amount). Low VNF capacity is twice the total amount of the demands divided by the number of nodes, that is, we need to install a VNF in at least half of the nodes (for lower values many instances were not feasible). Medium capacity is the average between high and low. The obtained 96 instances are summarized in Table 6, where each row refers to a network topology. Columns two to four report the network features from SNDLib (number of nodes, number of links and number of demands). Column five gives the sum of the demand amount based on the SNDLib values. Such value is also the high capacity value both for links and VNFs. The last three columns give the values of capacity of VNFs and links: the first two report the medium and low capacity values for the VNFs and the last one reports the low capacity value for the links. In the following, we denote with h , m and l the *high*, *medium* and *low* capacity level respectively.

Models are implemented with AMPL and instances are solved with IBM ILOG CPLEX 12.7.1.0 on an Intel Xeon, CPU E5-1620 v2 (4 cores), 3.7 GHz with 32 GB of RAM. A time limit of 3600s and a tree memory limit of 3000 MB are set.

First we present the results of continuous relaxation (Section 5.1), then we report about the integer problem (Section 5.2).

5.1 Continuous Relaxation Results

We first compare the results of the two formulations on the continuous relaxation, then we evaluate the impact of the valid inequalities described in Section 4.

Summarized results on the continuous relaxations of the two formulations are reported in Table 7. The instances are grouped based on the capacity levels

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

Table 6: Instance details

and aggregated values over the 16 topologies are reported. As we proved in Section 3, the SP formulation produces a bound that is never worse than the PR formulation, therefore, (in the first group of columns) we report about the improvement obtained by the SP continuous relaxation upon the PR one. In the second column the number of instances where SP obtains a better bound than PR is given. The third and fourth columns report the average and maximum percentage of the improvement, calculated as $100\% * \frac{CR_{SP} - CR_{PR}}{CR}$, where CR_{SP} and CR denote the continuous relaxation of SP and , respectively. The average value is computed based on the instances where SP improves upon PR. The last four columns report the average and the maximum computational times (in seconds) for the two formulations. The best average computational time is reported in bold.

As expected by the theoretical results, SP formulation always provides a continuous relaxation bound that is better than or equal to the one provided by PR. When the link capacity is high SP improves upon PR just in one or two instances, but the improvement in the bound can be quite significant (up to around 150% for the *lh* case) and the average and maximum computational times are smaller (at least one third less). When link capacity is low, the number of instances where the SP formulation produces a better bound increases greatly (11-12 cases over 16) and the average and maximum improvement is more significant (up to 350%). The improvement is obtained at the price of a longer computational time, and SP cannot find a feasible solution within the

time limit (3600s) for one instance, norway *l.l*, while PR does.

cap	#	Improvement SP vs. PR		Computational times (s)			
		avg	max	SP		PR	
				avg	max	avg	max
h.h	1	100.00	100.00	0.77	4.13	5.23	46.79
h.l	12	143.00	349.38	136.67	1502.60	83.13	698.42
m.h	2	62.00	100.00	1.35	7.89	7.24	61.48
m.l	12	143.00	349.38	116.32	1240.33	96.27	513.98
l.h	2	109.7	152.66	10.41	52.35	29.42	134.35
l.l	11*	131.92	270.54	138.41	1153.76	111.91	875.50

Table 7: Comparison between the continuous relaxations

We run tests on the valid inequalities introduced in Section 4. In Table 8, we report summarized qualitative results. In the first column we report the equation corresponding to the VI. Constraints (21) are enhanced versions of the VNF capacity constraint, so they replace the VNF capacity constraint (4). In the second and the third columns the change obtained by adding the VI, in terms of bound and computational time, is qualitatively described.

VI	Bound	Runtime
Eq.(21)	almost always better	almost always increased
Eq.(22)	almost always better	almost always decreased
Eq.(23)	always unchanged	unchanged/increased
Eq.(24) ()	always unchanged	unchanged/increased
Eq.(25) (SP)	always unchanged	almost always unch./decr.

Table 8: Adding valid inequalities to continuous relaxation: qualitative behavior

In the following, we discuss in detail the most effective valid inequalities in terms of bound improvement and/or computational time reduction, i.e. the enhanced capacity equation (21) (in the following VI1) and the lower bound on the minimum number of VNFs: equation (22) (in the following VI2). We consider them singularly and together (VI1+2).

