ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 11, No. 3, 9-39 (2026)
www.astesj.com
Special Issue on Multidisciplinary Frontiers in Engineering, Computing and Applied Sciences 2026

ASTES Journal
ISSN: 2415-6698

Many-objective Placement Optimization in Virtual Network Functions

Cristhian Jesus Benitez Dominguez, Fernando Ariel Britez Cantero, Luis Guillermo Moré Rodríguez*, Diego Pedro Pinto-Roa, José Domingo Colbes Sanabria

Facultad Politécnica, Universidad Nacional de Asunción, San Lorenzo, Paraguay

Email(s): cristhian.benitez5@fpuna.edu.py (C.J. Benitez Dominguez), Fbritezcantero@gmail.com (F.A. Britez Cantero), lmore@pol.una.py (L.G. Moré Rodríguez), dpinto@pol.una.py (D.P. Pinto-Roa), jcolbes@pol.una.py (J.D. Colbes Sanabria)

*Corresponding Author: Luis Guillermo Moré Rodríguez, Facultad Politécnica, Universidad Nacional de Asunción, San Lorenzo, Paraguay, Email: lmore@pol.una.py

ARTICLE INFO

Article history:
Received: 29 April, 2026
Revised: 05 June, 2026
Accepted: 10 June, 2026
Online: 24 June, 2026

Keywords:
NFV
VNF placement
MaOP
R-VEA
NSGA-III
MOEA/D
QoS
CAPEX
OPEX

ABSTRACT

Network Functions Virtualization (NFV) poses the VNF placement problem under multiple, potentially conflicting objectives, such as Quality of Service (QoS), costs, and resource efficiency. This work treats VNF placement *as a many-objective optimization problem (MaOP) and presents two primary contributions: (i) a correlation analysis of state-of-the-art objectives to reduce dimensionality while maintaining representativeness, resulting in the selection of 11 key objective functions — specifically, energy cost, bandwidth consumption, latency, traffic load, resource fragmentation, maximum link utilization, licensing costs, SLO costs, distance traveled, number of VNF instances, and effective* throughput*; and (ii) a systematic comparison of many-objective evolutionary algorithms (R-VEA, NSGA-III, and MOEA/D) across multiple topologies (ZIB54, INDIA, and EON) under various load levels. Results demonstrate through non-parametric Friedman and Wilcoxon statistical testing (adjusted* $p < 0.05$ *via Holm-Bonferroni) that R-VEA achieves superior overall performance. Its Angle-Penalized Distance (APD) metric effectively balances convergence and diversity across most scenarios, producing a Vargha-Delaney effect size (*A_{12}*) greater than 0.86 in most topologies compared to NSGA-III and MOEA/D. Conversely, MOEA/D exhibits competitive performance only in specific high-traffic edge cases (e.g., ZIB54 under extreme load), where it outpaces R-VEA due to front distortion. Taken together, the findings indicate that while no single algorithm dominates in every outlier condition, R-VEA emerges as the most statistically consistent and robust solution for the 11-objective MaOP* VNF placement *problem. This contributes to ensuring QoS and reducing CAPEX/OPEX through high-performance solutions along the Pareto front.*

1. Introduction

Driven by the rapid growth of telecommunications in recent years, there is a critical need of network architectures that offer greater flexibility and lower costs while eliminating dependency on proprietary hardware vendors. Consequently, several studies propose decoupling network functions from hardware [1], referring to traditional implementations as *Physical Network Functions* (PNFs) or hardware middleboxes. These consist of dedicated physical appliances designed to perform specific network tasks, such as transforming, inspecting, filtering or otherwise manipulating network traffic. Through these operations, various network objectives are achieved without compromising the Quality of Service (QoS) [2]. Common examples of such functions include firewalls, Deep Packet Inspection (DPI), Network Address Translation (NAT), WAN Optimization Controllers, Proxies, and Load Balancers [3] among others. Members of the *European Telecommunications Standards Institute* (ETSI) proposed virtualization to enable greater versatility in network functions, addressing the previously described challenges [4]. This gave origin to the term *Network Function Virtualization* (NFV). The main idea behind this is to build a network architecture that transforms the way to build and operate networks taking advantage of the standard IT virtualization technologies to consolidate network functions based on proprietary hardware of commercial devices [4].

Fundamentally, this technology entails replacing expensive ded-

icated hardware by software middleboxes, which are referred to as *Virtual Network Functions* (VNFs) in NFV terminology [5]. The adoption of this emergent technology results in significant reductions in *Capital Expenditure* (CapEx) and *Operating Expenditure* (OpEx), while simultaneously enhancing Quality of Service (QoS) for final users.

Several critical challenges exist within the NFV paradigm, most notably the *Virtual Network Function Placement* (VNF-P) problem, which is a central issue characterized by a NP-Hard complexity [6]. Various research efforts have addressed VNF-P as a *Multi-Objective Optimization Problem* (MOOP) [7]. Specifically, literature reports the use of bi-objective [8, 9] and four-objective heuristics [10] such as the *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) [11]. These techniques, classified as *Multi-Objective Evolutionary Algorithms* (MOEAs), are generally limited to maintaining high performance with a maximum of three objective functions [12]. Nevertheless, literature has reported an extensive group of objective functions and constraints, due to the inherent complexity and diverse aspects addressed in this field of study. Given that multi-objective approaches face scalability limitations regarding the number of objective functions, it is essential to conduct studies and propose strategies that incorporate a wide range of objective functions documented in the literature.

