A Mixed-Integer Linear Programming Approach for Congestion-Aware Optimized NFV Deployment

A Mixed-Integer Linear Programming Approach for Congestion-Aware Optimized NFV Deployment

Mohammad Ali Raayatpanah*, Thomas Weise†, Jocelyne Elias‡, Fabio Martignon§, Andrea Pimpinella§
* *Mathematical Sciences and Computer, Kharazmi University,Tehran, Iran*
† *Institute of Applied Optimization, School of Artificial Intelligence and Big Data, Hefei University, Hefei, Anhui, China*
‡ *Department of Computer Science and Engineering, University of Bologna, Italy*
§ *Department of Management, Information and Production Engineering, University of Bergamo, Italy*

***Abstract*—This paper introduces a novel optimization framework for Network Functions Virtualization (NFV) that addresses the efficient implementation of end-to-end service requests in physical networks. Our approach characterizes each server node by a reliability function reflecting its computational load, which aids in balancing workloads and mitigating congestion. By optimizing the reliability metrics along the route, our approach ensures robust end-to-end service quality. We formulate the NFV deployment problem as a non-convex mixed-integer nonlinear programming (MINLP) model aimed at minimizing both deployment and operational costs while maximizing resource utilization. Given the NP-hard nature of the problem, we develop efficient linearization techniques and bounding schemes, using also dynamic programming, to convert the formulation into a tractable mixed-integer linear programming (MILP) model. Additionally, a cutting-plane-based heuristic with a warm-start strategy is proposed to further accelerate convergence. Experimental evaluations on real-world network topologies demonstrate that our framework offers scalable and cost-effective solutions compared to existing approaches.**

***Index Terms*—Network Functions Virtualization (NFV), Non-Convex Optimization, Mixed-Integer Nonlinear Programming (MINLP), Resource Allocation, Cost Modeling, Cutting-Plane Heuristic**

I. Introduction

Network Functions Virtualization (NFV) has revolutionized modern network architectures by decoupling network services from dedicated hardware and implementing them as software running on general-purpose servers [1]. This paradigm shift enables dynamic deployment of Virtual Network Functions (VNFs) that can be instantiated and executed by Communication Service Providers (CSPs) to efficiently scale services based on demand. Beyond optimizing resource utilization, NFV enhances network flexibility, and reduces both capital (CAPEX) and operational (OPEX) expenditures. However, despite its promises, the deployment and optimization of NFV systems pose complex challenges, particularly in ensuring efficient resource allocation, network congestion management and service reliability [2].

In particular, the design of effective NFV systems necessitates robust optimization models to efficiently allocate VNFs to commercial off-the-shelf server nodes while accounting for computational capacity, traffic demand, and service reliability constraints. This problem is inherently complex [3], requiring advanced mathematical models to balance cost minimization, efficient resource utilization, and service quality. The non-convex nature of the problem adds complexity, requiring specialized techniques for effective decision-making.

In this paper, we address such challenges by introducing novel, advanced optimization techniques tailored for NFV deployments. By strategically assigning VNFs to servers based on their congestion levels and demands, our approach optimizes workload distribution and ensures both high resource utilization and robust service delivery. Precisely, we provide the following contributions:

- *Mathematical Modeling*: we formulate the NFV deployment problem as a non-convex Mixed-Integer Non-Linear Programming (MINLP) model [4], incorporating routing constraints, VNF placement decisions, and network capacity limitations.
- *Linearization Techniques*: we introduce a novel approach to transform non-linear constraints, using also dynamic programming, obtaining a tractable MILP model, thereby enabling efficient optimization and solution scalability.
- *Heuristic Optimization*: to tackle the NP-hard nature of large-scale NFV optimization, we develop a cutting-plane-based heuristic algorithm with bounding schemes. We further enhance computational efficiency by integrating a warm-start strategy that accelerates convergence.
- *Performance Evaluation*: we validate our approach using a real-world network topology, demonstrating its effectiveness in optimizing VNF placement, improving network efficiency, and ensuring robust service delivery.

The structure of this paper is as follows: Section II defines the NFV resource allocation problem, outlining its key components. Section III presents the mathematical formulation, detailing the objective function and constraints governing VNF placement and traffic routing. Section IV introduces a cutting-plane heuristic that constructs lower and upper bounding models to relax complex constraints in the NP-hard NFV deployment problem. It also incorporates a warm-start strategy to generate initial tight cuts, enhancing convergence. Section V analyzes numerical results that demonstrate the model's effectiveness in a real network topology. Section VI discusses related works, and Section VII concludes with future research directions.

II. Problem Statement

We study network functions virtualization for end-to-end requests in a physical network where each request, estimated from service contracts, must be routed reliably. Each server node is endowed with a *reliability function*—a concave, non-increasing function of the computational resources used to process Virtual Network Functions. This mechanism distributes workloads efficiently, thereby mitigating network congestion. In our model, the probability of successful data transfer from source to destination is given by the product of the reliability functions of the servers along the path.

Our objective is to minimize the overall NFV deployment cost, which comprises **capital expenditures** (costs of deploying VNFs on servers), **link activation costs** (costs associated with activating links), and **server activation costs** (costs associated with activating servers).

Even though the cost function explicitly minimizes these expenses, it implicitly encourages efficient resource utilization and ensures that end-to-end request delivery meets a specified reliability threshold. In particular, for each request k, the product of the reliability functions along its path must exceed a threshold τ_k.

We formulate this problem as a non-convex mixed-integer nonlinear programming (MINLP) problem, proven to be NP-hard. To solve it, we introduce an efficient linearization technique together with lower and upper bounding schemes used in a cutting-plane algorithm-based heuristic.