We compare the continuous relaxation bound obtained by the two formulations with the best upper bound known to evaluate its overall quality. The comparison is reported in Table 9. Results are reported in aggregated form, grouping instances based on the capacity level, for the case with noVI, and with VI1, VI2 and VI1+2. For each case, three columns are reported: the number of instances where the continuous relaxation bound is equal to the best UB and the average and maximum gap for the cases where they are not equal (calculated as $\frac{\text{best_UB} - \text{CR_LB}}{\text{best_UB}}$). No integer solution can be found for the instance norway l.l, therefore in this case we cannot calculate any gap with the best UB case. By

the way, SP cannot even find a CR value for such instance: indeed the instance has the greatest number of demands, thus suggesting that SP may experience some issues with the increasing number of demands.

SP continuous relaxation is equal to the best upper bound in all the uncapacitated instances (h_h). continuous relaxation is equal to the best upper bound in all the h_h instances but one, france, where an articulation point is present. In this case, as we proved in Section 4.1.1, is weaker than SP. We can infer that for the uncapacitated case, we cannot expect an improvement in terms of bound from the VIs, and the only possible improvement can be in terms of computational time, if any.

Adding VI1 does not increase the number of cases where the CR bound is equal to the best UB, but reduces in general the gap for the cases where they are not equal. Adding VI2 provides a larger impact both for the SP and formulation. This is not surprising, since VI2 imposes a lower bound on the number of installed VNFs, that is also a lower bound on the objective function and in the high capacity link case such bound is often tight. Combining VI1 and VI2 reduces a little more the gap for the cases where the CR bound and the best UB do not coincide.

Continuous relaxation bound for the SP formulation												
cap	# opt	noVI		# opt	VI1		# opt	VI2		#	VI1+2	
		gap			gap			gap			gap	
		avg	max		avg	max		avg	max		avg	max
h.h	16	-	-	16	-	-	16	-	-	16	-	-
h.l	2	35.35	50.00	2	33.21	48.93	2	35.35	50.00	2	33.21	48.93
m.h	1	49.2	50.00	1	10.94	16.67	16	-	-	16	-	-
m.l	0	37.79	50.00	0	20.01	42.79	7	28.9	45.03	7	28.53	42.79
l.h	0	87.57	93.33	0	8.08	16.67	16	-	-	16	-	-
l.l	0	78.96	91.67	0	8.37	16.67	15	-	-	15	-	-
Continuous relaxation bound for the formulation												
cap	# opt	noVI		# opt	VI1		# opt	VI2		#	VI1+2	
		gap			gap			gap			gap	
		avg	max		avg	max		avg	max		avg	max
h.h	15	50.00	50.00	15	50.00	50.00	15	50.00	50.00	15	50.00	50.00
h.l	2	61.82	80.00	2	49.19	60.60	2	61.82	80.00	2	49.19	60.60
m.h	0	50.00	50.00	0	10.72	16.67	16	-	-	16	-	-
m.l	0	60.00	80.00	0	27.71	59.54	7	43.33	60.00	7	42.81	59.54
l.h	0	88.90	93.33	0	8.08	16.67	16	-	-	16	-	-
l.l	0	88.67	93.33	0	8.37	16.67	15	-	-	15	-	-

Table 9: Comparison of the CR bound with respect of the best UB

Table 10 shows the average computational times for the SP and the formulations with and without the addition of VIs. The smallest computational time for each capacity case is reported in bold. The VI2 in general provides very good computational times, with few exceptions. Instead, VI1 always increases the computational times of both formulations, without improving the bound (which already coincides with the integer optimal solution).

Finally, we compare the CR of SP and when VIs are added. Figure 9 reports the percentage difference between the two continuous relaxation bounds for each VI (calculated as $100\% * \frac{CR_{SPVI_i} - CR_{VI_i}}{CR_{SPVI_i}}$). As expected VI2 reduces the difference between the two continuous relaxations only when the service capacity is medium or low, while it provides the naive bound (at least one service) when the service capacity is high. If the service capacity is medium and low, adding VIs reduces the difference between the continuous relaxations. We can observe that adding VIs reduces the computational time of while improving its bound, which is then similar to the SP one.