While several studies address VNF-P with two or three objectives, they fail to provide a holistic view that simultaneously balances Quality of Service (QoS), Capital Expenditures (CAPEX), and Operational Expenditures (OPEX). Real-world NFV deployments require optimizing significantly more than three conflicting objectives concurrently. However, traditional MOEAs experience a severe degradation in search capability (known as dominance resistance) when the number of objectives increases. This creates a critical research gap: the lack of scalable, many-objective algorithmic approaches capable of resolving the highly constrained, multidimensional VNF-P problem without discarding essential operational metrics.

In contrast to previous studies in the state of the art, this work addresses the VNF-P problem by optimizing criteria categorized under OPEX, CAPEX, and *Quality of Service* (QoS). This is formulated as a *Many-objective Optimization Problem* (MaOP) [13] utilizing competitive *Many-objective Evolutionary Algorithms* (MaOEAs) [12]. Consequently, the primary contributions of this work are as follows:

- A comprehensive review and analysis of the VNF-P problem design criteria reported in the literature.
- Selection of the most appropriate criteria for the optimization process within a many-objective framework.
- Implementation of evolutionary techniques to solve the VNF-P problem as a many objective optimization task.
- A systematic performance comparison of these evolutionary techniques using relevant metrics for many-objective optimization.

The remainder of this study is organized as follows:

- Section 2 provides a detailed analysis of the NFV architecture.
- Section 3 focuses on reviewing and analyzing the existing literature related to the VNF-P problem.
- Section 4 formalizes and defines the VNF-P problem.
- Section 5 addresses the challenges and specific characteristics of multi objective and many objective optimization problems.
- Section 6 describes in detail the VNF placement problem solving utilizing evolutionary algorithms.
- Section 7 describes the metrics utilized to evaluate the performance of the proposed approaches.
- Section 8 presents and discusses the experimental results.
- Section 9 summarizes the significant contributions, provides general conclusions, and outlines future research directions.

For clarity, Table 1 presents the acronyms employed throughout this work.

Table 1: Acronyms utilized for this work.

Acronym	Description
CAPEX	Capital Expenses
COTS	Commercial Off-The-Shelf
DAS	Direct Attached Storage
DPI	Deep Packet Inspection
EM	Element Manager
ETSI	European Telecommunications Standards Institute
MANO	Management and orchestration
MOP	Multi-objective Optimization Problem
MOEA	Multi-Objective Evolutionary Algorithm
MOEAD	Multi-Objective Evolutionary Algorithm based on Decomposition
NAS	Network Attached Storage
NAT	Network Address Translation
NFVI	NFV Infrastructure
NFV	Network Functions Virtualization
NFVO	NFV Orchestrator
NSGA III	Non-dominated Sorting Genetic Algorithm III
Opex	Operating Expenses
PNF	Physical Network Function
QoS	Quality of Service
RVEA	Random Vector Evolutionary Algorithm
SAN	Storage Area Network
SDN	Software-Defined Networking
SFC	Service Function Chaining
SLO	Service Level Objective
SLA	Service Level Agreement
QOS	Quality of service
VIM	Virtualized Infrastructure Manager
VNF	Virtual Network Function
VNFM-	Virtual Network Function Manager
VNF-P	Virtual Network Function Placement
VM	Virtual Machine
WAN	Wide Area Network

2. NFV architectural foundations.

The NFV architecture [4] is shown in Fig. 1. It abstracts network functions from the underlying hardware by dividing the architecture into three primary functional layers: the Network Functions Virtualization Infrastructure (NFVI), Management and Orchestration (MANO), and the VNF layer. These layers separate the data, control, and application planes, respectively.

