Offline Re-Optimization of VNF Placement Decisions for Existing Network Slices -A Game-Theoretic Algorithm

Ali El Amine, Olivier Brun

Offline Re-Optimization of VNF Placement Decisions for Existing Network Slices - A Game-Theoretic Algorithm

Ali El Amine and Olivier Brun
LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France,
{aelamine,brun}@laas.fr

December 3, 2021

Abstract

Network Function Virtualization (NFV) simplifies the deployment of network services by leveraging virtualization technologies to make the management of network functions more flexible and cost efficient. The deployment of these services requires the allocation of Virtual Network Function - Forwarding Graph (VNF-FG), which implies placing and chaining VNFs according to the requests of VNF-FGs. In this paper, we consider the offline allocation of VNF-FG problem to improve resource utilization and reduce total costs. We focus on how VNF-FG demands are routed so as to optimize resource utilization without adding capacity to the infrastructure. Given a non-linear cost function associated to each network resource, we formulate the problem as a non-linear single-path routing problem in an extended graph. Then, we propose to adapt a single-path routing heuristic algorithm inspired from game theory to solve it. We show that this algorithm converges and establishes its approximation ratio in a number of cases. Experimental results obtained for different network topologies and different cost functions show that this algorithm provides very good quality solutions with a substantially lower computing times compared to the optimal solution.

Keywords— Network Function Virtualization, Virtual Network Function - Forwarding Graph, Slice as a Service, Offline Allocation, Game Theory, Optimization

1 Introduction

Network Function Virtualization (NFV) opens up the possibility to replace hardwired middleboxes implementing specific network functions (e.g., firewalls or Web proxies) by Virtual Network Functions (VNFs) running in datacentres and operated as cloud services. In addition to reducing the cost and complexity of the network, this approach enables to deploy and compose VNFs so as to create *network slices* on demand. Network slices are isolated logical networks coexisting simultaneously on the same Physical Substrate Network (PSN), each one being tailored to the needs of a specific application. The VNFs composing the network slice as well as the virtual links established between them are described by a VNF Forwarding Graph (VNF-FG). A VNF-FG, according to ETSI, is a directed graph of an ordered set of VNFs that compose an end-to-end service [1].

The logical service graph described by a VNF-FG is decoupled from the network and has to be mapped to existing PSN capabilities [2]. This implies not only the fulfillment of slice's requirements in terms of Quality of Service (QoS), but also considering the constraints of the underlying infrastructure. Two types of decisions have to be made: (a) where to run the VNFs, and (b) how to interconnect them in the physical network, taking into account specific VNF ordering requirements. They are usually made so as to optimize a certain performance objective, which may be related for instance to the resource utilization of the PSN, while satisfying a number of technical constraints (e.g., some VNFs should be placed in the vicinity of end users). This problem, which is known as the *VNF Forwarding Graph Embedding* (VNF-FGE), is an extension of the Virtual Network Embedding problem [3, 4] which is known to be NP-hard [5]. The VNF-FGE problem has been studied in the online context, where requests to create network slices are submitted one after the other [6–10], and in the offline context where multiple VNF-FG have to be embedded in the physical network at the same time [11–14].

In this work, we consider a physical infrastructure in which transmission and computing resources are available. These resources have been used to set up a number of logical networks. As the requests for new

logical networks arrive one after the other in time, without knowing neither the future requests nor the logical networks that will be removed in the future, the individual placement and routing decisions made are necessarily sub-optimal in hindsight, which may result in poor resource utilisation. It may therefore be necessary to re-optimise the mapping of existing logical networks onto the physical infrastructure. It should allow for a better load distribution on existing resources, and thus increase the capacity of the physical infrastructure to accommodate new logical networks without having to add additional capacity.

Despite the previous efforts on the VNF-FG embedding problem, little considered the re-optimization of resources utilization on the physical infrastructure with non-linear cost functions. In [11], the authors aimed at minimizing the number of VNF instances mapped on the infrastructure to improve resource utilization. However, the work ignored the routing cost from connecting the VNFs of the VNF-FG request. Inspired by the previous work to share the same VNF by multiple VNF-FGs, the work in [12] and [13] studied the offline VNF-FGE problem to reduce both deployment and forwarding traffic costs. In [12], the authors proposed a graph-based heuristic inspired from the Viterbi algorithm to reduce the deployment cost of requested VNF-FG by reducing the number of PoPs hosting the VNF instances. However, this solution results in an unbalanced infrastructure resource utilization. In [13], the authors focused at minimizing the cost of routing and deploying the requested VNFs, while guaranteeing a certain level of reliability. The work however, does not consider an efficient use of resources for future requests. The closest to our work is [14] that focuses on minimizing both resource utilization and total placement costs. Despite the similarities with our work, the model considered is different from ours. First, the authors considered an infrastructure with only PoP nodes that are used to host VNFs. Hence, their work do not consider the routing problem. Second, the authors considered the possibility to deploy only one type of VNF in a PoP. Moreover, none of the previous works considered non-linear cost functions which are essential if one wishes to express that the cost per unit capacity of a resource grows with its utilization rate.

In the setting that we consider, we are given a physical network whose links have known transmission capacities and in which some nodes provide known computing resources. A non-linear cost function is associated to each network resource and allows to measure its congestion level. A set of traffic demands flow through the network, each traffic demand being characterised by its source, destination, volume and the sequence of VNFs it must flow through. The volume of a traffic demand may change after each VNF, due to, for example, encryption or data aggregation operations. An initial solution specifies, for each traffic demand, on which compute nodes the VNFs are executed and the path taken by the demand in the network between these VNFs. We show that the VNF placement and chaining problem can be formulated as a non-linear single-path routing problem in an extended graph. To solve this problem, we adapt a single-path routing algorithm inspired from Game Theory which was proposed in [15]. We show that this algorithm converges and establishes its approximation ratio in a number of cases. Experimental results obtained for different network topologies and different cost functions show that this algorithm provides very good quality solutions with a rather modest computation time.

The paper is organized as follows. We describe the problem addressed in this paper and introduce our main notations in Section 2. In Section 2.1, we introduce the concept of expanded network. The problem is then formulated as a non-linear single path routing problem in the expanded network in Section 2.2. We present the proposed algorithm in Section 3. Section 4 is devoted to the performance evaluation of this algorithm using diverse network topologies and cost functions. Finally, some conclusions are drawn in Section 5.

2 Problem Statement

We are given a physical network represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of links. A subset $\mathcal{D} \subseteq \mathcal{V}$ of the nodes have the required compute and storage resources to host VNF. These nodes will be called the *function nodes* in the following. They are used to execute one or more VNFs out of a set $\mathcal{F}$ of K VNFs, where function $f \in \mathcal{F}$ is available at function nodes $v \in \mathcal{D}(f) \subseteq \mathcal{D}$. A function node corresponds to a logical entity representing reserved computing and storage capacity in a data centre. Multiple VNFs may be deployed at the same function node, and a VNF may be replicated at different function nodes. The transmission links of the network and the function nodes are the critical resources in our problem and in the following we shall denote by $\mathcal{R} = \mathcal{E} \cup \mathcal{D}$ the set of these resources.

We are also given a set $\mathcal{W}$ of traffic demands. Each traffic demand $w \in \mathcal{W}$ is characterized by its source node s_w, its destination node t_w and the sequence $\mathcal{F}_w = \left(f_1^w, f_2^w, \ldots, f_{K_w}^w\right)$ of network functions it must flow through in that prescribed order. The sequence is called the *Service Chain* associated to the traffic demand. The processing path of demand w may therefore be decomposed in different stages, where stage 0 corresponds to the transmission of the original traffic volume from the source to VNF f_1^w (excluded), stage 1 represents the processing at VNF f_1^w followed by the transmission to VNF f_2^w, etc. The volume of a traffic demand may change at each stage due to, e.g., packet encryption/decryption or to data aggregation

Table 1: Notations.