The physical network is modeled as an undirected graph $G = (N, L)$ where each node (server) n has:

- A computational resource capacity (sc_n)
- A bandwidth capacity (sb_n)
- An activation cost (sa_n).

Table I summarizes the notations for the parameters and variables used throughout this paper. VNFs are of various types ($\forall f, f \in F$) with a processing capacity θ_f. End-to-end requests k are described by their source o^k, destination t^k, required bandwidth b^k, and the VNFs from set F needed to fulfill the request (the set F^k) with instance requirements. The required number of instances for each VNF f is determined by $m_f^k = \left\lceil \frac{b^k}{\theta_f} \right\rceil$.

Each VNF instance deployed on a server incurs a deployment cost $\alpha_{n,f}$ and requires computational resources $\eta_{n,f}^k$. Link capacities $lc_{i,j}$ and their costs $la_{i,j}$ are also modeled. The reliability of each server node is defined as $R_n = H(L_n)$, where L_n is the cumulative computational load from the VNFs hosted on n. For an end-to-end request, the overall path reliability is the product of the reliabilities of the nodes traversed, and must be at least τ_k.

TABLE I: Notations for parameters and variables

Parameter	Description
$G(N, L)$	Physical network G with nodes $n \in N$ and links L
sb_n	Bandwidth limit of server $n \in N$
sc_n	Comput. resource (e.g., CPU) capacity of server $n \in N$
sa_n	Cost to utilize physical resources on server $n \in N$ (Activation cost)
$la_{i,j}$	Cost to use link $(i, j) \in L$
$lc_{i,j}$	Capacity of physical link $(i, j) \in L$
K	Set of end-to-end requests $k \in K$
o^k, t^k	Source and destination node for request $k \in K$
b^k	Required bandwidth for request $k \in K$
τ_k	Smallest acceptable probability to fulfill request $k \in K$ with $0 \leq \tau_k \leq 1$
F	Set of possible VNF types $f \in F$
F^k	VNFs from set F needed to fulfill request $k \in K$
$\alpha_{n,f}$	Cost to deploy a VNF instance $f \in F$ on server node $n \in N$
$\eta_{n,f}^k$	Required computational resources for an instance of VNF $f \in F$ deployed at node $n \in N$ for request $k \in K$
θ_f	Maximum traffic that one instance of VNF $f \in F$ can process
m_f^k	Required nbr. of instances of VNF $f \in F$ for request $k \in K$

Variable	Description
x_n^k	Binary variable indicating whether request $k \in k$ is routed through switch $n \in N$: if yes, $x_n^k = 1$, otherwise $x_n^k = 0$.
$y_{i,j}^k$	Binary variable indicating whether request $k \in K$ is routed through link $(i, j) \in L$: if yes, $y_{i,j}^k = 1$, otherwise $y_{i,j}^k = 0$
ρ_{fn}^k	Integer variable representing the number of instance functions $f \in F$ deployed at server $n \in N$ for request $k \in K$
L_n	Total computational resources consumed on server $n \in N$
U_n	Bandwidth consumed on server $n \in N$
R_n	Reliability of server $n \in N$
p_n^k	Probability that end-to-end request $k \in K$ successfully reaches switch $n \in N$ from its source.

III. Mathematical Formulation

Our goal is to determine the optimal location of network function instances while satisfying the following constraints.

Routing Constraints: each request $k \in K$ must be routed through a single physical path, defined by binary variables $y_{i,j}^k$ for links $(i, j) \in L$ and x_n^k for nodes $n \in N$. Flow conservation is ensured by:

$$\sum_{j:(i,j)\in L} y_{i,j}^k - \sum_{j:(j,i)\in L} y_{j,i}^k = \begin{cases} 1, & i = o^k, \\ -1, & i = t^k, \\ 0, & \text{otherwise} \end{cases} \qquad \forall k \in K, i \in N. \tag{1}$$

Node-based routing constraints are given by:

$$x_n^k \leq \sum_{\{j|(n,j)\in L\}} (y_{n,j}^k + y_{j,n}^k), \quad \forall k \in K, n \in N, \tag{2}$$

$$x_n^k \geq (y_{n,j}^k + y_{j,n}^k), \qquad \forall k \in K, n \in N, (n, j) \in L. \tag{3}$$

Capacity Constraints: the total bandwidth used on each link should not exceed its capacity:

$$\sum_{k\in K} y_{i,j}^k * b^k \leq lc_{i,j}, \qquad \forall (i, j) \in L. \tag{4}$$

Similarly, the total bandwidth used on a server node is given by:

$$U_n = \sum_{k\in K} \sum_{f\in F^k} \theta_f \rho_{f,n}^k, \qquad \forall n \in N, \tag{5}$$

$$U_n \leq sb_n, \qquad \forall n \in N. \tag{6}$$

VNF Placement: instances of VNF $f \in F$ for request k should be deployed on the nodes along the request's path:

$$\rho^k_{fn} \leq m^k_f * x^k_n, \qquad \forall n \in N, k \in K, f \in F^k, \tag{7}$$

$$\sum_{n \in N} \rho^k_{fn} = m^k_f, \qquad \forall k \in K, f \in F^k. \tag{8}$$