Computational times for the SP formulation continuous relaxation								
cap	noVI		VII		VI2		VII+2	
	avg	max	avg	max	avg	max	avg	max
h.h	0.77	4.13	0.85	4.64	4.35	26.02	7.4	78.91
h.l	136.67	1502.6	137.83	1603.9	73.04	792.81	106.95	1268.05
m.h	1.35	7.89	20.62	130.63	5.03	36.63	7.16	56.01
m.l	116.32	1240.33	140.31	1555.4	64.9	759.71	76.57	805.44
l.h	10.41	52.35	39.13	278.51	3.37	31.44	2.86	19.66
l.l	354.69	3598.88	71.92	597.26	6.55	75.62	7.82	78.92
Computational times for the formulation continuous relaxation								
cap	noVI		VII		VI2		VII+2	
	avg	max	avg	max	avg	max	avg	max
h.h	5.23	46.79	9.96	82.29	8.87	85.5	12.14	131.3
h.l	83.13	698.42	163.46	2002.48	66.01	515.51	30.68	293.53
m.h	7.24	61.48	43	285.57	7.00	51.12	18.11	148.33
m.l	96.27	513.98	143.75	2008.5	33.88	273.42	108.31	1125.63
l.h	29.42	134.35	39.46	297.89	0.79	4.08	1.19	10.46
l.l	111.91	875.5	61.92	483.14	6.54	72.25	5.6	60.22

Table 10: Computational time with and without VIs for SP and formulations

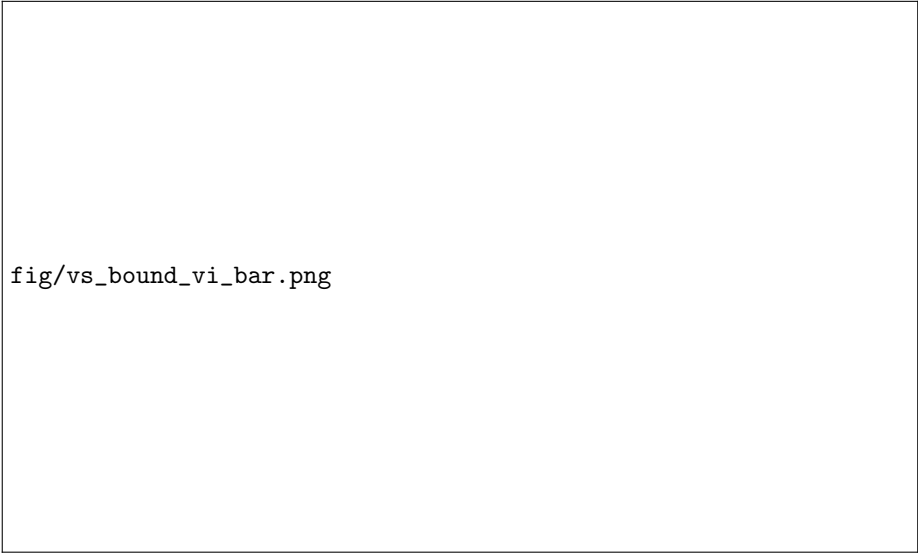
5.2 ILP results

Let us now compare the behavior of the two formulations when solving the integer problem. In Figures 10a - 10c (and respectively Figures 10b - 10d), the number of optimal and integer feasible solutions found by the SP (respectively) formulation are reported.

SP performs better than , in fact SP finds the optimal solution when the link capacity is high. When it is low, SP cannot find the optimal solution for five instances out of 16 when no VIs are added, but this number improves adding VIs. The number of feasible solutions is similar to the number of optimal solutions, indeed if SP can find a feasible solution it can almost always prove its optimality. When SP cannot prove optimality, the gap is still reasonable (around 27% for janos_us instance with h.l and m.l capacities). Adding VIs increases the number of feasible solutions that can be found when the link capacity is low.

can find the optimal solution in more than half of the instances for the h.h case and the l.h case. For all the other capacity cases, it can solve to optimality less than half of the instances. The number of feasible solutions found is slightly higher than the number of optimal solutions.

Adding VIs not always improves the performance, for example in the h.l



fig/vs_bound_vi_bar.png

Figure 9: Average percentage difference between SP model and model

case the number of optimal (and feasible integer) solutions is reduced. The low link capacity makes the instances more challenging for both formulations.