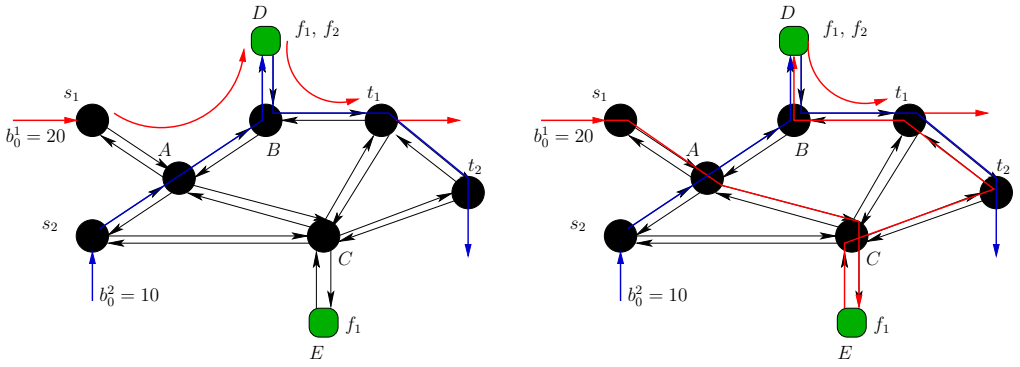
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Physical network, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of links.
$\mathcal{D}$	Set of function nodes.
$\mathcal{R}$	Set of network resources (transmission links and function nodes).
$\mathcal{F}$ (resp. $\mathcal{F}_w$)	Set of K network functions (resp. service chain for demand w).
$\mathcal{D}(f)$	Set of function nodes where VNF f can be executed.
$\mathcal{W}$	Set of traffic demands.
s_w (resp. t_w)	Source (resp. destination) node of demand w.
K_w	Number of VNFs in the service chain $\mathcal{F}_w$.
f_k^w	k^{th} VNF in the service chain $\mathcal{F}_w$.
$b_{k,r}^w$	Capacity required by demand w from resource r at stage k.

operations. Moreover, the processing capacity required per unit of traffic may be different for one VNF to another. We therefore define $b_{k,r}^w$ as the capacity required from resource r for processing demand w at stage k. If r is a transmission link $e \in \mathcal{E}$, then this capacity just represents the volume of traffic send from VNF f_k^w to VNF f_{k+1}^w for $0 < k < K(w)$, and the volume of traffic from s_w (resp. VNF $f_{K(w)}^w$) to VNF f_1^w (resp. t_w) for $k = 0$ (resp. $= K(w)$). This volume of traffic should be the same for all links e and is expressed in bit/s. Similarly, if r is a function node $v \in \mathcal{D}(f_k^w)$, the capacity $b_{k,r}^w$ represents the (fractional) number of cores required for processing the traffic of demand w by VNF f_k^w. Our notations are summarized in Table 1.

Figure 1 provides a simple example of the setting considered in the paper, in which there are two traffic demands. The service chain for the traffic demand $s_1 - t_1$ is $\mathcal{F}_1 = (f_1, f_2)$, whereas the service chain for the traffic demand $s_2 - t_2$ is $\mathcal{F}_2 = (f_1)$. The volume of traffic demand $s_1 - t_1$ is $b_0^1 = 20$ units of traffic and the volume of traffic demand $s_2 - t_2$ is 10 units. To simplify, we assume that the volumes do not change along the paths from the sources to the destinations. We also assume that VNF f_1 requires 1.5 cores per unit of traffic, whereas VNF f_2 requires 1.0 core per unit of traffic. The capacity of all links are taken to be $c_e = 60$ units of traffic, and the processing capacity of the function nodes are $c_D = 70$ and $c_E = 40$. The figure illustrates two possible deployment scenarios. In the first scenario (see Fig. 1a), all instances of all VNFs are deployed at node D, whereas in the second scenario (see Fig. 1b) the instances of VNF 1 required for processing the traffic demand $s_1 - t_1$ are executed at node E. We analyze below each scenario, assuming that the cost function for resource r is $\frac{y_r}{c_r - y_r}$:

- **(a) Scenario 1**: In this scenario, the utilization rates of the network links are as follows: $\frac{1}{3}$ for one link, $\frac{1}{6}$ for two links and $\frac{1}{2}$ for four links. This yields a total link cost of $1 \times \frac{1/3}{1-1/3} + 2 \times \frac{1/6}{1-1/6} + 4 \times \frac{1/2}{1-1/2} = 4.9$. However, the total capacity required at node D is 65 cores ($(20 + 10) \times 1.5$ for VNF f_1, and 20×1 for VNF f_2), yielding a total cost for function nodes equals to $\frac{65}{70-65} = 13$. Hence the total network cost for this deployment scenario is 17.9.
- **(a) Scenario 2**: In this scenario, the utilization rates of the network links are as follows: $\frac{1}{3}$ for seven links, $\frac{1}{6}$ for three links and $\frac{1}{2}$ for three other links. The total link cost is therefore $7 \times \frac{1/3}{1-1/3} + 3 \times \frac{1/6}{1-1/6} + 3 \times \frac{1/2}{1-1/2} = 7.1$. The computing capacity required at node D is 35 cores: 10×1.5 for VNF f_1, and 20×1 for VNF f_2. The computing capacity required at node E is $20 \times 1.5 = 30$ cores. The total computing cost of function nodes is hence $\frac{35}{70-35} + \frac{30}{40-30} = 4$. We conclude that the total network cost in this scenario is 11.1, which shows that it achieves a better resource utilization.

The problem at hands amounts to choosing a single path through the network for each traffic demand such that the function nodes are visited in the prescribed order and so that a certain network cost function is minimized. We shall shortly discuss the network cost function to be optimized, but we would first like to emphasize that a first difficulty is related to the modelling of precedence constraints on the order in which the function nodes must be visited by each traffic request. The key observation is that the overall path from s_w to t_w for traffic demand w can be viewed as a collection of segments, the first segment connecting s_w to one of the function nodes hosting f_1^w through some intermediate nodes, the second segment connecting the end node of the first segment with one of the function nodes hosting f_2^w, etc, until the destination node t_w is reached. These segments are connected by decision points where we choose which nodes will compute the functions in $\mathcal{F}_w$. As observed in [16], it is possible to construct an expanded network to model both the segment construction and the function selection at the nodes. We shall discuss the construction of this expanded network in Section 2.1. In Section 2.2, we formulate the single-path routing problem considered in this paper using the concept of expanded network.



(a) Deployment scenario 1 in which all VNFs are executed at node D.

(b) Deployment scenario 2 in which instances of VNF 1 are executed at node E for the traffic demand $s_1 - t_1$.

Figure 1: A simple example network , in which we assume that VNF f_1 may be executed at nodes D and E, whereas VNF f_2 is only available at node D.

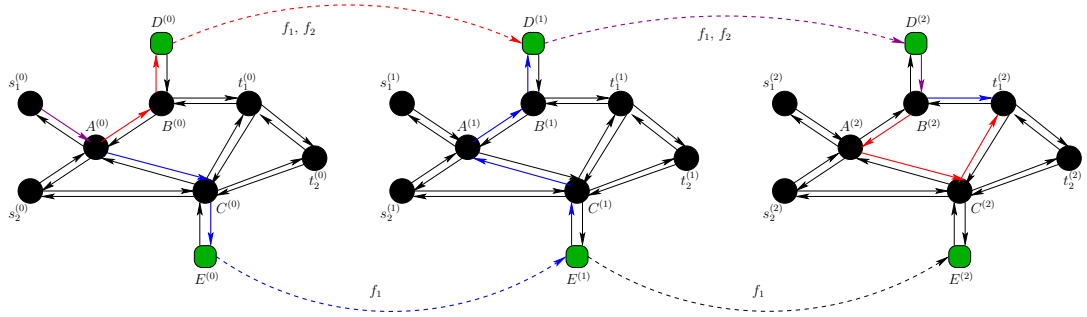


Figure 2: Expanded network for the example of Figure 1. The artificial edges between consecutive layers are shown with dotted lines. Two feasible paths for the demand from node s_1 to node t_1 are shown, with blue-coloured edges for the first one and with red-coloured edges for the other one (common edges are purple-coloured).

2.1 Expanded Network

The expanded network $\mathcal{G}^*$ is made up of K layers. Each layer $\mathcal{G}^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)})$, $k = 0, \ldots, K$, is a copy of of the original network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We shall use the notation $v^{(k)}$ (resp. $e^{(k)}$) to refer to node $v \in \mathcal{V}$ (resp. link $e \in \mathcal{E}$) at layer k. We use each layer to find one segment of the total path for each traffic demand w. Consecutive layers are connected by artificial edges which are used to model the function selection at the nodes. More precisely, for each function node $v \in \mathcal{D}$ and each layer $k = 0, \ldots, K-1$, there is an artificial directed edge from node $v^{(k)}$ to node $v^{(k+1)}$. The interpretation is that a path for traffic demand w goes through this artificial link if and only if VNF f_k^w is executed at node $v \in \mathcal{D}(f_k^w)$. In the following, we shall use the notation $\ell_v^{(k)}$ to refer to the artificial edge $(v^{(k)}, v^{(k+1)})$ and we shall denote the set of artificial edges between layers $k = 0, \ldots, K-1$ and $k+1$ by $\mathcal{L}^{(k)} = \left\{ \ell_v^{(k)} \ : \ v \in \mathcal{D} \right\}$. In summary, the set of nodes of the expanded network $\mathcal{G}^*$ is

$$\mathcal{V}^* = \bigcup_{k=0}^{K} \mathcal{V}^{(k)},$$

whereas its set of edges is

$$\mathcal{E}^* = \bigcup_{k=0}^{K} \mathcal{E}^{(k)} \cup \bigcup_{k=0}^{K-1} \mathcal{L}^{(k)}.$$

The construction of the expanded network is shown in Figure 2 for the example network of Figure 1.