The total computational resources used on each node must not exceed its capacity:

$$L_n = \sum_{k \in K} \sum_{f \in F^k} \eta^k_{n,f} * \rho^k_{fn}, \qquad \forall n \in N, \tag{9}$$

$$L_n \leq sc_n, \qquad \forall n \in N. \tag{10}$$

Reliability Constraints: the reliability of a server node is modeled as a function of its computational load:

$$R_n = H(L_n), \qquad \forall n \in N. \tag{11}$$

The probability of successful transmission for each request should meet a minimum threshold:

$$\prod_{n \in N} R_n^{x^k_n} \geq \tau_k, \qquad \forall k \in K. \tag{12}$$

A. Cost Modeling

Resource allocation in NFV is balanced against service performance. Deploying more VNFs can improve performance but increases cost. We categorize costs as follows:

Capital Expenditures

The deployment of VNF instances on servers incurs costs due to factors such as standby energy consumption, licensing fees, image transfer, and booting expenses. The total capital expenditures are given by:

$$C_{capex} = \sum_{k \in K} \sum_{f \in F} \sum_{n \in N} \rho^k_{fn} \alpha_{n,f}.$$

Link Activation Cost

Based on an on/off power model, each activated link incurs a cost independent of traffic volume. The total link activation cost is:

$$C_{bandwidth} = \sum_{k \in K} \sum_{(i,j) \in L} y^k_{i,j} la_{i,j}.$$

Server Activation Cost

The cost of using server nodes, which varies based on their specialization and capabilities, is given by:

$$C_{server} = \sum_{k \in K} \sum_{n \in N} x^k_n sa_n.$$

Total Cost Function

The total cost is the sum of the above components:

$$C_{total} = C_{capex} + C_{bandwidth} + C_{server},$$

and the objective is to minimize C_{total}.

This leads to the following MINLP formulation:

$$\min C_{total} \quad \text{s.t. constraints (1)-(12)},$$

which is a specialized version of the integer multicommodity flow problem incorporating function placement, reliability, and heterogeneous node capacity constraints. Due to its NP-hardness, we subsequently reformulate the problem as a mixed-integer linear program (MILP).

B. Linearization of the Mathematical Model

In this section, we introduce a linearization technique inspired by [5] to reformulate the nonlinear terms in constraint (12). Define p^k_n, which is the probability that request $k \in \mathcal{R}$ successfully reaches switch $n \in \mathcal{N}$ from its source (assuming n is on the path). Constraint (12) is thus replaced by:

$$p^k_{o^k} = R_{o^k}, \quad \forall k \in \mathcal{R}, \tag{13a}$$

$$p^k_n \leq p^k_j R_n + \Big(1 - y^k_{j,n}\Big), \quad \forall k \in \mathcal{R},\ (j,n) \in \mathcal{L}, \tag{13b}$$

$$p^k_n \geq \tau_k, \quad \forall k \in \mathcal{R},\ n \in \mathcal{N}. \tag{13c}$$

When $y^k_{j,n} = 1$ (i.e., link (j,n) is used), (13b) ensures that the probability to reach n does not exceed $p^k_j R_n$. Otherwise, the constraint is redundant.

Since $R_n p^k_j$ is nonlinear, we exploit the fact that for each request (routed on a single path) the computational resource L_n and hence R_n can take only a finite number of values.

Two-Stage Approach:

- **First Stage:** A reverse depth-first search (DFS) from server n identifies sources sending flow through n, and a forward DFS from n checks destination reachability. Let K'_n be the set of requests where both conditions hold.
- **Second Stage:** We compute all possible values for L_n by considering sums of the form

  $$\sum_{k \in \mathcal{R}'} \sum_{f \in \mathcal{F}'} \eta^k_{n,f},$$

  for all subsets $\mathcal{R}' \subseteq K'_n$ and $\mathcal{F}' \subseteq F^k$. Using dynamic programming, define the binary variable

  $$w_n(q_1, q_2, s),$$

  which equals 1 if there exist subsets of the first q_1 requests and q_2 VNFs with sum s. The recursion is:

  $$w_n(q_1,q_2,s) = \max\Big\{ w_n(q_1-1,q_2,s),\ w_n(q_1,q_2-1,s), \\ w_n(q_1-1,q_2-1,s-\eta^{q_1}_{n,f_{q_2}}) \Big\}. \tag{14}$$

Feasible values for L_n are those s for which

$$w_n(|K'_n|, |F|, s) = 1.$$

At the end of stage two, define

$$\Gamma_n = \{\gamma_0, \ldots, \gamma_{G_n}\},$$

and via (11),

$$\Phi_n = \{\phi_0, \ldots, \phi_{G_n}\}.$$

The two sets Γ_n and Φ_n represent all possible values for L_n and R_n, respectively. To enforce these discrete values, introduce binary variables u_n^g for $g = 0, \ldots, G_n$ with $u_n^g = 1$ if $L_n = \gamma_g$ (thus, $R_n = \phi_g$). For ease of presentation, define the set $\mathcal{G} = \{0, \ldots, G_n\}$. Then, replace (9) and (11) with:

$$\sum_{g=0}^{G_n} \gamma_g \, u_n^g = \sum_{k \in \mathcal{R}} \sum_{f \in \mathcal{F}} \eta_{n,f}^k \, \rho_{fn}^k, \quad \forall n \in \mathcal{N}, \tag{15a}$$

$$R_n = \sum_{g=0}^{G_n} \phi_g \, u_n^g, \quad \forall n \in \mathcal{N}. \tag{15b}$$

Also, enforce:

$$\sum_{g=0}^{G_n} u_n^g \leq 1, \quad \forall n \in \mathcal{N}, \tag{16a}$$

$$u_n^g \in \{0, 1\}, \quad \forall n \in \mathcal{N}, \; g \in \mathcal{G}. \tag{16b}$$