In Table 11, computational times (in seconds) are reported. In the first block, the average times when the optimality is proved are reported, in the second block the average computational time including the cases where the time limit (3600s) is reached are summarized. The SP formulation in general performs better than the formulation, and adding both valid inequalities seems to produce the best results, reducing in almost all cases the total computational time.

In Table 12 we report about the UB and the LB of each combination of formulation and VIs w.r.t. the best known. Each group of columns refers to a VI case. The following data are reported: in the first column, the number of cases where the bound is equal to the best known, in the second column, the number of cases where the bound cannot be found and in the third column, the average gap with respect to the best known bound (when the gap is not null). The SP formulation with the VI1+2 shows the best performance among all possible variants, producing the best bounds (UB and LB) in the largest number of instances.

5.3 Articulation point based preprocessing

We tested the articulation point preprocessing on 3 network topologies of SNDLib which contain at least one articulation point: france, ta2 and zib54. Instance

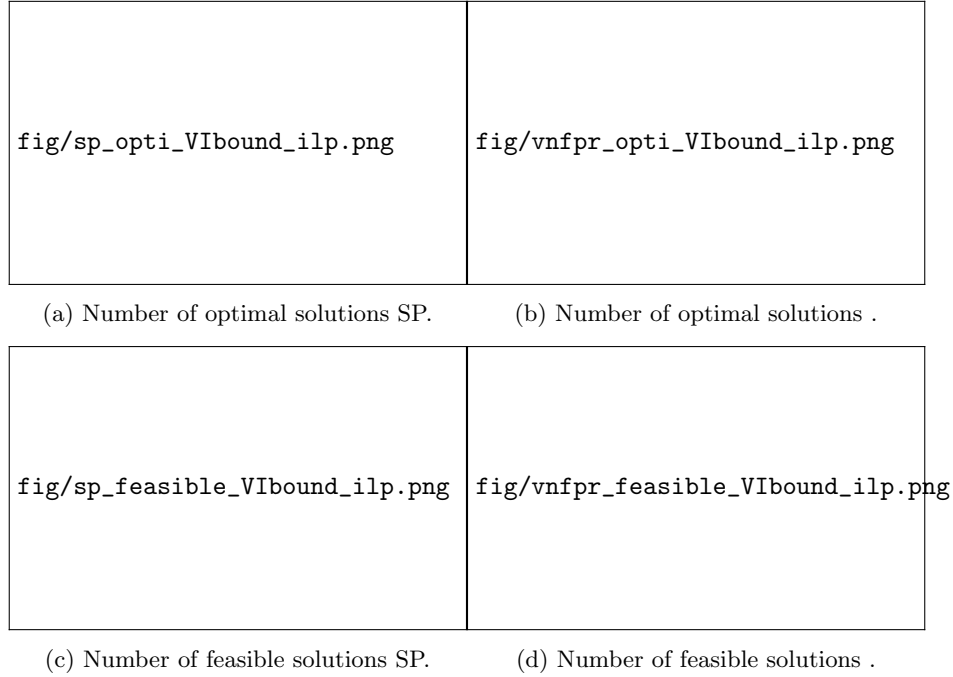


Figure 10: Number of optimal and integer feasible solutions found

ta2 (zib54, respectively) contain 54 nodes, 81 edges and 1501 demands (65, 108 and 1069, respectively). The preprocessing is quite fast (few seconds) and its addition can improve the continuous relaxation, especially for the formulation where otherwise often the naive relaxation - one service - is found. As for the integer problem, its addition reduces almost always the computational time and allows solving to optimality, despite their big size, instances for which no feasible solution can be found without preprocessing (ta2 and zib54 for the l.h capacity case and the zib54 for the l.l capacity case). Nevertheless, in some capacity cases (all the h.l and m.l cases) no feasible solution can be found even using the preprocessing. However, as it needs negligible computational time and is beneficial for some instances, although not for all, the articulation point based preprocessing is worth performing.

6 Conclusions

This paper addresses a Virtual Network Function Placement and Routing problem, where each demand can be routed only through a simple path. The same service is required by all the demands and the number of needed service instances is minimized.