In the following, we say that an artificial edge $\ell_v^{(k)} \in \mathcal{L}^{(k)}$ is feasible for demand w if and only if $v \in \mathcal{D}(f_k^w)$, that is, it connects two consecutive copies of a node v hosting function f_k^w. A feasible path

$\pi_w \subset \mathcal{E}^*$ for the traffic demand $w \in \mathcal{W}$ is then defined as a path from node $s_w^{(0)}$ to node $t_w^{(K_w)}$ in the expanded network such that all artificial edges in the path are feasible for demand w. Figure 2 shows two feasible paths for the demand from node s_1 to node t_1. Note that the artificial edges appearing in a path specify the function nodes at which the VNFs are executed, while the other edges in the path specify the path segments used to reach these function nodes. The feasibility of a path ensures that the network functions are executed in the prescribed order.

Given a set $\pi \subset \mathcal{E}^*$ of edges in the expanded network and a link $e \in \mathcal{E}$ of the original network, we define the constant $\delta_\pi^{e,k}$ as 1 if $e^{(k)} \in \pi$, and 0 otherwise. Similarly, for any set $\pi \subseteq \mathcal{E}^*$ and any function node $v \in \mathcal{D}$, we define the constant $\delta_\pi^{v,k}$ as 1 if the artificial edge $\ell_v^{(k)} \in \pi$, and 0 otherwise. In the following, the notation $\delta_\pi^{r,k}$ will refer to one of the above-defined constants, depending on whether resource $r \in \mathcal{R}$ is a transmission link e or a function node v. We shall say for short that resource r appears in path π at layer k of the expanded network and write $(r,k) \in \pi$ when $\delta_\pi^{r,k} = 1$. Similarly, we write $r \in \pi$ if $\delta_\pi^{r,k} = 1$ for some k.

2.2 Single-Path Routing Problem

In the following, we shall assume that we are given a set Π_w of candidate feasible paths through the expanded network for each demand $w \in \mathcal{W}$. A feasible solution for the joint routing and VNF placement problem is then defined as a vector $\boldsymbol{\pi} = (\pi_w) \in \Pi$, where π_w is the path assigned to traffic demand w and $\Pi = \bigotimes_{w \in \mathcal{W}} \Pi_w$. Given such a feasible solution $\boldsymbol{\pi}$, the traffic sent by demand w on resource $r \in \mathcal{R}$ of the original network is:

$$y_r^w(\pi_w) = \sum_{k=0}^{K} \delta_{\pi_w}^{r,k}\, b_{k,r}^w, \tag{1}$$

from which it follows that the traffic flowing on resource r is $y_r(\boldsymbol{\pi}) = \sum_{w \in \mathcal{W}} y_r^w(\pi_w)$.

We assume that to each resource $r \in \mathcal{R}$ of the physical network is associated a non-decreasing function $\phi_r : \mathbb{R}_+ \to \mathbb{R}_+$ and that the cost of this resource has the form $y_r \phi_r(y_r)$. The function $\phi_r()$ may be interpreted as the cost per unit of capacity of resource r, and it may depend on the total traffic flowing on that resource. A typical example is the Kleinrock function $\phi_r(y_r) = 1/(c_r - y_r)$, where c_r is a constant representing the (transmission or processing) capacity of resource r. This function assumes that the total traffic on a resource is smaller than its capacity and that the cost per unit of capacity grows unboundedly as the former approaches the latter. Other examples are the linear function $\phi_r(y_r) = y_r/c_r$ or the quadratic function $\phi_r(y_r) = y_r^2/c_r^2$, which are valid even for $y_r \geq c_r$. In the above examples, the cost per unit capacity grows as the traffic y_r on the resource increases, in contrast with linear models in which ϕ_r is assumed to be a constant function. We emphasize that the functions $\phi_r()$ may differ for different types of resources or even from one resource to the other.

The goal is to find a feasible solution $\boldsymbol{\pi} \in \Pi$ that minimizes the network cost $F(\boldsymbol{\pi}) = \sum_{r \in \mathcal{R}} y_r(\boldsymbol{\pi})\, \phi_r\left(y_r(\boldsymbol{\pi})\right)$. Formally, the problem is as follows:

$$\text{minimize } F(\boldsymbol{\pi}) = \sum_{r \in \mathcal{R}} y_r(\boldsymbol{\pi})\, \phi_r\left(y_r(\boldsymbol{\pi})\right) \tag{OPT}$$

subject to:

$$\pi_w \in \Pi_w, w \in \mathcal{W}. \tag{2}$$

It is worth mentioning that problem (OPT) can be cast as a 0-1 mathematical programming problem by introducing the following binary variables:

$$x_{w,\pi} = \begin{cases} 1 & \text{if demand } w \text{ is routed along path } \pi \text{ in } \mathcal{G}^*, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

so that it becomes equivalent to:

$$\text{minimize} \sum_{r\in\mathcal{R}} y_r\,\phi_r(y_r)$$

subject to:

$$y_r = \sum_{w\in\mathcal{W}}\sum_{\pi\in\Pi_w}\left(\sum_{k=0}^{K}\delta_\pi^{r,k}\,b_{k,r}^{w}\right)x_{w,\pi},\ \text{for all } r\in\mathcal{R}, \tag{4}$$

$$\sum_{\pi\in\Pi_w} x_{w,\pi} = 1,\ \text{for all } w\in\mathcal{W}, \tag{5}$$

$$x_{w,\pi}\in\{0,1\},\ \text{for all } \pi\in\Pi_w \text{ and } w\in\mathcal{W}. \tag{6}$$

Except when the functions $\phi_r()$ are constant functions, the above problem belongs to the class of non-linear mathematical programs with integer variables, a class of mathematical programs which are known to be extremely hard to solve. This motivates the development of an efficient approximation method for solving problem (OPT).

3 Penalized Best-Response Algorithm

As discussed in Section 2.2, the joint routing and VNF placement problem can be cast as a non-linear single path routing problem in the expanded network. The problem differs however significantly from traditional single-path routing problems as on one hand the same resource appears at different layers of the expanded network and on the other hand the volume of a traffic demand may change from one layer to the other. Nevertheless the heuristic algorithm that we propose in this section is directly inspired from an algorithm proposed in [15] for solving such problems. The idea of the algorithm is to view the traffic demands as the players of a non-cooperative game in which each each player independently optimizes its own objective function. Starting from an arbitrary initial feasible solution, the algorithm then mimics the best-response dynamics of the game, that is, the players take turns in some order to adapt their strategy based on the most recent known strategy of the others. The player objective functions are designed in such a way that (a) the best-response dynamics converges to a Nash equilibrium of the game in a finite number of steps, (b) all optimal solutions of problem (OPT) are Nash equilibria of the game, and (c) an upper bound on the approximation ratio of the algorithm can be proven in some cases.

Let us think of the traffic demands as the players of the game. The strategy of player $w\in\mathcal{W}$ is the path π_w it chooses in the set Π_w, and a strategy profile of the game is a feasible solution to problem (OPT), that is, a vector $\boldsymbol{\pi}\in\Pi$. Given the strategy of the other players $\boldsymbol{\pi}_{-w} = (\pi_u)_{u\neq w}$, we shall assume that the player w seeks to solve the following problem:

$$\text{minimize}_{\pi\in\Pi_w}\, c_w(\pi,\boldsymbol{\pi}_{-w}) = f_w(\pi,\boldsymbol{\pi}_{-w}) + p_w(\pi,\boldsymbol{\pi}_{-w}), \tag{OPT-w}$$

where the value $f_w(\pi,\boldsymbol{\pi}_{-w})$ associated to path π by player w reflects the cost of this path, whereas the term $p_w(\pi,\boldsymbol{\pi}_{-w})$ is a penalty term measuring the impact of player w's choice on other players. More precisely, we shall assume that the selfish objective function of player w is:

$$f_w(\pi,\boldsymbol{\pi}_{-w}) = \sum_{r\in\pi} y_r^w(\pi)\,\phi_r\left(y_r^{-w} + y_r^w(\pi)\right), \tag{7}$$

where $y_r^{-w} = \sum_{u\neq w} y_r^u(\pi_u)$ is the total traffic sent over resource r by all the other players $u\neq w$. The term $f_w(\pi,\boldsymbol{\pi}_{-w})$ thus represents the cost of path π for player w, given the fixed strategies $\boldsymbol{\pi}_{-w}$ of the other players.