After substitution into (13b), the product becomes $u_n^g p_j^k$. Since this is the product of a binary and a bounded continuous variable, introduce the continuous variable

$$w_{jn}^{gk} = p_j^k \, u_n^g,$$

enforced via:

$$w_{jn}^{gk} \leq u_n^g, \quad \forall k, \, (j,n) \in \mathcal{L}, \; g \in \mathcal{G}, \tag{17a}$$

$$w_{jn}^{gk} \leq y_{j,n}^k, \quad \forall k, \, (j,n) \in \mathcal{L}, \; g \in \mathcal{G}, \tag{17b}$$

$$w_{jn}^{gk} \leq p_j^k, \quad \forall k, \, (j,n) \in \mathcal{L}, \; g \in \mathcal{G}, \tag{17c}$$

$$w_{jn}^{gk} \geq p_j^k - (1 - u_n^g) - (1 - y_{j,n}^k), \quad \forall k, \, (j,n) \in \mathcal{L}, \; g \in \mathcal{G}. \tag{17d}$$

Let χ denote the set of (Y, X, ρ, U) satisfying constraints (1)–(6). Then, the MINLP is reformulated as the following MILP:

$$\min \; C_{total} = C_{capex} + C_{bandwidth} + C_{server} \tag{18a}$$

$$\text{s.t.} \quad (Y, X, \rho, U) \in \chi, \tag{18b}$$

$$\sum_{g=0}^{G_n} \gamma_g \, u_n^g = \sum_{k \in \mathcal{R}} \sum_{f \in \mathcal{F}} \eta_{n,f}^k \, \rho_{fn}^k, \quad \forall n \in \mathcal{N}, \tag{18c}$$

$$\sum_{g=0}^{G_n} \gamma_g \, u_n^g \leq sc_n, \quad \forall n \in \mathcal{N}, \tag{18d}$$

$$R_n = \sum_{g=0}^{G_n} \phi_g \, u_n^g, \quad \forall n \in \mathcal{N}, \tag{18e}$$

$$\sum_{g=0}^{G_n} u_n^g \leq 1, \quad \forall n \in \mathcal{N}, \tag{18f}$$

$$p_{o^k}^k = R_{o^k}, \quad \forall k \in \mathcal{R}, \tag{18g}$$

$$p_n^k \leq \sum_{g=0}^{G_n} \phi_g \, w_{jn}^{gk} + \left(1 - y_{j,n}^k\right), \quad \forall k \in \mathcal{R}, \; (j,n) \in \mathcal{L}, \tag{18h}$$

$$p_n^k \geq \tau_k, \quad \forall k \in \mathcal{R}, \; n \in \mathcal{N}, \tag{18i}$$

$$w_{jn}^{gk} \leq u_n^g, \forall k, \, (j,n) \in \mathcal{L}, g \in \mathcal{G}, \tag{18j}$$

$$w_{jn}^{gk} \leq y_{j,n}^k, \forall k, \, (j,n) \in \mathcal{L}, g \in \mathcal{G}, \tag{18k}$$

$$w_{jn}^{gk} \leq p_j^k, \forall k, \, (j,n) \in \mathcal{L}, g \in \mathcal{G}, \tag{18l}$$

$$w_{jn}^{gk} \geq p_j^k - (1 - u_n^g) - (1 - y_{j,n}^k), \quad \forall k, \, (j,n) \in \mathcal{L}, \; g \in \mathcal{G}, \tag{18m}$$

$$u_n^g \in \{0, 1\}, \quad \forall n \in \mathcal{N}, \; g \in \mathcal{G}. \tag{18n}$$

Note that the size of the MILP grows with G_n. For large values, solving the MILP may be challenging, and an alternative solution approach is proposed in the next section.

IV. Cutting Plane-Based Heuristic

As mentioned earlier, the proposed model is NP-hard and the exact formulation in Section III-B may be impractical for large networks. Hence, we use a cutting plane approach to obtain an optimal solution by first constructing lower and upper-bounding models that relax the challenging constraints (11) and (12).

In these models, we replace (12) with the equivalent constraints (13a), (13b), and (13c) and substitute estimated values for L_n and R_n as described next.

Lower and Upper-Bounding Models

For each server node n, we identify $\mu_n + 1$ possible sums of the form

$$\sum_{k \in K} \sum_{f \in F^k} \rho_{fn}^k,$$

which yield the potential values

$$\{d_0, \ldots, d_{\mu_n}\},$$

with $d_0 = 0$ and d_{μ_n} equal to the maximum possible load. Using the reliability function (11), we obtain the corresponding reliability values

$$\{H_0, \ldots, H_{\mu_n}\},$$

where $H_0 = 1$ and $H_g \geq H_{g+1}$.

For the lower-bound model, we round down L_n to the nearest value by introducing a binary variable $\tilde{u}_n^g$ that equals 1 if $L_n \geq d_g$ and 0 otherwise. This is enforced by the forcing constraint:

$$\tilde{u}_n^g \geq \frac{L_n - d_g + 1}{\sum_{k \in K} \sum_{f \in F^k} \rho_{fn}^k - d_g + 1}, \quad \forall n \in \mathcal{N}, \; g = 1, \ldots, \mu_n. \tag{19}$$

Then, the estimated reliability is given by

$$R_n^L = 1 + \sum_{g=1}^{\mu_n} \tilde{u}_n^g \Big(H_g - H_{g-1} \Big). \tag{20}$$

The lower bounding formulation is obtained by (i) replacing R_n in the MINLP model with R_n^L, (ii) adding constraint (19), and (iii) replacing (11) by (20). Because the load is rounded down, this yields an overestimated reliability, so every solution feasible for the original model is also feasible for the lower-bound model.