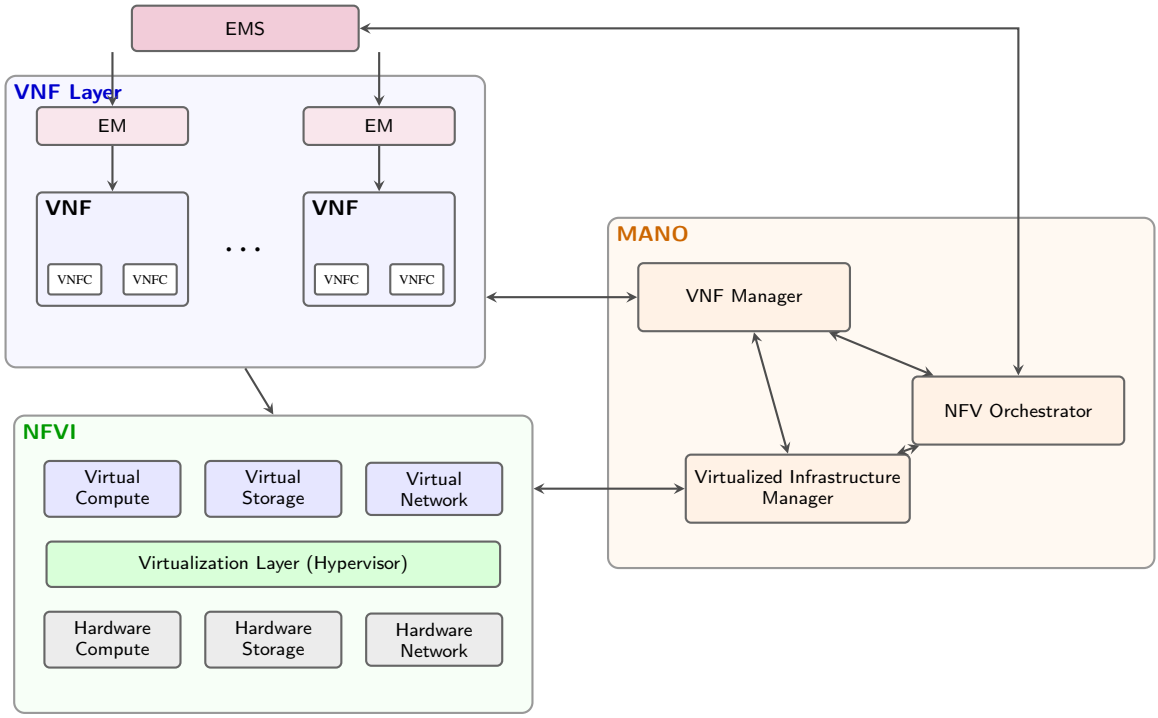


Figure 1: Standard ETSI NFV reference architecture illustrating the functional layers and management orchestration blocks.

2.1. Network Functions Virtualization Infrastructure (NFVI)

The NFVI corresponds to the data plane, providing the necessary resources to execute network services. It uses general-purpose hardware (COTS) to create a virtualization environment. The NFVI is subdivided into three components:

1. **Physical Infrastructure Layer:** Comprises general-purpose servers that provide computing (multi-core CPUs), storage (SSDs/HDDs), and networking (L2/L3 switches, NICs) capabilities [14].

2. **Virtualization Layer:** Positioned between the physical and virtual infrastructure, this software layer uses hypervisors (e.g., KVM, VMware ESXi) to allocate physical resources into isolated environments, like Virtual Machines (VMs). It manages dynamic mapping to ensure high portability across VMs [6, 15].

3. **Virtual Infrastructure Layer:** Includes virtual compute (VMs provisioned by hypervisor APIs), virtual storage (virtualized SAN/NAS), and virtual networks (interconnecting VMs via virtual switches like Open vSwitch) [6].

Despite ETSI's framework, providers often customize NFVI implementations to suit their network constraints and geographic distributions.

2.2. Management and Orchestration (MANO)

MANO acts as the control plane, managing the virtualized context within the NFV framework, including hardware orchestration and VNF lifecycles. It comprises three blocks:

- *VNF Orchestrator (NFVO):* Orchestrates NFVI resources and manages the lifecycle of multiple chained VNFs, calculating optimal service paths.
- *VNF Manager (VNF-M):* Manages specific VNF instances throughout their lifecycle (instantiation, scaling, termination).
- *Virtualized Infrastructure Manager (VIM):* Controls the NFVI resources (compute, storage, network) and establishes connectivity between network endpoints [4].

2.3. VNF Layer

The VNF layer corresponds to the application plane. It aims to implement Physical Network Functions (PNFs)—like firewalls or DHCP servers—as software-based VNFs running on COTS hardware. Each VNF is composed of multiple VNF Components (VNFCs) managed by an Element Management System (EMS). The EMS collaborates directly with the VNF-M to execute lifecycle management effectively. Chaining multiple VNFs across different locations forms a complete Service Function Chain (SFC) capable of replacing traditional, proprietary hardware appliances.

3. Related Work

The VNF Placement (VNF-P) problem is a critical research area aimed at reducing CapEx and OpEx while ensuring the QoS of network traffic. Over the years, multiple optimization approaches have been proposed to tackle this problem under different network conditions.

Existing works addressing the VNF-P problem can be broadly classified based on the number of objectives optimized and the

resolution strategy employed. As shown in Table 3, approaches are divided into Single-objective, Multi-objective (two or three objectives), and Many-objective (four or more objectives). Furthermore, the optimization strategies can be categorized into Scalarization methods (such as Weighted Sum and Lexicographic order) and Pareto-based approaches.

While Single-objective and Multi-objective approaches have been extensively studied, Many-objective optimization (MaOO) applied to VNF-P remains largely unexplored, particularly when combined with Pareto-based dominance. Scalarization methods often require prior knowledge of the objective weights, which is not always feasible in dynamic environments. In contrast, Pareto-based approaches provide a set of trade-off solutions, empowering network operators to make informed decisions. As highlighted in Table 3, there is a clear gap in the literature regarding Many-objective Pareto-based solutions for VNF-P, which this work aims to address.

Table 2 details the most common objective functions found in the literature.

Table 2: Objective functions proposed in the literature for static VNF-P

Function	Definition
f_1	Energy cost (USD)
f_2	Bandwidth consumption (Mbps)
f_3	Latency (ms)
f_4	Traffic charge (kb)
f_5	Installation costs (USD)
f_6	Total cost of resources (USD)
f_7	Resource fragmentation (USD)
f_8	Utilization cost of all network links (USD)
f_9	Maximum network link utilization (Mbps)
f_{10}	Licensing costs (USD)
f_{11}	SLO costs (USD)
f_{12}	Traveled distance (km)
f_{13}	Number of hops
f_{14}	Number of servers
f_{15}	Number of VNF instances
f_{16}	Effective network throughput (%)
f_{17}	Percentage of allocated VNFs (%)

Table 3: Taxonomy of Optimization Approaches in VNF-P

Objectives	**Scalarization (Weighted Sum / Lexicographic)**	**Pareto-based**
Single-objective (1)	[16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29]	N/A
Multi-objective (2-3)	[6], [30], [31], [5], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [9], [8]	[42], [2], [43], [44], [10]
Many-objective (≥ 4)	None	**Our Work**

4. VNF-P Problem Statement

4.1. Mathematical Symbols in VNF-P

This section presents the mathematical notation formulated for the VNF-P problem, as summarized in Table 4. A NFV environment comprises an underlying physical structure and various virtual components. The environment is organized through a network topology $G = (N, E)$, where N and E represent the set of physical node set and links, respectively. Each link $e \in E$ is associated to a bandwidth capacity B_e, a physical length L_e and a transmission delay Y_e.