In order to define the penalty term $p_w(\pi,\boldsymbol{\pi}_{-w})$, assume that player w chooses path $\pi\in\Pi_w$ and consider a resource r appearing in the path π at some layer. Then, for all players $u\neq w$, the cost of this resource increases by:

$$y_r^u(\pi_u)\,\left[\phi_r\left(y_r^{-w} + y_r^w(\pi)\right) - \phi_r\left(y_r^{-w}\right)\right].$$

As a consequence, we define the penalty term $p_w(\pi,\boldsymbol{\pi}_{-w})$ as follows:

$$\begin{aligned} p_w(\pi,\boldsymbol{\pi}_{-w}) &= \sum_{r\in\mathcal{R}}\sum_{u\neq w} y_r^u(\pi_u)\,\left[\phi_r\left(y_r^{-w} + y_r^w(\pi)\right) - \phi_r\left(y_r^{-w}\right)\right] \\ &= \sum_{r\in\pi} y_r^{-w}\,\left[\phi_r\left(y_r^{-w} + y_r^w(\pi)\right) - \phi_r\left(y_r^{-w}\right)\right] \end{aligned} \tag{8}$$

With (7) and (8), the objective function $c_w(\pi, \boldsymbol{\pi}_{-w}) = f_w(\pi, \boldsymbol{\pi}_{-w}) + p_w(\pi, \boldsymbol{\pi}_{-w})$ of player w is perfectly defined. Given the strategies $\boldsymbol{\pi}_{-w}$ of the other players, the path π minimizing $c_w(\pi, \boldsymbol{\pi}_{-w})$ in problem (OPT-w) is known as the best response of player w.

The pseudocode of our heuristic is given in Algorithm 1. The algorithm starts from an initial feasible solution $\boldsymbol{\pi}^{(0)}$. At each iteration n, the players update their strategies in a given order by computing their best responses to the strategies of the others (lines $3-5$). Note that a player w deviates from its strategy $\pi_w^{(n)}$ at iteration n to a new strategy π' if and only if $c_w\left(\pi', \boldsymbol{\pi}_{-w}^{(n)}\right) < c_w\left(\boldsymbol{\pi}^{(n)}\right)$. The algorithm stops when no player can decrease its cost by unilaterally deviating from its strategy, that is, $\boldsymbol{\pi}^{(n+1)} = \boldsymbol{\pi}^{(n)}$, which means that a Nash equilibrium has been reached (line 7).

Algorithm 1 Penalized best-response

Require: $\boldsymbol{\pi}^{(0)}$
1: $n \leftarrow 0$
2: **repeat**
3: **for** $w \in \mathcal{W}$ **do**
4: $\pi_w^{(n+1)} \leftarrow \operatorname{argmin}(\text{OPT-}w)$
5: **end for**
6: $n \leftarrow n + 1$
7: **until** $\boldsymbol{\pi}^{(n+1)} \neq \boldsymbol{\pi}^{(n)}$.
8: **return** $\boldsymbol{\pi}^{(n)}$

Theorem 1 below states the main properties of Algorithm 1. The proofs of properties (a), (b) and (c) are straightforward adaptations of those of similar results in [15]. The proof of property (d) is notably more involved, though it follows the same lines as the proof of Theorem 3 in [15]. Note that property (c) follows from property (d) by taking $d = 0$.

Theorem 1. *Algorithm 1 has the following properties:*

(a) It converges in a finite number of steps.

(b) If it ever reaches a global optimum of problem (OPT)*, it returns this optimal solution.*

(c) It computes an optimal solution of problem (OPT) *for linear resource costs, that is, when the functions $\phi_r()$ are constant functions $\phi_r(y) = a_r$.*

(d) Its approximation ratio is $\left(2^{\frac{1}{d+1}} - 1\right)^{-(d+1)}$ when the functions $\phi_r()$ are polynomials of degree d, that is, $\phi_r(y) = \sum_{n=0}^{d} a_{r,n} y^n$.

Proof. See Appendix A. □

Algorithm 1 is optimal for linear resource costs and returns a solution whose cost is at most $5.83\times$ the cost of an optimal solution for quadratic resource costs. Unfortunately, the approximation ratio of the algorithm for polynomial resource costs degrades quickly with the degree of the polynomial (it is already equals to 56.9 for $d = 2$). Our numerical results suggest however that the heuristic provides close-to-optimal solutions in most cases.

The proof of Theorem 1 shows that the game that we consider is a *potential game*, which implies that the worst-case complexity of Algorithm 1 is exponential in the number of traffic demands [17]. However, as we will see in Section 4 below, its convergence in practice is much faster than suggested by this worst-case result.

4 Performance Evaluation

In this section, we experimentally evaluate the performance of the penalized best-response algorithm. We first describe the different types of objective functions used to solve the penalized

best-response algorithm and the procedure for generating random instances in sections 4.1 and 4.2. Then, we present in section 4.3 the numerical results for the different topologies evaluated under the different types of objective functions.

4.1 Cost Functions

Given the cost per-unit of capacity of resource r function ϕ_r, we denote by $\Phi_r = y_r\phi_r(y_r)$ the cost function of resource r. We consider four types of cost functions: linear, piece-wise linear, quadratic and the Kleinrock Function ($M/M/1$).

1. Linear cost function: the per-unit cost does not depend on the utilization rate of the resource r ($\phi_r(y_r) = a$, where a is a constant). Here, the objective function is linear with the amount of traffic which is solved using Gurobi 9.0 as the ILP solver. We note that in this case the solution may over utilize the resources capacity. We choose $a = 5$, and we have:

$$\Phi_r = 5 \times y_r \tag{9}$$

2. Piece-Wise Linear cost function: we consider the following increasing per-unit cost function:

$$\phi_r(y_r) = \begin{cases} 3 & \text{if } 0 \leq \frac{y_r}{c_r} \leq \frac{1}{4}, \\ 5 - \frac{1}{2}\frac{c_r}{y_r} & \text{if } \frac{1}{4} < \frac{y_r}{c_r} \leq \frac{3}{4}, \\ 10 - \frac{17}{4}\frac{c_r}{y_r} & \text{if } \frac{3}{4} < \frac{y_r}{c_r}. \end{cases} \tag{10}$$

   where c_r is the capacity of resource r. This function expresses that it is cheap to utilize a resource with a small utilization rate, whereas, as the load approaches c_r, it becomes more expensive. Problem (OTP) can then be formulated as a Mixed-Integer Linear Problem and solved using Gurobi 9.0 as the MILP solver:

$$\text{minimize } \sum_{r\in\mathcal{R}} \Phi_r$$

$$\text{subject to:}$$

$$y_r = \sum_{w\in\mathcal{W}} \sum_{\pi\in\Pi_w} \left(\sum_{k=0}^{K} \delta_\pi^{r,k}\, b_{k,r}^w \right) x_{w,\pi}, \text{ for all } r \in \mathcal{R}, \tag{11}$$

$$\sum_{\pi\in\Pi_w} x_{w,\pi} = 1, \text{ for all } w \in \mathcal{W}, \tag{12}$$

$$\Phi_r \geq a y_r - b c_r, \text{ for all } r \in \mathcal{R} \text{ and } (a,b) \in \mathcal{C}, \tag{13}$$

$$x_{w,\pi} \in \{0,1\}, \text{ for all } \pi \in \Pi_w \text{ and } w \in \mathcal{W}. \tag{14}$$

$$y_r \geq 0, \Phi_r \geq 0, \text{ for all } r \in \mathcal{R}, \tag{15}$$

   where $\mathcal{C} = \{(3,0), (5,\frac{1}{2}), (10,\frac{17}{4})\}$.

3. Quadratic cost function: The per-unit cost function is linear ($\phi_r(y_r) = a y_r$). This results in a quadratic objective function which is solved using Gurobi 9.0 as the Quadratic Integer Program solver. Similar to the previous function, the more is the utilization rate of a resource, the more expensive it becomes.

$$\Phi_r = y_r^2/c_r^2 \tag{16}$$

4. Kleinrock cost function ($M/M/1$): with this function, it becomes hard to find the optimal solution. However, it guarantees that the utilization of a resource r does not exceed its capacity c_r.

$$\Phi_r = \frac{y_r}{c_r - y_r} \tag{17}$$

   We formulate problem (OPT) as a Bilinear Program, and we solve it with Gurobi 9.0 Bilinear solver.