For the upper-bound model, we define $\tilde{u}_n^g$ to equal 1 if $L_n > d_g$ (and 0 otherwise). The forcing constraint becomes

$$\tilde{u}_n^g \geq \frac{L_n - d_g}{\sum_{k \in K} \sum_{f \in F^k} \rho_{fn}^k - d_g}, \quad \forall n \in \mathcal{N},\ g = 1, \ldots, \mu_n, \tag{21}$$

and the estimated reliability is

$$R_n^U = 1 + \sum_{g=0}^{\mu_n - 1} \tilde{u}_n^g \Big(H_{g+1} - H_g \Big). \tag{22}$$

The upper bounding model is then obtained by substituting R_n with R_n^U and adding (21).

Both models are then linearized using the techniques described in Section III-B.

Cutting Planes

We next solve the lower-bound model to obtain an integer-feasible solution

$$(\overline{Y}, \overline{X}, \overline{\rho}, \overline{U}).$$

An algorithmic implementation of our method is provided in Algorithm 1. To this end, we define two parameters, LB and UB, representing the lower and upper bounds of the objective value, respectively. The initial value of LB, denoted by LB', is obtained from the objective function value of the linear relaxation of the MILP presented in Section (III-B). For UB, the initial value, denoted by UB', is derived considering that the variables $y_{i,j}^k$ and x_n^k are binary and ρ_{fn}^k is an integer variable bounded by m_f^k. After computing the actual values of L_n and R_n (via (9) and (11)), we check each end-to-end request k by computing

$$\prod_{n \in \mathcal{N}} R_n^{x_n^k},$$

and if this product is below τ_k, we mark k as violated (i.e., $k \in \bar{B}$).

For each violated request, we define:

- $\tilde{L}^k = \{(i,j) \in \mathcal{L} \mid \overline{y_{i,j}^k} = 1\}$,
- $P^k = \{n \mid \overline{x_n^k} > 0\}$,
- $\bar{F}^k = \{(n, f, k') \mid n \in P^k,\ f \in F,\ \overline{\rho_{fn}^{k'}} = 1\}$.

A valid cut is then added to force a change:

$$\sum_{(n,f,k') \in \bar{F}^k} (1 - \rho_{fn}^{k'}) + \sum_{(i,j) \in \mathcal{L}} (1 - y_{i,j}^k) \geq 1, \quad \forall k \in \bar{B}. \tag{23}$$

This inequality eliminates the current (infeasible) solution from the lower-bound feasible region.

Algorithm 1 Cutting Plane Algorithm-Based Heuristic for the Proposed Problem

Input: All parameters, $\bar{K} = \emptyset$, tolerance ϵ, initial bounds $LB = LB'$ and $UB = UB'$.
Output: Optimal reliable design.
Formulate lower and upper bounding models using subsets $\{d_0, \ldots, d_{\mu_n}\}$ and $\{H_0, \ldots, H_{\mu_n}\}$.
Add initial cuts (24)–(25) to both models.
while $UB - LB \geq \epsilon$ **do**
  for each $n \in \mathcal{N}$ **do**
    Compute $\overline{L_n}$ and $\overline{R_n}$ via Eqs. (9) and (11).
  end for
  Set $\bar{B} = \emptyset$.
  for each $k \in K$ **do**
    if $\prod_{n \in \mathcal{N}} R_n^{x_n^k} < \tau_k$ **then**
      $\bar{B} \leftarrow \bar{B} \cup \{k\}$.
    end if
  end for
  if $\bar{B} = \emptyset$ **then**
    Return feasible solution $(\overline{x_n^k}, \overline{y_{i,j}^k}, \overline{\rho_{fn}^{k'}})$ with current bounds LB, UB.
  else
    for each $k \in \bar{B}$ **do**
      Add cut (23) to both models.
    end for
  end if
end while=0

Warm-Start Strategy

To improve convergence, we generate an initial set of tight cuts. For example:

- **Connectivity Constraints:** Enforced at any intermediate node n (i.e., $n \notin \{o^k, t^k\}$):

$$y_{n,j}^k \leq \sum_{h:\,(h,n) \in \mathcal{L}} y_{h,n}^k, \tag{24}$$

$$y_{j,n}^k \leq \sum_{h:\,(n,h) \in \mathcal{L}} y_{n,h}^k. \tag{25}$$

- **Cover Constraints:** Guarantee that the total capacity leaving the source and entering the destination meets the bandwidth requirement:

$$\sum_{j:\,(o^k,j) \in \mathcal{L}} lc_{o^k,j}\, y_{o^k,j}^k \geq b^k, \tag{26}$$

$$\sum_{j:\,(j,t^k) \in \mathcal{L}} lc_{j,t^k}\, y_{j,t^k}^k \geq b^k. \tag{27}$$

The initial lower bound LB' is derived from the linear relaxation of the MILP in Section III-B, while an initial upper bound UB' is computed as

$$C_{total} \leq \sum_{k \in K} \sum_{f \in F} \sum_{n \in \mathcal{N}} m_f^k\, \alpha_{n,f} + \sum_{k \in K} \sum_{(i,j) \in \mathcal{L}} la_{i,j} + \sum_{k \in K} \sum_{n \in \mathcal{N}} sa_n. \tag{28}$$

V. Numerical Results

In this section, we evaluate the performance of the proposed optimization model. The network topology, links capacities and associated traffic demands (i.e., source-destination pairs and flow amounts) are extracted from the SNDLib database [6], a key resource in network design research. In the following, we describe the experimental setup, the service reliability model, and the performance metrics used for evaluation.