To simplify the mathematical model we consider associated usage costs for every VNF $v \in V$, among we can cite:

- Licensing cost λC_v,
- Deploying cost δC,
- Processing delay Ψ_v,
- Resource requirements M_{rv}.

In the VNF-P problem, a set of requests is provided, where each request $t \in T$ is composed of a service function chain W_t, a source node $o_t \in N$, a destination node $\omega_t \in N$, the requested traffic bandwidth β_t, the maximum allowed delay Δ_t, cost of failing to meet the maximum delay requirement ρC_t, i.e., $t = (o_t, \omega_t, \beta_t, \Delta_t, \rho C_t, W_t)$.

4.2. Basic example of a VNF-P solution

Consider a simple network topology for this practical example, as shown in Figure 2. The figure illustrates the placement of a service chain corresponding to two requests. The first request is depicted in green, with source node n_1, destination node n_5, and a service chain comprising a Proxy deployed at node n_7, an IDS at n_7, and a Firewall at n_8. Conversely, the second request, depicted in red, originates at node n_1, terminates at node n_8, and has a service chain comprising a DPI at node n_4, an IDS at node n_3, and a Firewall at node n_8.

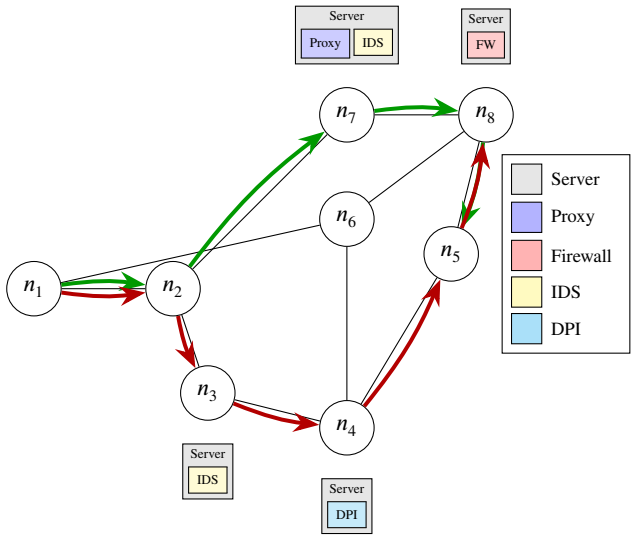


Figure 2: VNF allocation within a network.

Table 4: Comprehensive Summary of Mathematical Notation in VNF-P

Symbol	Description (Physical & Virtual Network)	Symbol	Description (Traffic Requests & Multi-Stage Graph)
Physical Substrate Network		**Traffic Request & Service Chains**	
$G = (N, E)$	Physical network topology (nodes N, links E)	T	Set of traffic placement requests
$n \in N$	Physical node in the network	$t \in T$	A single traffic request tuple $(o_t, \omega_t, \beta_t, \Delta_t, \rho C_t, W_t)$
$e \in E$	Physical link connecting nodes n_i, n_j	$o_t \in N$	Source node of traffic request t
B_e	Bandwidth capacity of link e	$\omega_t \in N$	Destination node of traffic request t
L_e	Physical length of link e	β_t	Requested bandwidth capacity for traffic t
Y_e	Physical transmission delay of link e	Δ_t	Maximum allowable latency/delay in SLA for traffic t
S	Set of available servers in the network	ρC_t	SLA penalty cost factor per unit of delay violation
$s \in S$	A physical server instance	$W_t \in V^*$	Service Function Chain (SFC) requested by traffic t
Virtual Network Functions (VNFs) & Resources		**Multi-Stage Graph Modeling (Section VI)**	
V	Set of available VNF types	$G_t = (N_t, E_t)$	Virtual multi-state multi-stage graph for request t
$v \in V$	A specific Virtual Network Function type	E_i	Virtual node stage index $i \in \{1, \ldots, \lvert W_t \rvert + 2\}$
R	Set of resource types (CPU, RAM, storage)	$E_1 = \{o_t\}$	Start stage containing the source node
Q_{rs}	Available capacity of resource $r \in R$ on server s	$E_{\lvert W_t \rvert+2} = \{\omega_t\}$	Terminal stage containing the destination node
M_{rv}	Demand of resource $r \in R$ to instantiate VNF v	E_i $(i > 1)$	Intermediate node stages composed of server candidate nodes
λC_v	License cost associated with utilizing VNF v	(n_j, n_k)	Virtual directed transition link from stage E_i to E_{i+1}
δC	Standard physical deployment cost per server activation	P_t	Selected feasible routing path for traffic request t
Decision Variables			
X_{tvn}	Binary variable: 1 if VNF v requested by traffic t is allocated at node n; 0 otherwise.		
A_{tvs}	Binary variable: 1 if VNF v requested by traffic t is allocated on server s; 0 otherwise.		
X_{vs}	Binary variable: 1 if VNF type v is active on server s; 0 otherwise.		
X_{ns}	Binary variable: 1 if server s is active/installed at physical node n; 0 otherwise.		
X_{te}	Binary variable: 1 if traffic request t utilizes physical link e; 0 otherwise.		
Y_t	Binary variable: 1 if traffic request t is successfully placed and accepted; 0 otherwise (SLA reject).		
Y_s, Y_e	Binary variables indicating if server s or physical link e are active/utilized; 0 otherwise.		