Table 2: Topologies : number of nodes and links.

Topology	# Nodes	# Links
AARNET	19	46
ARPANET	25	56
NSFNET	13	30
IBM	18	48
CESNET	10	18

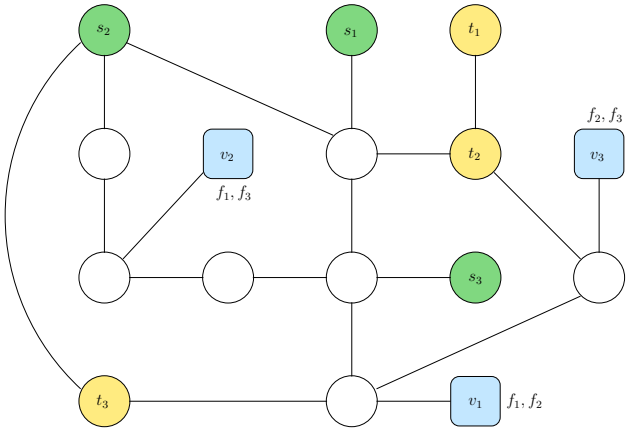


Figure 3: NSFNET topology which has been expanded with function nodes v_1, v_2 and v_3. The nodes s_1, s_2 and s_3 are the source nodes and the nodes t_1, t_2 and t_3 are the destination nodes of the traffic demands.

$$\text{minimize} \sum_{r \in \mathcal{R}} \Phi_r$$

subject to:

$$y_r = \sum_{w \in \mathcal{W}} \sum_{\pi \in \Pi_w} \left(\sum_{k=0}^{K} \delta_\pi^{r,k} \, b_{k,r}^w \right) x_{w,\pi}, \text{ for all } r \in \mathcal{R}, \tag{18}$$

$$\sum_{\pi \in \Pi_w} x_{w,\pi} = 1, \text{ for all } w \in \mathcal{W}, \tag{19}$$

$$y_r - \Phi_r c_r + \Phi_r y = 0, \text{ for all } r \in \mathcal{R}, \tag{20}$$

$$x_{w,\pi} \in \{0,1\}, \text{ for all } \pi \in \Pi_w \text{ and } w \in \mathcal{W}. \tag{21}$$

$$y_r \geq 0, \Phi_r \geq 0, \text{ for all } r \in \mathcal{R}, \tag{22}$$

4.2 Generation of Random Instances

In order to evaluate the performance of the penalized best-response algorithm, we use five network topologies (see Table 2) collected from the IEEE literature and from *The Internet Zoo Topology* [18]. As illustrated in Figure 3 for the NSFNET topology, we choose three source nodes (s_1, s_2 and s_3) and three destination nodes (t_1, t_2 and t_3) in each topology. The original topology is also expanded by adding three function nodes v_1, v_2 and v_3 and connecting them to the other nodes. We consider three VNFs f_1, f_2 and f_3 and assume that each one requires one core per unit of traffic. Function node v_1 can host VNFs f_1 and f_2, whereas function node v_2 (resp. v_3) can host VNFs f_1 and f_3 (resp. f_2 and f_3). As shown in Figure 3, each VNF is thus available at two different locations.

We then generate 100 random instances for each network topology. A random instance is composed of 25 traffic demands, which are randomly generated as follows. The source and destination

Table 3: Number of candidate paths Π_w.

Topology	**Max**	**Min**	**Average**
AARNET	128	32	52
ARPANET	128	32	47
NSFNET	128	32	48
IBM	128	32	51
CESNET	8	4	5

nodes of a traffic demand are chosen randomly in the sets $\{s_1, s_2, s_3\}$ and $\{t_1, t_2, t_3\}$, respectively, according to uniform distributions. The volume of a traffic demand is drawn from a uniform distribution in the interval $[1, 5]$, and, to simplify, we assume that it does not change along the path from the source to the destination. Finally, the processing path of a traffic demand in chosen uniformly at random in the set $\{(f_1, f_2), (f_1, f_3), (f_2, f_3), (f_1, f_2, f_3)\}$.

Each traffic demand must thus go through two or three VNFs, which are available at multiple locations. The possible paths for a traffic demand w are generated as follows. We first compute two possible paths between the source node s_w and each location where the first VNF f_1^w is available by solving a 2-shortest path problem (assuming unit weights for the edges of the expanded network). We then apply the same procedure for computing two possible paths between each possible location of the VNF f_1^w and each possible location of f_2^w, etc. The final set of candidate paths Π_w is obtained by connecting the different segments forming the overall path from s_w to t_w.

The initial solution is obtained by choosing randomly a path for each traffic demand. We assume that all links and all function nodes have the same capacity. This capacity is computed for each random instance so as to obtain a network congestion rate of 0.83.

We report in Table 3 the number of candidate paths for a traffic demand $|\Pi_w|$ calculated for each studied topology. The max (min, resp.) value is obtained when the size of the processing path is three (two, resp.) resulting in $|\Pi_w| = 2^7$ ($|\Pi_w| = 2^5$, resp.). We remind the reader that we have $k = 2$ candidate paths for each VNF option. The average value is obtained over 100 problem instances where traffic demands have a probability of 3/4 (1/4, resp.) to choose a processing path size of two (three, resp.). In the case of CESNET topology and under all the generated instances, we could not find two candidate paths for each VNF option, hence the reduced size of Π_w.

The final set of candidate paths calculated for each traffic demand is used to find the optimal solution with Gurobi solver. However, when running the penalized best-response algorithm, we consider only the k'-shortest paths from the previously obtained set. For instance, a traffic demand in the AARNET topology has on average 52 different paths that will be used to find the optimal solution. For $k' = 10$, only the 10 shortest paths will be considered in the penalized best-response algorithm.

4.3 Numerical Results

We report in Tables 4 and 5 the performance of the penalized Best Response (BR) algorithm for different cost functions under distinct topologies. We set $k' = 10$, and we evaluate the algorithm performance by calculating the relative gap between the penalized BR solution and the optimal solution from Gurobi. Following the obtained results, the penalized BR algorithm provides very good quality solutions for the different cost functions with impressive computation time. Its average relative gap to the optimal solution is under 5%, and it is almost negligible for small topologies such as CESNET that has a result value of at most 0.07%. The average execution time of the algorithm depends on the size of the topology and on the cost function used. It is also directly proportional to the number of paths considered for each demand. Nevertheless except for the PWL cost function where the MILP solver is known to be efficient and quick at solving, the solver is significantly slower for the other more complex cost functions. In the case of the quadratic cost function, the maximum relative error of the BR algorithm over the 100 instances is 3.725%. However, the if we look at the computing times in Table 5, we observe that the BR algorithm is at most 16 times faster and at worst 6.3 times faster than the solver. It is interesting to note that for the $M/M/1$ cost function, the solver has reached its time limit that has been set to 5 minutes,

Table 4: Relative gap (% ± std) to the optimal solution with $k' = 10$.

Topology	PWL	Quadratic	M/M/1
AARNET	0.71 ±0.65	1.81 ±0.85	1.43 ±1.76
ARPANET	1.15 ±0.5	3.725 ±1.55	1.5 ±0.79
NSFNET	0.42 ±0.49	1.67 ±1.12	0.31 ±0.32
IBM	1.7 ±0.8	2.85 ±0.98	0.53 ±0.71
CESNET	0.051 ±0.003	0.047 ±0.008	0.070 ±0.056

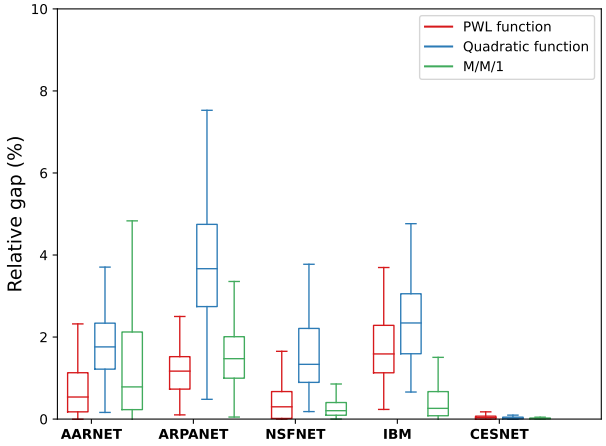


Figure 4: Box plot of the relative gap to the optimal solution.

Table 5: Average computing times (seconds±std) for the 100 problem instances with $k' = 10$.