1) Experiments Design: From the options available, we select from the SNDLib database the *Abilene* topology, which consists of $N = 12$ nodes and 15 bidirectional links. We conduct several experiments investigating the interplay between two critical aspects of VNF placement: i) the variation in the service demand (i.e., the number $|K|$ of service requests in the network) and ii) the variation in the service requests reliability threshold τ_k (i.e., the tightness of constraint (12)). We let $|K|$ take values in $\{2, 4, 6, 8\}$, while τ_k is set to the same value for all requests ranging from 0 to 1 in increments of 0.1. We leave the case of heterogeneous reliability requirements for future work. We consider three types of network nodes, characterized by different computational and storage capabilities [1], namely: off-the-shelf server nodes, smart NICs and PISA switches. Without loss of generality, we assume that each VNF instance is able to process the same amount of input traffic (i.e., θ_f) and requires the same computational resources (i.e., $\eta^k_{n,f}$), leaving to future work the analysis of scenarios with differentiated VNFs options. Table II summarizes the values of i) sb_n and sc_n for each considered node type, ii) θ_f and $\eta^k_{n,f}$ for VNF instantiation, and iii) activation costs for nodes, links, and VNFs (i.e., sa_n, $la_{i,j}$ and $\alpha_{n,f}$).

2) Reliability Model: The reliability of each server $n \in N$ is modeled as a non-increasing, concave, and continuous function of its computational load L_n. Specifically, we define:

$$R_n = 1 - \left(\frac{L_n}{sc_n}\right)^2, \quad \forall n \in N, \tag{29}$$

with $L_n \leq sc_n$ to ensure that $R_n \geq 0$. This formulation captures the gradual degradation of server performance as computational load increases, with an accelerated decline as L_n approaches sc_n.

3) Performance Metrics: To assess the efficiency of our model in allocating VNF instances, we analyze the utilization of server computational resources. For each server $n \in N$, we define the resource utilization ξ_n as:

$$\xi_n = \frac{L_n}{sc_n}, \tag{30}$$

which represents the fraction of the computational capacity consumed at server n. To characterize the distribution of ξ_n across the network, we compute the Coefficient of Variation (CV) and the maximum utilization value ξ^{max} as follows:

$$\text{CV} = \frac{\sigma_{\xi_n}}{\sum_n \xi_n / N}, \tag{31}$$

TABLE II: Summary of Model Parameter Values

Nodes	
sb_n [Gbps]	Server: 40; Smart NIC: 80; PISA Switch: 160
sc_n [# CPU]	Server: 8; Smart NIC: 12; PISA Switch: 16
Links	
$la_{i,j}$	$\mathcal{U}([100, 1200])$
$lc_{i,j}$	As from [7]
VNFs	
$\alpha_{n,f}$ [$]	$\mathcal{U}([50, 1000])$ [8]
$\eta^k_{n,f}$	2 CPU
θ_f [Mbps]	900

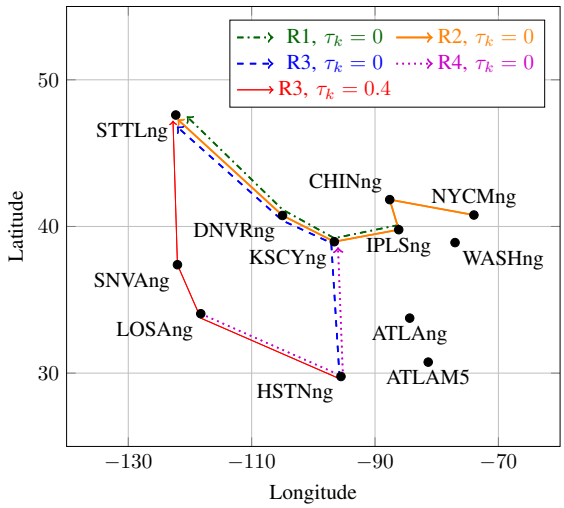

Fig. 1: Example network scenario with service requests routing paths for $|K| = 4$. The red line represents how request R3 is re-routed after increasing τ_k from 0 to 0.4.

$$\xi^{max} = \max_{n \in N} (\xi_n), \tag{32}$$

where σ_{ξ_n} is the standard deviation of ξ_n. In the next section, we discuss how the proposed model improves performance by (i) reducing the CV, indicating a more balanced resource allocation, and (ii) lowering ξ^{max}, which mitigates potential congestion bottlenecks.

A. Impact on Server Resource Utilization

Figure 1 illustrates a solution for a representative scenario with $|K| = 4$ and $\tau_k = 0$. The figure also demonstrates how stricter reliability requirements affect routing. Specifically, service request R3 is shown as a blue, dashed line from node *HSTNng* to node *STTLng* when τ_k = 0. As τ_k increases to 0.4, the optimization model re-routes R3—now depicted in red—to satisfy the higher quality-of-service requirement.