It is noteworthy that the different traffic flows utilize the Firewall located on the same server n_8 without the need to instantiate it twice. It must be considered that when traffic traverses the links, it consumes bandwidth, and therefore, the physical capacity constraints of the links must be respected.

4.3. Multi/Many Objective Optimization Problems

The goal of a *Multi-Objective Optimization Problem* (MOOP) [45, 46] is to find a solution capable of simultaneously optimizing a set of objective functions while satisfying the defined constraints of the problem. Optimization implies either minimizing or maximizing an objective function; accordingly, a MOOP may require the minimization of certain objectives while simultaneously maximizing others.

A MOOP can be defined in a general manner as follows [46]:

$$\begin{aligned} \text{Minimize/Maximize } & f_m(x), && m = 1, 2, \ldots, \mathtt{M} \\ \text{subject to: } & g_j(x) \geq 0 && j = 1, 2, \ldots, \mathtt{J} \\ & s, \mathtt{K} \\ & x_i^L \leq x_i \leq x_i^U && i = 1, 2, \ldots, \mathtt{N}. \end{aligned} \tag{1}$$

A solution x consists of a vector of $\mathtt{N}$ decision variables: $x = (x_1, \ldots, x_{\mathtt{N}})$. The value a decision variable can take lies between a lower bound x_i^L and an upper bound x_i^U. These bounding restrictions constitute the decision space $\mathtt{D}$.

The problem constraints consist of $\mathtt{J}$ inequalities and $\mathtt{K}$ equations, where $g_j(x)$ and $h_k(x)$ are referred to as constraint functions. The set of solutions that satisfy the $\mathtt{J{+}K}$ constraints and the $\mathtt{2N}$ bounds is defined as the feasible region $\mathtt{S}$, where $\mathtt{D} \subseteq \mathtt{S}$.

In a MOOP, there are $\mathtt{M}$ objective functions $f(x) = (f_1(x), \ldots, f_{\mathtt{M}}(x))$ that constitute a multidimensional space referred to as the objective space $\mathtt{Z}$, where each objective function represents either a minimization or a maximization problem. For each solution $x \in \mathtt{S}$, there is a corresponding point in the objective space, denoted as $f(x) = z = (z_1(x), \ldots, z_{\mathtt{M}}(x))$. Note that this maps an $\mathtt{N}$-dimensional vector to an $\mathtt{M}$-dimensional vector.

Finding a single optimal solution that simultaneously optimizes all objective functions in a MOOP is a fundamentally different task compared to single-objective optimization, due to the presence of conflicting objective functions. This leads to the existence of compromise solutions, meaning that improving one objective function inevitably degrades another conflicting objective.

Without loss of generality, let us assume that all objective functions are to be minimized. In this context, given two solutions $x, y \in \mathtt{S}$, it is said that x dominates y, denoted as $x \succ y$, if $\forall m$ it holds that $f_m(x) \leq f_m(y)$ and $\exists m'$ for which $f_{m'}(x) < f_{m'}(y)$. In a MOOP, a set of mutually non-dominated solutions is called the Pareto Set, denoted as $\mathtt{CP} = \{\forall x, y \in \mathtt{S} : x \not\succ y, y \not\succ x\}$. The mapping of $\mathtt{CP}$ to the objective space $\mathtt{Z}$ is referred to as the Pareto Front, $\mathtt{FP} = \{z = f(x) : x \in \mathtt{CP}\} \subseteq \mathtt{Z}$. Specifically, the set of non-dominated solutions within the entire feasible region is defined as the Optimal Pareto Set $\mathtt{CP}^* = \{x \in \mathtt{S} : \nexists y \succ x\}$, and its corresponding mapping is the Optimal Pareto Front $\mathtt{FP}^* = \{z = f(x) : x \in \mathtt{CP}^*\} \subseteq \mathtt{Z}$. Note that for any given $\mathtt{CP}$, it holds true that $\mathtt{CP}^* \succeq \mathtt{CP}$.

To illustrate this explanation, a two-dimensional graph can help visualize key concepts in MOOPs: the decision space, the objective space, the Pareto set, and the Pareto front. To construct the graph, random points are generated in the decision space and projected onto the objective space, highlighting the Pareto front.