We first prove some problem properties:

Average computational time when optimal integer solution is found								
cap	SP				noVI	VI1	VI2	VI1+2
	noVI	VI1	VI2	VI1+2				
h_h	2.46	2.54	6.46	6.79	256.85	254.23	442.72	453.55
h_l	586.58	615.06	256.13	533.93	481.80	2386.38	1156.05	383.09
m_h	351.03	63.77	53.10	10.53	481.69	21.14	23.04	6.27
m_l	766.31	408.06	270.33	233.91	219.88	458.82	161.16	1548.00
l_h	120.04	53.77	7.15	5.16	83.08	48.38	13.02	25.30
l_l	429.55	647.06	17.74	10.22	358.12	616.65	73.58	117.17
Total average computational time considering also timelimit (3600s)								
cap	SP				noVI	VI1	VI2	VI1+2
	noVI	VI1	VI2	VI1+2				
h_h	2.46	2.54	6.46	6.79	1679.58	1681.81	1919.80	1928.85
h_l	1528.01	1427.23	1026.41	1294.09	2936.61	3198.52	3087.09	3034.33
m_h	351.03	63.77	53.10	10.53	2301.89	2212.81	1812.10	1670.79
m_l	1605.13	1396.10	1243.67	1228.73	3388.13	2705.17	2754.96	2942.03
l_h	120.04	53.77	7.15	5.16	1474.12	1167.35	1581.80	1588.63
l_l	1420.11	1569.64	445.12	369.82	1864.82	1704.02	1665.86	1703.43

Table 11: Average computational times (in seconds) for the ILP

- the single service case is equivalent to the multiple service one, when the sequence of services is the same for all the demands and there is no limit on the number of service instances installed on a node;
- an instance of the service must be installed on the articulation points that belong to biconnected components with internal demands.

Then two formulations are compared which are inspired by the two main modeling strategies proposed in the literature. The first modeling strategy (SP) is based on splitting each demand path into two sub-paths, one connecting the source with the service, the second one connecting the service with the destination. The second one () uses arc flow variables and forbids cycles exploiting node labels. We discuss the relation between the formulations and prove that SP always provides a better bound than . Valid inequalities are also proposed and their impact evaluated.

As expected from the theoretical results, computational tests show that stand alone SP improves upon , which in turn benefits more from the VIs and seems to suffer less from the increasing number of demands.

We are currently extending and SP formulations to the multi-service and multi-sequence case.

SP formulations comparison vs best known UB												
cap	noVI			VI1			VI2			VI1+2		
	best	none	gap	best	none	gap	best	none	gap	best	none	gap
h.h	16	0	-	16	0	-	16	0	-	16	0	-
h.l	11	5	-	13	3	-	13	3	-	13	3	-
m.h	16	0	-	16	0	-	16	0	-	16	0	-
m.l	11	5	-	13	3	-	12	4	-	13	3	-
l.h	16	0	-	16	0	-	16	0	-	16	0	-
l.l	11	5	-	11	5	-	14	2	-	15	1	-
formulations comparison vs best known UB												
cap	noVI			VI1			VI2			VI1+2		
	best	none	gap	best	none	gap	best	none	gap	best	none	gap
h.h	9	0	7.00	9	1	4.5	8	2	5.50	8	2	5.00
h.l	4	10	4.50	3	11	3.00	3	12	3.00	2	11	2.00
m.h	7	7	2.50	6	7	2.33	7	7	5.00	8	7	9.00
m.l	2	11	1.33	5	10	1.00	4	11	4.00	4	11	1.00
l.h	9	6	2.00	9	7	-	9	6	1.00	9	6	4.00
l.l	7	9	-	6	10	-	9	7	-	8	8	-
SP formulations comparison vs best known LB												
cap	noVI			VI1			VI2			VI1+2		
	best	none	gap	best	none	gap	best	none	gap	best	none	gap
h.h	16	0	-	16	0	-	16	0	-	16	0	-
h.l	14	0	0.98	14	0	0.54	14	0	0.47	15	0	1.07
m.h	16	0	-	16	0	-	16	0	-	16	0	-
m.l	13	0	0.62	13	0	0.11	15	0	0.33	15	0	0.01
l.h	16	0	-	16	0	-	16	0	-	16	0	-
l.l	12	1	0.80	12	0	0.70	16	1	-	16	0	-
formulations comparison vs best known LB												
cap	noVI			VI1			VI2			VI1+2		
	best	none	gap	best	none	gap	best	none	gap	best	none	gap
h.h	15	0	1.00	15	0	1.00	15	0	1.00	15	0	1.00
h.l	6	0	2.22	6	0	1.40	5	0	2.11	4	0	1.26
m.h	8	0	0.37	9	0	0.15	16	0	-	16	0	-
m.l	2	0	1.31	6	0	1.12	7	0	1.57	7	0	1.29
l.h	11	0	0.77	12	0	0.71	16	0	-	16	0	-
l.l	9	0	0.70	8	0	0.60	16	0	-	16	0	-