Figure 2 shows the performance results obtained from our optimization model. The figure plots the CV (green, solid lines) and ξ^{max} (blue, dashed lines) as functions of τ_k. The

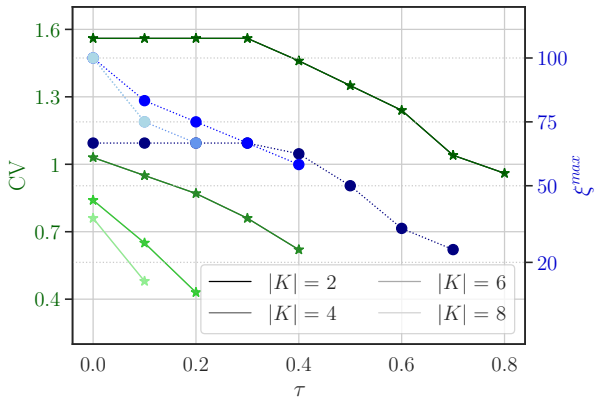

Fig. 2: Left y-axis: CV of server utilization (green, solid lines). Right y-axis: ξ^{max} (blue, dashed lines). Darker colors indicate scenarios with fewer service requests ($|K|$ is lower). Note that for ξ^{max}, data points corresponding to different values of $|K|$ overlap when $\tau_k \leq 0.4$.

line tones indicate the number of service requests in the network, with darker lines representing lower service loads. We observe that, for a given set of initial parameters, the model becomes infeasible beyond certain values of τ_k. For example, when $|K| = 2$, solutions exist up to $\tau_k = 0.8$; in contrast, when $|K| = 8$, no solution is available for $\tau_k \geq 0.2$. Lower CV values indicate a more uniform workload distribution, while higher CV values suggest disparities in server utilization. Regardless of the number of requests, CV consistently decreases as τ_k increases, demonstrating the model's ability to distribute the load more evenly when higher reliability is required. In particular, compared to the baseline case of $\tau_k = 0$ (i.e., when no requirement is set on reliability), we observe an improvement in workload distribution between 37% ($|K| = 8$, $\tau_k = 0.1$) and 49% ($|K| = 6$, $\tau_k = 0.2$) at the highest feasible reliability thresholds. Reducing ξ^{max} is equally important, as it helps manage network resources and reduces the risk of congestion. Similar to CV, ξ^{max} shows a clear decreasing trend with increasing τ_k. Our results indicate that the proposed optimization framework can significantly lower maximum server utilization—from 25% in the most heavily loaded scenario ($|K| = 8$, τ_k = 0.1) to 63% in the least loaded scenario ($|K| = 2$, τ_k = 0.8). In our final experiment, we increased the network link capacities ($2 \times lc_{i,j}$) and the computational resource capacities ($4 \times sc_n$) to examine the impact on server utilization ξ_n when the number of service requests $|K|$ exceeds 8. Table III presents the results for $|K| = 10$ and $|K| = 15$. As observed, both CV and ξ^{max} exhibit a decreasing trend with respect to τ_k, consistent with the trends observed in previous experiments.

B. Cost Components and Running Time

Table IV summarizes the cost components (server and link activation costs, CAPEX and total cost) and the corresponding computation times for the proposed optimization model. With respect to costs, regardless of the number of service requests, increasing τ_k leads to non-decreasing CAPEX. This is due to the potential deployment of additional VNF instances to meet the reliability requirements. Moreover, the model tends to prefer deploying new VNF instances on already activated server nodes rather than re-routing a service request. Consequently, server costs typically remain stable while CAPEX increases, unless new nodes must be activated to satisfy τ_k (as observed for request R3 in Figure 1). In such cases, server costs may also increase. In some instances, the algorithm can reduce overall link costs by selecting a cheaper connection to a new server, even though the total cost always increases with higher τ_k. This behavior aligns with the model's strategy of balancing the weights of the three cost components.

TABLE III: Servers' utilization statistics with increased link ($2 \times lc_{i,j}$) and computational capacities ($4 \times sc_n$).

$\|K\|$	τ_k	CV	ξ^{max} [%]
10	0.4	1.21	65.6
	0.5	1.12	62.5
	0.6	1.03	53.1
	0.7	0.88	40.6
15	0	0.92	100
	0.1	0.73	79.2
	0.2	0.64	75.0

Regarding computation times, the experiments were conducted on a commodity server equipped with an Intel i5-5200U and 16 GB of memory. As reported in the Table, the optimization program typically requires less than 7 minutes to solve, irrespective of the number of service requests or the reliability requirement. However, increasing both $|K|$ and τ_k generally results in higher computation times. For instance, when $|K| = 8$, setting $\tau_k = 0.1$ requires roughly 400% more solving time than the case with no reliability requirement.

VI. Related Works

Many studies in the literature have investigated efficient solutions for the placement of VNFs, either leveraging the renewed interest in deep learning approaches [9], [10] or relying on heuristics and approximation techniques. Regarding the latter, the placement of virtual middle-boxes is often optimized with respect to CAPEX and OPEX [11], the balancing of server nodes' workload [12], [13], or the routing of service demands [8], [14], [15]. In [11] the authors consider a hybrid software defined networking environment and formulate an ILP problem to jointly optimize VNF placement while minimizing system costs. They then apply various approximation strategies based on either decomposition theory or hidden Markov models, to evaluate the trade-off between CAPEX and OPEX minimization. Differently, [12] examines the effects of dynamic traffic patterns on VNF placement and proposes an effective load balancing solution tailored to large size, long-duration elephant flows. Load balancing is also the focus of [13], where the authors analyze data center environments to demonstrate the optimality of Join-the-Idle-Queue, a balancing scheme able to reduce the average response time of servers

TABLE IV: Impact of τ_k on cost components and model runtime ($\mathcal{T}$). Costs are scaled by 10^3.