Function Costs	AARNET		ARPANET		NSFNET		IBM		CESNET	
	BR	Gurobi	BR	Gurobi	BR	Gurobi	BR	Gurobi	BR	Gurobi
PWL	7.1±0	1.2	8.09±0	1.72	4.93±0	0.72	6.83±0	0.9	1.96±0	0.07
Quadratic	7.89±0	126±0	8.89±0	124±0	6.3±0	40±0	8.27±0	109±0	2.33±0	0.3
M/M/1	8.17±0	300	9.84±0	300	7.52±0	300	9.17	300	2.79±0	300

while the BR algorithm takes less than 10 seconds to execute with a solution as close as 1.43% to the solver.

The box plot in Figure 4 compares the distribution of the 100 instances generated for each scenario. The line that divides the box into two parts marks the median (Q2) of the data. The upper quartile (Q3) value is represented by the upper box line. 75% of the values fall below Q3. The lower quartile (Q1) value is the lower box line. 25% of the values fall below Q1. Hence, the values inside the box represent 50% of the data. The end of the lower and upper whiskers represents the minimum and maximum values, respectively. We first note that there are no outliers in the data samples which means that all the values of the data fall within the normal range. We observe that the values obtained with the quadratic cost function are more dispersed than the values obtained from the other cost functions.

In order to evaluate the choice of k', we report in Table 6 the average relative gap and the computing times over different values of k'. We evaluate the results with the quadratic cost function (similar results were observed for the other cost functions). As expected, the optimality gap decreases when more routing paths are added to the set of candidate paths. However, the decrease of this gap is insignificant for $k' > 10$. In the case of the CESNET topology and since on average there are only 5 possible paths for a given demand (refer to Table 3), the gap optimality remains constant for $k' \geq 5$. On the other hand, increasing the routing paths increases the running time of the BR algorithm. In this work, we found that the best trade-off between quality solution and running time is for $k' = 10$.

Table 6: Average relative gap and computing times for the 100 problem instances with the quadratic cost function.

Topology	$k' = 5$		$k' = 10$		$k' = 20$		$k' = 32$	
	gap	runtime	gap	runtime	gap	runtime	gap	runtime
AARNET	5.2%	3.8 sec	1.8%	7.9 sec	1.4%	17 sec	1.4%	26.7 sec
ARPANET	5.5%	4 sec	3.7%	8.9 sec	3%	19 sec	3%	30 sec
NSFNET	1.8%	2.9 sec	1.7%	6.3 sec	1.6%	12 sec	1.6%	18 sec
IBM	5.88%	3.5 sec	2.85%	8.2 sec	2.77%	12.8 sec	2.75%	30 sec
CESNET	0.05%	1.9 sec	0.05%	2.3 sec	0.05%	2.4 sec	0.05%	2.4 sec

5 Conclusion

In this paper, we have considered the offline VNF placement and chaining problem, assuming that a non-linear cost function is associated to each network resources. It is also assumed that the volume of a traffic demand can change after each VNF. With respect to previous works, the main originality of the considered model is that the cost per unit capacity of a resource is not constant, but instead that it grows with its utilization rate, which is an essential feature to achieve a better load distribution. We have formulated the problem as a single-path routing problem in an extended network and adapted an existing game-theoretic algorithm to solve it. Our numerical results suggest that the algorithm provides near-optimal solutions in a substantially lower computing times than the solver, in particular for highly non-linear cost functions.

References

[1] ISG NFV, ETSI, "Network Functions Virtualisation (NFV): Architectural framework," 2013.

[2] J. Gil Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[3] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[4] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2019.

[5] E. Amaldi, S. Coniglio, A. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electron. Notes Discret. Math.*, vol. 52, pp. 213–220, 2016.

[6] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "A green vnf-fg embedding algorithm," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018, pp. 141–149.

[7] M. Mechtri, C. Ghribi, and D. Zeghlache, "Vnf placement and chaining in distributed cloud," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016, pp. 376–383.

[8] F. Carpio, W. Bziuk, and A. Jukan, "Replication of virtual network functions: Optimizing link utilization and resource costs," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, pp. 521–526.

[9] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and multi-domain adaptive allocation algorithms for vnf forwarding graph embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 98–112, 2019.

[10] A. El Amine, O. Brun, S. Abdellatif, and P. Berthou, "Shortening the deployment time of SFC by adaptively querying resource providers," in *2021 IEEE Global Communications Conference: Next-Generation Networking and Internet (Globecom2021 NGNI)*, Madrid, Spain, Dec. 2021.

[11] 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)*. IEEE, 2015.

[12] N. Tastevin, M. Obadia, and M. Bouet, "A graph approach to placement of service functions chains," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017.

[13] X. Zhong, Y. Wang, and X. Qiu, "Cost-aware service function chaining with reliability guarantees in nfv-enabled inter-dc network," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019.

[14] H. Guo, Y. Wang, Z. Li, X. Qiu, H. An, N. Yuan *et al.*, "Cost-aware placement and chaining of service function chain with vnf instance sharing," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020.

[15] O. Brun, B. Prabhu, and J. Vallet, "A penalized best-response algorithm for non-linear single-path routing problems," *Networks*, vol. 69, no. 1, pp. 52–66, 2017.

[16] T. M. Nguyen, "Optimizing resource allocation in infrastructure networks based on network function virtualization," Ph.D. dissertation, Université Pierre et Marie Curie - Paris VI, 2017.

[17] A. Fabrikant, C. Papadimitriou, and K. Talwar, "The complexity of pure nash equilibria," in *Proc. of STOC'04*, ACM, Ed., New York, NY, USA, 2004, pp. 604–612.

[18] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, 2011.

A Proof of Theorem 1

Before proving Theorem 1, we remark that if Algorithm 1 converges, it necessarily converges to a Nash equilibrium of the game. By definition, the game is in a Nash equilibrium if and only if no player has anything to gain by unilaterally deviating from its current strategy, that is, the current strategy of each player is its best response to the strategy of the others. Formally, $\boldsymbol{\pi}^{(n)} = \left(\pi_w^{(n)}\right)_{w\in\mathcal{W}}$ is a Nash equilibrium of our game if and only if:

$$c_w(\pi_w^{(n)}, \boldsymbol{\pi}_{-w}^{(n)}) \leq c_w(\pi, \boldsymbol{\pi}_{-w}^{(n)}), \forall \pi \in \Pi_w, \forall w \in \mathcal{W},$$

which is precisely the condition under which Algorithm 1 stops (see line 7).

In order to prove property (d), we will need the following technical lemma which is an extension of Lemma 2 in [15].

Lemma 1. *Let $d \in \mathbb{N}$ and $x \geq 0$. For any non-negative numbers $a_1, a_2, \ldots, a_N$ and $b_1, b_2, \ldots, b_N$ such that $a_i b_i = 0$ for all $i = 1, 2, \ldots, N$, it holds that*

$$\bar{x}^{d+1} - x^{d+1} = \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} \left[a_j \left(\bar{x} + b_j\right)^k - b_j \left(\bar{x} - a_j\right)^k \right], \tag{23}$$

where $\bar{x} = x + \sum_{j=1}^{N}(a_j - b_j)$

Proof. We have

$$\begin{aligned}
\bar{x}^{d+1} - x^{d+1} &= \left(\sum_{j=1}^{N} a_j - \sum_{j=1}^{N} b_j\right) \sum_{k=0}^{d} \bar{x}^k x^{d-k}, \\
&= \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} a_j \bar{x}^k - \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} b_j \bar{x}^k, \\
&= \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} a_j (\bar{x} + b_j - b_j)^k - \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} b_j (\bar{x} + a_j - a_j)^k, \\
&= \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} a_j \sum_{n=0}^{k} \binom{k}{n} (\bar{x} + b_j)^n (-b_j)^{k-n} \\
&\quad - \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} b_j \sum_{n=0}^{k} \binom{k}{n} (\bar{x} - a_j)^n (a_j)^{k-n}, \\
&= \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} a_j (\bar{x} + b_j)^k - \sum_{k=0}^{d} x^{d-k} \sum_{j=1}^{N} b_j (\bar{x} - a_j)^k,
\end{aligned}$$

where the last equality follows from $a_j b_j = 0$. □

We will also need Lemma 2 below (which is basically a simple application of Hölder's inequality).

Lemma 2. *For non-negative numbers a_e, x_e z_e, and non-negative integer m,*

$$\sum_e a_e (x_e + z_e)^{m+1} \leq \left(\left(\sum_e a_e x_e^{m+1} \right)^{1/(m+1)} + \left(\sum_e a_e z_e^{m+1} \right)^{1/(m+1)} \right)^{m+1}. \tag{24}$$

Proof. See the proof of Lemma 3 in [15]. □

We are now in position to prove Theorem 1.