Table 12: Comparison of UB and LB found for the different formulations (with and without VIs) with respect to the best known UB and LB

SP and models apply different modeling strategies to guarantee the simple path routing of each demand. Indeed, other modeling approaches have been proposed in the literature to forbid non simple paths, although for different problems: for instance in [18] a flow based formulation is proposed for the shortest path problem with negative cost. As future work, the possibility of applying alternative modeling techniques to the VNF-PR_{SP} is worth analyzing.

References

References

- [1] B. Addis, G. Carello, M. Gao, On the complexity of a virtual network function placement and routing problem, in: ENDM - Alio Euro 2018 special issue, 2018.
- [2] S. Orlowski, R. Wessäly, M. Pióro, A. Tomaszewski, SNDlib 1.0 – Survivable Network Design library, *Networks* 55 (3) (2010) 276–286. doi: 10.1002/net.v55:3. URL <http://dx.doi.org/10.1002/net.v55:3>
- [3] J. Soares, M. Dias, J. Carapinha, B. Parreira, S. Sargento, Cloud4NFV: A platform for virtual network functions, in: 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), 2014, pp. 288–293. doi: 10.1109/CloudNet.2014.6969010.
- [4] J. Hwang, K. K. Ramakrishnan, T. Wood, Netvm: High performance and flexible networking using virtualization on commodity platforms, *IEEE Transactions on Network and Service Management* 12 (1) (2015) 34–47. doi:10.1109/TNSM.2015.2401568.
- [5] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, L. P. Gaspar, 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), 2015, pp. 98–106. doi:10.1109/INM.2015.7140281.
- [6] B. Addis, D. Belabed, M. Bouet, S. Secci, Virtual network functions placement and routing optimization, in: 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), 2015, pp. 171–177. doi: 10.1109/CloudNet.2015.7335301.
- [7] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, X. Hesselbach, Virtual network embedding: A survey, *IEEE Communications Surveys Tutorials* 15 (4) (2013) 1888–1906. doi:10.1109/SURV.2013.013013.00155.
- [8] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, D. Soldani, A novel approach to virtual networks embedding for SDN management and orchestration, in: 2014 IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–7. doi: 10.1109/NOMS.2014.6838244.
- [9] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, On orchestrating virtual network functions, in: 2015 11th International Conference on Network and Service Management (CNSM), 2015, pp. 50–56. doi: 10.1109/CNSM.2015.7367338.

- [10] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, 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), 2015, pp. 1–9. doi:10.1109/NETSOFT.2015.7116120.
- [11] T. Lukovszki, S. 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, Springer-Verlag New York, Inc., New York, NY, USA, 2015, pp. 104–118. doi:10.1007/978-3-319-25258-2_8. URL http://dx.doi.org/10.1007/978-3-319-25258-2_8
- [12] S. Mehraghdam, M. Keller, H. Karl, Specifying and placing chains of virtual network functions, in: 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), 2014, pp. 7–13. doi:10.1109/CloudNet.2014.6968961.
- [13] M. Bouet, J. Leguay, V. Conan, Cost-based placement of vDPI functions in NFV infrastructures, in: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–9. doi:10.1109/NETSOFT.2015.7116121.
- [14] M. Gao, B. Addis, M. Bouet, S. Secci, Optimal orchestration of virtual network functions, CoRR abs/1706.04762. arXiv:1706.04762. URL <http://arxiv.org/abs/1706.04762>
- [15] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, E. Gourdin, Virtual function placement for service chaining with partial orders and anti-affinity rules, Networks 71 (2) 97–106. doi:10.1002/net.21768. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21768>
- [16] I. Contreras, E. Fernández, General network design: A unified view of combined location and network design problems, European Journal of Operational Research 219 (3) (2012) 680 – 697, feature Clusters. doi:http://dx.doi.org/10.1016/j.ejor.2011.11.009. URL <http://www.sciencedirect.com/science/article/pii/S0377221711009969>
- [17] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, L. P. Gaspar, A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining, Computer Communications 102 (2017) 67 – 77. doi:https://doi.org/10.1016/j.comcom.2016.11.002. URL <http://www.sciencedirect.com/science/article/pii/S0140366416305485>
- [18] M. Ibrahim, N. Maculan, M. Minoux, A strong flowbased formulation for the shortest path problem in digraphs with negative cycles,