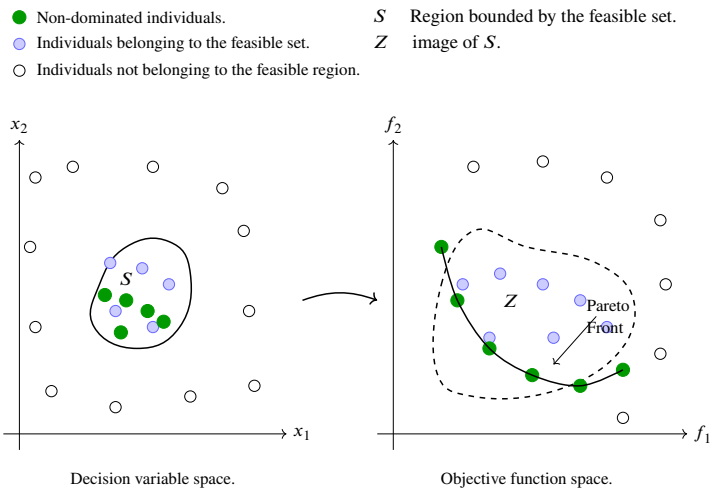


Figure 3: Optimal Pareto Set

On the left side of Figure 3, the space representing the decision variables $x = (x_1, x_2)$ is shown, with points representing the decision variable space. Feasible points are colored in light blue and green, which together form the feasible set S. The non-dominated points are highlighted in green. Finally, points that do not belong to the feasible set are shown in pink.

On the right side, the objective space is depicted, where each point corresponds to $f = (f_1, f_2)$ associated with the solutions from the decision space. The green points represent the Pareto Front (non-dominated solutions).

The size of the Optimal Pareto Set increases significantly as the number of objective functions grows, thereby increasing the complexity of the problem. In this regard, the literature distinguishes problems as low complexity if the MOOP considers two or three objective functions. However, if the number of objectives is greater than three, it is classified as a Many-Objective Optimization Problem (MaOP) [12].

4.4. Multi-Objective Formulation of the VNF-P Problem

Given an NFV environment defined by a topology G, a set of available physical servers S in the network, a set of available VNFs V, and a set of traffic requests T, the VNF-P problem seeks to compute a solution Θ that simultaneously minimizes the objective functions detailed in Table 2. This is formulated in the following expression:

$$Minimize\ F = (f_1, f_2, \ldots, f_M) \tag{2}$$

where $M = 17$ and $f_i \in [0, \infty)$, subject to the available resource constraints on links and nodes.

The formulations and explanations of the objective functions are presented in equations (3) through (22), whereas the constraints are provided in equations (23) through (29).

4.5. VNF-P Problem Objective Functions

The objective functions to be optimized in the VNF-P problem can be classified into OPEX, CAPEX, and QoS levels, and they are defined as follows.

4.5.1. Energy Cost

This function calculates energy consumption across node devices such as routers and servers:

$$f_1 = f_{1.1} + f_{1.2} \tag{3}$$

where:

- $f_{1.1}$ is the energy consumption of the nodes to satisfy the traffic requests:

$$f_{1.1} = \sum_{t \in T} \sum_{n \in N_t} \tau C_n, \tag{4}$$

- $f_{1.2}$ is the energy consumption of the servers, calculated as:

$$f_{1.2} = \sum_{s \in S} Y_s \cdot \left((maxP_s - P_s) \cdot \frac{U_{cpu\ s}}{Q_{cpu\ s}} + P_s \right) \cdot \tau C_n, \tag{5}$$

- τC_n is the unitary energy consumption cost for the active physical node n,
- Y_s is a binary variable, 1 if server s is active, 0 otherwise,
- $maxP_s$ is the maximum energy available on server s,
- P_s is the energy consumed by server s when idle,
- Q_{rs} is the quantity of computational resources $r = cpu$ available on server s, and
- U_{rs} is the quantity of computational resources $r = cpu$ utilized on server s.

4.5.2. Bandwidth Consumption

This function computes the total bandwidth consumed across the network:

$$f_2 = \sum_{e \in E} uB_e \tag{6}$$

where uB_e is the bandwidth utilized on link e.

4.5.3. Latency

This function calculates the total traffic latency, which is generated by link propagation delays and the processing delay of each VNF type deployed on the servers:

$$f_3 = \sum_{t \in T} \left(\sum_{e \in E_t} Y_e + \sum_{v \in W_t} \Psi_v \right) \tag{7}$$

where:

- Y_e is the data transmission delay on link e utilized by traffic t, and
- Ψ_v is the processing delay generated by VNF v.

4.5.4. Traffic Load

This objective function computes the traffic load by considering the operational cost of each active physical link, based on the delay and the total allocated bandwidth of the virtual links embedded within the physical link. In this work, the originally proposed distance variable found in the state of the art [33] was replaced by the delay variable.

$$f_4 = \sum_{n \neq n' \in N} Y_{nn'} \cdot \sum_{t \in T} \sum_{v \neq v' \in W^t} uB_{tvv'} \cdot X_{tvn} \cdot X_{tv'n'} \quad (8)$$

where:

- $Y_{nn'}$ is the propagation delay of a route segment between nodes n and n',
- $uB_{tvv'}$ is the bandwidth utilized in the route segment defined by VNFs v and v' for traffic t, and
- $X_{tvn}, X_{tv'n'}$ are binary variables, set to 1 if VNF v (or v') requested by traffic t is allocated to the server connected to node n (or n'), and 0 otherwise.

4.5.5. Installation Cost

This function computes the total installation cost for the servers plus the deployment cost for the VNFs:

$$f_5 = \sum_{s \in S} Y_s \cdot \left(\delta C_s + \sum_{t \in T} \sum_{v \in W_t} A_{tvs} \cdot \psi C_v \right) \quad (9)$$

where:

- δC_s is the deployment cost on server s,
- ψC_v is the unitary cost to deploy VNF v,
- W is the service function chain (SFC) requested by the traffics,
- A_{tvs} is a binary variable, 1 indicates that VNF v from request t is deployed on server $s \in S$, 0 otherwise, and
- Y_s is a binary variable, 1 indicates that server s is active, 0 otherwise.