Proof of Theorem 1. We first prove **property (a)**. We observe that the cost function $c_w(\pi_w, \boldsymbol{\pi}_{-w})$ of player w can be rewritten as follows

$$\begin{aligned}
c_w(\pi, \boldsymbol{\pi}_{-w}) &= f_w(\pi, \boldsymbol{\pi}_{-w}) + p_w(\pi, \boldsymbol{\pi}_{-w}), \\
&= \sum_{r\in\pi} y_r^w(\pi)\, \phi_r\left(y_r^{-w} + y_r^w(\pi)\right) + \sum_{r\in\pi} y_r^{-w}\, \left[\phi_r\left(y_r^{-w} + y_r^w(\pi)\right) - \phi_r\left(y_r^{-w}\right)\right], \\
&= \sum_{r\in\pi} y_r(\pi, \boldsymbol{\pi}_{-w})\, \phi_r\left(y_r(\pi, \boldsymbol{\pi}_{-w})\right) - \sum_{r\in\pi} y_r^{-w} \phi_r\left(y_r^{-w}\right), \\
&= \sum_{r\in\mathcal{R}} y_r(\pi, \boldsymbol{\pi}_{-w})\, \phi_r\left(y_r(\pi, \boldsymbol{\pi}_{-w})\right) - \left(\sum_{r\notin\pi} y_r^{-w} \phi_r\left(y_r^{-w}\right) + \sum_{r\in\pi} y_r^{-w} \phi_r\left(y_r^{-w}\right)\right), \\
&= F(\pi, \boldsymbol{\pi}_{-w}) - \sum_{r\in\mathcal{R}} y_r^{-w} \phi_r\left(y_r^{-w}\right), \qquad (25)
\end{aligned}$$

where the penultimate equality is obtained by noting that $y_r(\pi, \boldsymbol{\pi}_{-w}) = y_r^{-w}$ for all resources $r \notin \pi$. Introducing $F(\boldsymbol{\pi}_{-w}) = \sum_{r\in\mathcal{R}} y_r^{-w} \phi_r\left(y_r^{-w}\right)$, which represents the network cost that would be obtained if demand w was removed from the network, (25) yields

$$\begin{aligned}
c_w(\pi', \boldsymbol{\pi}_{-w}) - c_w(\pi, \boldsymbol{\pi}_{-w}) &= \left(F(\pi', \boldsymbol{\pi}_{-w}) - F(\boldsymbol{\pi}_{-w})\right) - \left(F(\pi, \boldsymbol{\pi}_{-w}) - F(\boldsymbol{\pi}_{-w})\right), \\
&= F(\pi', \boldsymbol{\pi}_{-w}) - F(\pi, \boldsymbol{\pi}_{-w}), \qquad (26)
\end{aligned}$$

which means that the network cost function $F()$ is a potential function of the game, that is, any best-response move of a player w decreases the total network cost. As Algorithm 1 computes the best response of all players at each iteration and as the network cost $F()$ can take only a finite number of values, this implies the convergence of the algorithm in a finite number of steps.

We now prove **property (b)**. Assume that Algorithm 1 has reached an optimal solution $\boldsymbol{\pi}^*$ of problem (OPT). The strategy of each player w is therefore π_w^*. For any player w and any path $\pi \in \Pi_w$, it follows from (26) that

$$c_w(\pi, \boldsymbol{\pi}_{-w}^*) - c_w(\pi_w^*, \boldsymbol{\pi}_{-w}^*) = F(\pi, \boldsymbol{\pi}_{-w}^*) - F(\boldsymbol{\pi}^*) \geq 0, \qquad (27)$$

which implies that $\boldsymbol{\pi}^*$ is Nash equilibrium of our game. Hence, Algorithm 1 returns this solution.

Consider now **property (c)**. Note that it directly follows from property (d). Nevertheless, we provide a direct proof below. Assume that $\phi_r(y) = a_r$ for all $y \geq 0$ and all resources $r \in \mathcal{R}$. The network cost $F(\boldsymbol{\pi})$ can then be written as follows

$$F(\boldsymbol{\pi}) = \sum_{r\in\mathcal{R}} y_r(\boldsymbol{\pi})\, \phi_r\left(y_r(\boldsymbol{\pi})\right) = \sum_{r\in\mathcal{R}} \sum_{w\in\mathcal{W}} \sum_{k=0}^{K} a_r \delta_{\pi_w}^{r,k} b_{k,r}^w = \sum_{w\in\mathcal{W}} \left(\sum_{(r,k)\in\pi_w} a_r b_{k,r}^w\right), \qquad (28)$$

from which it follows that it is optimal to route each demand w on a path

$$\pi_w \in \operatorname{argmin}_{\pi\in\Pi_w} \left(\sum_{(r,k)\in\pi} a_r b_{k,r}^w\right).$$

Note from (8) that $p_w(\pi, \boldsymbol{\pi}_{-w}) = 0$ in this case, from which it follows that

$$c_w(\pi, \boldsymbol{\pi}_{-w}) = f_w(\pi, \boldsymbol{\pi}_{-w}) = \sum_{r\in\pi} a_r y_r^w(\pi) = \sum_{(r,k)\in\pi} a_r b_{k,r}^w. \qquad (29)$$

Therefore, the best-response strategy of a player is an optimal path, independently of the strategies of the other players. This implies that Algorithm 1 returns a globally optimal solution for linear resource costs.

Finally, let us prove **property (d)**. We focus on the case of monomial functions $\phi_r(y) = a_r y^d$, as the extension to polynomials is straightforward (see the proof of Theorem 3 in [15]). In that case, we have

$$F(\boldsymbol{\pi}) = \sum_{r \in \mathcal{R}} a_r y_r(\boldsymbol{\pi})^{d+1}. \tag{30}$$

Given an instance of the problem, let $\boldsymbol{\pi}$ be the solution returned by Algorithm 1 and $\boldsymbol{\pi}^*$ be any optimal solution. Consider a traffic demand $w \in \mathcal{W}$ and consider the solution obtained from $\boldsymbol{\pi}$ by deviating this demand from path π_w to path π_w^*, with the paths of the other demands unchanged. As $\boldsymbol{\pi}$ is a Nash equilibrium, we have $c_w(\pi_w^*, \boldsymbol{\pi}_{-w}) \geq c_w(\pi_w, \boldsymbol{\pi}_{-w})$, and with (26) it follows that

$$F(\pi_w^*, \boldsymbol{\pi}_{-w}) \geq F(\boldsymbol{\pi}). \tag{31}$$

Consider now the impact of this deviation on the traffic flowing on the edges of the expanded network. Let $y_{r,k}^w(\pi_w)$ and $y_{r,k}^w(\pi_w^*)$ be the traffic sent by demand w on the edge (r,k) of the expanded network before and after the deviation, respectively. We have

$$\begin{aligned} y_{r,k}^w(\pi_w^*) &= \delta_{\pi_w^*}^{r,k} b_{k,r}^w, \\ &= \delta_{\pi_w}^{r,k} b_{k,r}^w + (\delta_{\pi_w^*}^{r,k} - \delta_{\pi_w}^{r,k}) b_{k,r}^w, \\ &= y_{r,k}^w(\pi_w) + (\delta_{\pi_w^*}^{r,k} - \delta_{\pi_w}^{r,k}) b_{k,r}^w, \end{aligned} \tag{32}$$

from which we obtain

$$y_r^w(\pi_w^*) = \sum_{k=0}^{K} y_{r,k}^w(\pi_w^*) = y_r^w(\pi_w) + \sum_{k=0}^{K} (\delta_{\pi_w^*}^{r,k} - \delta_{\pi_w}^{r,k}) b_{k,r}^w, \tag{33}$$

and

$$\begin{aligned} y_r(\pi_w^*, \boldsymbol{\pi}_{-w}) &= y_r^w(\pi_w^*) + y_r^{-w}, \\ &= y_r^w(\pi_w) + y_r^{-w} + \sum_{k=0}^{K} (\delta_{\pi_w^*}^{r,k} - \delta_{\pi_w}^{r,k}) b_{k,r}^w, \\ &= y_r(\boldsymbol{\pi}) + \sum_{k=0}^{K} (\delta_{\pi_w^*}^{r,k} - \delta_{\pi_w}^{r,k}) b_{k,r}^w. \end{aligned} \tag{34}$$

With (31), it yields the following inequality

$$\sum_{r \in \mathcal{R}} a_r \left[\left(y_r(\boldsymbol{\pi}) + \sum_{k=0}^{K} (\delta_{\pi_w^*}^{r,k} - \delta_{\pi_w}^{r,k}) b_{k,r}^w \right)^{d+1} - y_r(\boldsymbol{\pi})^{d+1} \right] \geq 0. \tag{35}$$