$\lvert\mathbf{K}\rvert$	τ_k	**Server**	**Link**	**CAPEX**	**Total**	$\mathcal{T}$ [s]
2	0.0	39.5	7.5	2.0	49.0	1.73
	0.1	39.5	7.5	2.0	49.0	1.15
	0.2	39.5	7.5	2.0	49.0	0.94
	0.3	39.5	7.5	2.0	49.0	0.64
	0.4	39.5	7.5	2.0	49.0	1.60
	0.5	39.5	7.5	2.1	49.1	1.12
	0.6	39.5	7.5	2.3	49.3	4.66
	0.7	39.5	7.5	2.7	49.7	3.03
	0.8	43.5	7.6	3.9	55.0	11.32
4	0.0	67.1	14.0	5.0	86.1	0.44
	0.1	67.1	14.0	5.4	86.5	1.73
	0.2	67.1	14.0	5.8	86.9	1.67
	0.3	67.1	14.0	6.7	87.8	73.93
	0.4	68.5	13.5	8.0	90.0	72.77
6	0.0	103.8	21.9	7.4	133.1	0.90
	0.1	103.8	21.9	8.9	134.6	25.84
	0.2	105.2	21.4	10.1	136.7	320.80
8	0.0	130.5	26.4	9.3	166.2	1.46
	0.1	132.0	25.9	11.6	169.5	400.56

jobs. Considering path-based formulations, [8] addresses VNF placement and routing by formalizing a MILP optimization problem that finds a routing path for each service demand and optimally associates functions to nodes while minimizing functions installation and node activation costs. The proposed formulation is evaluated on a set of realistic telecommunication instances, and is shown to outperform earlier state-of-the-art compact MILP formulation. Finally, cost-aware multi-path routing and candidate path selection in heterogeneous service chaining scenarios is also the focus of [15], where authors unify NFV and computational offloading to optimize resource management. Results on simplified network topologies demonstrate the algorithm's ability to minimize system costs while achieving a high quality of service.

VII. Conclusion

This paper addressed the complex challenge of optimizing Network Functions Virtualization deployments, focusing on efficient Virtual Network Function placement and resource allocation. By leveraging advanced optimization techniques, including novel linearization strategies and a cutting-plane heuristic with warm-start strategies, we transformed a non-convex MINLP formulation into a tractable MILP model, thereby effectively mitigating the inherent computational complexities of NFV systems.

Extensive experimentation on a real-world network topology (i.e., the Abilene network) validated the effectiveness of our approach. In particular, our framework achieved up to a 49% improvement in workload distribution—as measured by a reduction in the Coefficient of Variation—and reduced the maximum server utilization by as much as 63%. These results show the model's ability to balance resource usage and alleviate congestion bottlenecks while maintaining cost efficiency. Moreover, our optimization model demonstrated scalability by solving realistic instances in under 7 minutes on a commodity server, even as the number of service requests and reliability constraints increased.

These promising findings demonstrate our approach's potential to deliver scalable, cost-effective, and reliable NFV deployments. Future work will extend the framework for dynamic traffic patterns and evolving service requirements, integrate machine learning to predict congestion and resource demands, and explore decentralized or distributed optimization for enhanced scalability.

Acknowledgments

This work is supported by the University of Montpellier's MAK'IT (Montpellier Advanced Knowledge Institute on Transitions) visiting scientist program, by projects *SERICS* (PE00000014) under the NRRP MUR program, funded by the EU - NGEU, and PRIN *NEWTON* (2022ZA8T22).

References

[1] J. G. 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.

[2] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu, "Recent advances of resource allocation in network function virtualization," IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 2, pp. 295–314, 2020.

[3] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, "VNF and CNF Placement in 5G: Recent Advances and Future Trends," IEEE Trans. on Netw. and Service Management, vol. 20, no. 4, pp. 4698–4733, 2023.

[4] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," Surveys in Operations Research and Management Science, vol. 17, pp. 97–106, 2012.

[5] A. K. Andreas and J. C. Smith, "Mathematical programming algorithms for two-path routing problems with reliability considerations," INFORMS Journal on Computing, vol. 20, no. 4, pp. 553–564, 2008.

[6] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0-Survivable network design library," Networks: An International Journal, vol. 55, no. 3, pp. 276–286, 2010.

[7] Y. Zhang, "6 months of abilene traffic matrices." [Online]. Available: https://www.cs.utexas.edu/~yzhang/research/AbileneTM/

[8] A. Mouaci, É. Gourdin, I. LjubiĆ, and N. Perrot, "Virtual network functions placement and routing problem: Path formulation," in IFIP Networking Conference (Networking), 2020, pp. 55–63.

[9] Y. Bi, C. C. Meixner, M. Bunyakitanon, X. Vasilakos, R. Nejabati, and D. Simeonidou, "Multi-Objective Deep Reinforcement Learning Assisted Service Function Chains Placement," IEEE Transactions on Network and Service Management, vol. 18, no. 4, pp. 4134–4150, 2021.

[10] N. He, S. Yang, F. Li, S. Trajanovski, L. Zhu, Y. Wang, and X. Fu, "Leveraging Deep Reinforcement Learning With Attention Mechanism for Virtual Network Function Placement and Routing," IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 4, pp. 1186–1201, 2023.

[11] C. Ren, H. Li, Y. Li, Y. Wang, H. Xiang, and X. Chen, "On efficient service function chaining in hybrid software defined networks," IEEE Transactions on Network and Service Management, vol. 19, no. 2, pp. 1614–1628, 2021.

[12] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in IEEE Conference on Computer Communications (INFOCOM), 2017, pp. 1–9.

[13] S. Bhambay and A. Mukhopadhyay, "Optimal load balancing in heterogeneous server systems," in 20th IEEE International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt), 2022, pp. 113–120.

[14] M. Gao, B. Addis, M. Bouet, and S. Secci, "Optimal orchestration of virtual network functions," Comp. Netw., vol. 142, pp. 108–127, 2018.

[15] Y. Kim, H.-W. Lee, and S. Chong, "Control of multi-resource infrastructures: Application to NFV and computation offloading," in 16th IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2018, pp. 1–8.