4.5.6. Total Resource Cost

This function calculates the cost of utilizing computational resources such as memory, CPU, and storage:

$$f_6 = \sum_{r \in R} \sum_{s \in S} U_{rs} \cdot \kappa C_{rs} \quad (10)$$

where:

- U_{rs} is the amount of computational resources of type r utilized in server s, and
- κC_{rs} is the unitary cost of computational resource r in server s.

To provide a realistic estimation of Operational Expenditures (OPEX) driven by resource utilization, the unitary cost parameters can be modeled after real-world public cloud pricing schemas, such as Amazon Web Services (AWS) Elastic Compute Cloud (EC2). For instance, an AWS EC2 `m5.large` instance (featuring 2 vCPUs and 8 GB of RAM) incurs a standard on-demand cost of approximately $0.096 per hour in the US East region. By translating these instance-level prices into unitary costs for individual compute and memory resources (κC_{rs}), the objective function accurately reflects real monetary expenditures, thereby facilitating practical OPEX minimization in NFV deployments.

4.5.7. Resource Fragmentation

This function computes the cost of computational resources and bandwidth that have been allocated but remain unutilized:

$$f_7 = \sum_{s \in S} Y_s \cdot \sum_{r \in R} (Q_{rs} - U_{rs}) \cdot \phi C_r + \sum_{e \in E} Y_e \cdot (B_e - uB_e) \cdot \sigma C \quad (11)$$

where:

- Y_s is a binary variable, set to 1 if server s is utilized, and 0 otherwise,
- Y_e is a binary variable, set to 1 if link e is utilized, and 0 otherwise,
- Q_{rs} is the quantity of computational resources of type r available in server s,
- U_{rs} is the quantity of computational resources of type r utilized in server s,
- B_e is the bandwidth available on link e,
- uB_e is the bandwidth utilized on link e,
- σC is the unitary cost for bandwidth underutilization, and
- ϕC_r is the unitary cost for computational resource r underutilization.

4.5.8. Utilization Cost of All Network Links

This function computes the economic costs of the bandwidth utilized by traffic requests:

$$f_8 = \sum_{e \in E} uB_e \cdot \beta C_e \quad (12)$$

where:

- uB_e is the bandwidth utilized on link e, and
- βC_e is the unitary cost for the bandwidth of link e.

4.5.9. Maximum Utilized Network Link

This function calculates the link bearing the heaviest traffic load, as shown in the formula:

$$f_9 = \max \left\{ uB_e : e \in E \right\} \quad (13)$$

where uB_e is the bandwidth utilized on link e.

4.5.10. License Cost

This function computes the total sum of the license costs for both VNFs and servers:

$$f_{10} = \sum_{s \in S} Y_s \cdot \left(\eta C_s + \sum_{v \in V} X_{vs} \cdot \lambda C_v \right) \tag{14}$$

where:

- X_{vs} is a binary variable, set to 1 if VNF v is allocated on server s, and 0 otherwise,
- λC_v is the license cost of VNF v,
- ηC_s is the license cost of server s, and
- Y_s is a binary variable, set to 1 if server s is utilized, and 0 otherwise.

4.5.11. SLO Cost

This objective function calculates the penalty that must be paid to the client for Service Level Agreement (SLA) / Service Level Objective (SLO) violations:

$$f_{11} = \sum_{t \in T} \varrho C_t \tag{15}$$

where:

$$\varrho C_t = \begin{cases} \varrho C_t, \text{ if } D_t > \Delta_t \\ 0, \text{otherwise} \end{cases} \tag{16}$$

- ϱC_t is the penalty cost for an SLA violation for traffic t,
- Δ_t is the maximum allowed data transmission delay for traffic t, and
- D_t is the total propagation delay for traffic t.

4.5.12. Distance Traveled

This function calculates the sum of the total distance traveled by the traffic requests:

$$f_{12} = \sum_{t \in T} \sum_{e \in E_t} L_e \tag{17}$$

where L_e is the distance of link e.

4.5.13. Number of Hops

This function computes the total number of hops made by the traffic requests:

$$f_{13} = \sum_{e \in E} Z_e \tag{18}$$

where Z_e is an integer variable that counts the number of times traffic passes through link e.

4.5.14. Number of Servers

This objective function counts the number of servers utilized for VNF placement:

$$f_{14} = \sum_{s \in S} Y_s \tag{19}$$

where Y_s is a binary variable set to 1 if server s is utilized, and 0 otherwise.

4.5.15. Number of VNF Instances

This function sums the quantity of deployed VNF instances.

$$f_{15} = \sum_{s \in S} \sum_{v \in V} X_{sv} \tag{20}$$

where X_{sv} is a binary variable set to 1 if VNF v is allocated on server s, and 0 otherwise.

4.5.16. Effective Network Throughput

Given a list of traffic requests, this function aims to improve bandwidth utilization on each link by maximizing the ratio between accepted and rejected requests:

$$f_{16} = \frac{\sum_{t \in T} \beta_t \cdot Y_t}{\sum_{t \in T} \beta_t} \tag{21}$$

where:

- $\mathcal{B}_t$ is the requested bandwidth for traffic t, and
- Y_t is a binary variable, 1 if traffic t is placed within the network, 0 otherwise.