International Transactions in Operational Research 16 (3) (2009) 361–369. arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-3995.2008.00681.x>, doi:10.1111/j.1475-3995.2008.00681.x.
URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2008.00681.x>

A Extended SP model for multiple services

We now consider to have multiple services and that each demand can ask for a different number of services among the available ones. We need to introduce the following notation:

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

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

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

Decision variables must be extended accordingly:

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

The resulting extended models is:

$$\begin{aligned}
& \min \sum_{i \in N} \sum_{f \in F} y_{if} \\
& \sum_{i \in N} z_{if}^k = 1 \quad \forall k \in D, f \in F : VNF_f^k = 1 \\
& z_{if}^k \leq y_{if} \quad \forall k \in D, i \in N, f \in F \\
& \sum_{k \in D} \sum_{s \in 1..n_k+1} d_k x_{ij}^{ks} \leq u \quad \forall (i, j) \in A \\
& \sum_{k \in D : VNF_f^k = 1} d_k z_{if}^k \leq q_f \quad \forall i \in N, f \in F \\
& \sum_{j : (i, j) \in A} x_{ij}^{k,s} - \sum_{j : (j, i) \in A} x_{ji}^{k,s} = z_{i, f^k(s-1)}^k - z_{i, f^k(s)}^k \quad \forall k \in D, i \in N, s \in 2..n_k \\
& \sum_{j : (i, j) \in A} x_{ij}^{k,1} - \sum_{j : (j, i) \in A} x_{ji}^{k,1} = \begin{cases} 1 - z_{i, f^k(1)}^k & \text{if } i = o_k \\ -z_{i, f^k(1)}^k & \text{otherwise} \end{cases} \quad \forall k \in D, i \in N \\
& \sum_{j : (i, j) \in A} x_{ij}^{k, n_k+1} - \sum_{j : (j, i) \in A} x_{ji}^{k, n_k+1} = \begin{cases} z_{i, f^k(n_k)}^k - 1 & \text{if } i = t_k \\ z_{i, f^k(n_k)}^k & \text{otherwise} \end{cases} \quad \forall k \in D, i \in N \\
& \sum_{s \in 1..n_k+1} \sum_{l : (i, j) \in A} x_{ji}^{ks} \leq 1 \quad \forall k \in D, i \in N \\
& \sum_{s \in 1..n_k+1} \sum_{j : (i, j) \in A} x_{ij}^{ks} \leq 1 \quad \forall k \in D, i \in N
\end{aligned}$$

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

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

Only the flow balancing constraints are affected by this change:

$$\begin{aligned}
& \sum_{j : (i, j) \in A} x_{ij}^{k,s} - \sum_{j : (j, i) \in A} x_{ji}^{k,s} = w_i^{k, s-1} - w_i^{k, s} \quad \forall k \in D, i \in N, s \in 2..n_k \\
& \sum_{j : (i, j) \in A} x_{ij}^{k,1} - \sum_{j : (j, i) \in A} x_{ji}^{k,1} = \begin{cases} 1 - w_i^{k,1} & \text{if } i = o_k \\ -w_i^{k,1} & \text{otherwise} \end{cases} \quad \forall k \in D, i \in N \\
& \sum_{j : (i, j) \in A} x_{ij}^{k, n_k+1} - \sum_{j : (j, i) \in A} x_{ji}^{k, n_k+1} = \begin{cases} w_i^{k, n_k} - 1 & \text{if } i = t_k \\ w_i^{k, n_k} & \text{otherwise} \end{cases} \quad \forall k \in D, i \in N
\end{aligned}$$

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

$$\sum_{f \in F: VNF_f^k=1} z_{if}^k = \sum_{s \in 1..n_k+1} w_i^{k,s} \quad \forall k \in D, i \in N$$