Let us now introduce the following constants

$$\alpha_{r,k}^+(w) = \begin{cases} 1 & \text{if } (r,k) \in \pi_w^* \setminus \pi_w, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and } \alpha_{r,k}^-(w) = \begin{cases} 1 & \text{if } (r,k) \in \pi_w \setminus \pi_w^*, \\ 0 & \text{otherwise,} \end{cases}$$

The constant $\alpha_{r,k}^+(w)$ is 1 if the deviation increases the traffic of demand w on the edge (r,k) of the expanded network, and similarly, $\alpha_{r,k}^-(w)$ is 1 if the deviation decreases the traffic of demand w on the edge (r,k) of the expanded network. Note that $\alpha_{r,k}^+(w)\, \alpha_{r,k}^-(w) = 0$. As $\delta_{\pi_w^*}^{r,k} - \delta_{\pi_w}^{r,k} = \alpha_{r,k}^+(w) - \alpha_{r,k}^-(w)$, we can rewrite (35) as follows

$$\sum_{r \in \mathcal{R}} a_r \left[\left(y_r(\boldsymbol{\pi}) + \sum_{k=0}^{K} (\alpha_{r,k}^+(w) - \alpha_{r,k}^-(w)) b_{k,r}^w \right)^{d+1} - y_r(\boldsymbol{\pi})^{d+1} \right] \geq 0. \tag{36}$$

Applying Lemma 23 to (36) with $a_k = \alpha_{r,k}^+(w) b_{k,r}^w$ and $b_k = \alpha_{r,k}^-(w) b_{k,r}^w$, we obtain

$$
\begin{aligned}
0 \leq \sum_{r\in\mathcal{R}} a_r \sum_{n=0}^{d} y_r(\boldsymbol{\pi})^{d-n} \sum_{k=0}^{K} \Bigg[& \alpha^+_{r,k}(w) b^w_{k,r} \left(y_r(\boldsymbol{\pi}) + \sum_j \alpha^+_{r,j}(w) b^w_j - \sum_{j\neq k} \alpha^-_{r,j}(w) b^w_j \right)^n \\
& - \alpha^-_{r,k}(w) b^w_{k,r} \left(y_r(\boldsymbol{\pi}) + \sum_{j\neq k} \alpha^+_{r,j}(w) b^w_j - \sum_j \alpha^-_{r,j}(w) b^w_j \right)^n \Bigg] \\
= \sum_{r\in\mathcal{R}} a_r \sum_{n=1}^{d} y_r(\boldsymbol{\pi})^{d-n} \sum_{k=0}^{K} \Bigg[& \alpha^+_{r,k}(w) b^w_{k,r} \left(y_r(\boldsymbol{\pi}) + \sum_j \alpha^+_{r,j}(w) b^w_j - \sum_{j\neq k} \alpha^-_{r,j}(w) b^w_j \right)^n \\
& - \alpha^-_{r,k}(w) b^w_{k,r} \left(y_r(\boldsymbol{\pi}) + \sum_{j\neq k} \alpha^+_{r,j}(w) b^w_j - \sum_j \alpha^-_{r,j}(w) b^w_j \right)^n \Bigg] \\
& + \sum_{r\in\mathcal{R}} a_r y_r(\boldsymbol{\pi})^d \sum_{k=0}^{K} (\alpha^+_{r,k}(w) - \alpha^-_{r,k}(w)) b^w_{k,r},
\end{aligned}
$$

and it yields

$$
\begin{aligned}
0 \leq \sum_{r\in\mathcal{R}} a_r \sum_{n=1}^{d} y_r(\boldsymbol{\pi})^{d-n} \sum_{k=0}^{K} & \alpha^+_{r,k}(w) b^w_{k,r} \left(y_r(\boldsymbol{\pi}) + \sum_j \alpha^+_{r,j}(w) b^w_j - \sum_{j\neq k} \alpha^-_{r,j}(w) b^w_j \right)^n \\
& + \sum_{r\in\mathcal{R}} a_r y_r(\boldsymbol{\pi})^d \sum_{k=0}^{K} (\alpha^+_{r,k}(w) - \alpha^-_{r,k}(w)) b^w_{k,r}. \qquad (37)
\end{aligned}
$$

Observe that

$$
\begin{aligned}
\sum_j \alpha^+_{r,j}(w) b^w_j - \sum_{j\neq k} \alpha^-_{r,j}(w) b^w_j &\leq \sum_j \alpha^+_{r,j}(w) b^w_j, \\
&\leq \sum_j \delta^{r,k}_{\pi^*_w} b^w_j, \\
&= y^w_r(\pi^*_w), \\
&\leq y_r(\boldsymbol{\pi}^*),
\end{aligned}
$$

so that (37) implies

$$
\begin{aligned}
\sum_{r\in\mathcal{R}} a_r \sum_{n=1}^{d} y_r(\boldsymbol{\pi})^{d-n} \sum_{k=0}^{K} & \alpha^+_{r,k}(w) b^w_{k,r} \left(y_r(\boldsymbol{\pi}) + y_r(\boldsymbol{\pi}^*) \right)^n \\
& + \sum_{r\in\mathcal{R}} a_r y_r(\boldsymbol{\pi})^d \sum_{k=0}^{K} (\alpha^+_{r,k}(w) - \alpha^-_{r,k}(w)) b^w_{k,r} \geq 0. \qquad (38)
\end{aligned}
$$

Observe that

$$
\sum_{w\in\mathcal{W}} \sum_{k=0}^{K} \alpha^+_{r,k}(w) b^w_{k,r} \leq \sum_{w\in\mathcal{W}} \sum_{k=0}^{K} \delta^{r,k}_{\pi^*_w} b^w_{k,r} = y_r(\boldsymbol{\pi}^*),
$$

and

$$
\sum_{w\in\mathcal{W}} \sum_{k=0}^{K} (\alpha^+_{r,k}(w) - \alpha^-_{r,k}(w)) b^w_{k,r} = \sum_{w\in\mathcal{W}} \sum_{k=0}^{K} (\delta^{r,k}_{\pi^*_w} - \delta^{r,k}_{\pi_w}) b^w_{k,r} = y_r(\boldsymbol{\pi}^*) - y_r(\boldsymbol{\pi}).
$$

Hence, summing inequalities (38) over all $w \in \mathcal{W}$, we get

$$\sum_{r\in\mathcal{R}} a_r y_r(\boldsymbol{\pi}^*) \sum_{n=1}^{d} y_r(\boldsymbol{\pi})^{d-n} \left(y_r(\boldsymbol{\pi}) + y_r(\boldsymbol{\pi}^*)\right)^n + \sum_{r\in\mathcal{R}} a_r y_r(\boldsymbol{\pi})^d \left[y_r(\boldsymbol{\pi}^*) - y_r(\boldsymbol{\pi})\right] \geq 0, \tag{39}$$

which gives

$$\begin{aligned} F(\boldsymbol{\pi}) &\leq \sum_{r\in\mathcal{R}} a_r y_r(\boldsymbol{\pi}^*) \sum_{n=0}^{d} y_r(\boldsymbol{\pi})^{d-n} \left(y_r(\boldsymbol{\pi}) + y_r(\boldsymbol{\pi}^*)\right)^n, \\ &= \sum_{r\in\mathcal{R}} a_r \left[\left(y_r(\boldsymbol{\pi}) + y_r(\boldsymbol{\pi}^*)\right)^{d+1} - y_r(\boldsymbol{\pi})^{d+1}\right], \\ &= \sum_{r\in\mathcal{R}} a_r \left(y_r(\boldsymbol{\pi}) + y_r(\boldsymbol{\pi}^*)\right)^{d+1} - F(\boldsymbol{\pi}). \end{aligned} \tag{40}$$

With Lemma 2, it yields,

$$\begin{aligned} 2F(\boldsymbol{\pi}) &\leq \left[\left(\sum_{r\in\mathcal{R}} a_r y_r(\boldsymbol{\pi})^{d+1}\right)^{\frac{1}{d+1}} + \left(\sum_{r\in\mathcal{R}} a_r y_r(\boldsymbol{\pi}^*)^{d+1}\right)^{\frac{1}{d+1}}\right]^{d+1}, \\ &= \left[F(\boldsymbol{\pi})^{\frac{1}{d+1}} + F(\boldsymbol{\pi}^*)^{\frac{1}{d+1}}\right]^{d+1}, \end{aligned} \tag{41}$$

from which we obtain

$$\frac{F(\boldsymbol{\pi})}{F(\boldsymbol{\pi}^*)} \leq \left(\frac{1}{2^{\frac{1}{d+1}} - 1}\right)^{d+1}, \tag{42}$$

which concludes the proof. □