4.5.17. Percentage of Allocated VNFs

This function calculates the number of allocated VNFs requested by the traffic flows:

$$f_{17} = \frac{\sum_{t \in T} \sum_{s \in S} \sum_{v \in W_t} A_{tvs}}{\sum_{t \in T} \sum_{v \in W_t} H_{tv}} \tag{22}$$

where:

- A_{tvs} is a binary variable, 1 if VNF v from traffic t is deployed on server s, 0 otherwise, and
- H_{tv} is a binary constant, 1 if VNF v is requested by traffic t, 0 otherwise.

4.6. VNF-P Problem Constraints

The VNF-P problem is subject to constraints at both the physical node and link levels, as well as at the virtual resource level, which are detailed below.

4.6.1. Server to Node Allocation

A server can only be allocated to a single node, expressed as follows:

$$\sum_{s \in S} X_{ns} = 1, \quad \forall n \in N \tag{23}$$

where X_{ns} is a binary variable set to 1 if server s is installed in node n, and 0 otherwise.

4.6.2. Server Resource Capacity

The sum of the resources utilized on a server must not exceed the total capacity of that server.

$$U_{rs} \leq Q_{rs}, \quad \forall r \in R, \quad \forall s \in S \tag{24}$$

where:

- U_{rs} represents the quantity of resources of type r utilized in server s, $U = \{U_{rs} : r \in R, s \in S\}$.
- Q_{rs} represents the quantity of computational resources of type r available in server s, $Q = \{Q_{rs} : r \in R, s \in S\}$.

4.6.3. VNF Resource Capacity

The sum of the resources utilized by a specific VNF type must not exceed the total capacity of the installed VNF of that same type available for reuse.

$$K_{rv} \leq J_{rv}, \quad \forall v \in V, \quad \forall r \in R \tag{25}$$

where:

- K_{rv} represents the quantity of resources of type r utilized in the installed VNF v, $K = \{K_{rv} : r \in R, v \in V\}$.
- J_{rv} represents the quantity of available resources that the installed VNF can support, $J = \{J_{rv} : r \in R, v \in V\}$.

4.6.4. Bandwidth Capacity

The sum of the bandwidth utilized on a link must not exceed the available bandwidth of that link.

$$uB_e \leq B_e, \quad \forall e \in E \tag{26}$$

where:

- B_e is the bandwidth available on link e, $B = \{B_e : e \in E\}$.
- uB_e is the bandwidth utilized on link e, $uB = \{uB_e : e \in E\}$.

4.6.5. VNF Installation on Server

The resource demand of a VNF must not exceed the available capacity of the server where it is deployed.

$$M_{rv} \cdot X_{vs} \leq (Q_{rs} - U_{rs}), \quad \forall s \in S, \forall v \in V, \forall r \in R \tag{27}$$

where:

- U_{rs} is the quantity of computational resources of type r utilized in server s, $U = \{U_{rs} : r \in R, s \in S\}$.
- Q_{rs} is the quantity of computational resources of type r available in server s, $Q = \{Q_{rs} : r \in R, s \in S\}$.
- M_{rv} is the computational resource requirement of type r needed by VNF v, $M = \{M_{rv} : r \in R, v \in V\}$.
- X_{vs} is a binary variable, set to 1 if VNF v is allocated on server s, and 0 otherwise.

4.6.6. VNF Reuse

The resource demand of the VNF type intended to be reused must not exceed the available capacity of the already installed VNF of the same type.

$$M_{rv} \leq (J_{rv} - K_{rv}), \quad \forall v \in V, \quad \forall r \in R \tag{28}$$

where:

- K_{rv} is the quantity of computational resources of type r utilized in the installed VNF v, $K = \{K_{rv} : r \in R, v \in V\}$.
- J_{rv} is the quantity of available computational resources of type r that the installed VNF v can support, $J = \{J_{rv} : r \in R, v \in V\}$.
- M_{rv} is the computational resource requirement of type r that VNF v will reuse, $M = \{M_{rv} : r \in R, v \in V\}$.
- X_{vs} is a binary variable, set to 1 if VNF v is allocated on server s, and 0 otherwise.

4.6.7. Traffic Bandwidth

The bandwidth requested by the traffic must not exceed the bandwidth available on the link:

$$uB_{tvv'} \cdot X_{te} \leq (B_e - uB_e), \quad \forall e \in E, \forall v \in V, \forall t \in T \tag{29}$$

where:

- B_e is the available bandwidth on link e.
- uB_e is the utilized bandwidth on link e.
- $uB_{tv_a v_b}$ is the bandwidth utilized in the route segment defined by VNFs v_a and v_b for traffic t.
- X_{te} is a binary variable set to 1 if request t utilizes link e, and 0 otherwise.

4.7. Numerical Example

This section introduces a basic example to elucidate the formulations provided in the preceding section. The problem instance is illustrated in Figure 4, and its input parameters are detailed in Tables 5 through 10.

Table 5 presents five VNF types in the rows, while the columns indicate costs (which are reference values in generic units) and associated characteristics. The VNF types, delay, deployment cost, license cost, bandwidth modification factor, required secondary memory, required number of cores, and required RAM are defined by $V, \Psi, \psi C, \lambda C, \alpha\beta, M_{v_i\, r_1}, M_{v_i\, r_2}$, and $M_{v_i\, r_3}$, respectively. The considered VNFs can be deployed on the same server depending on available capacity and requirements. Note that the Bandwidth Modification Factor column indicates the modification rate that the VNF exerts on the incoming traffic.

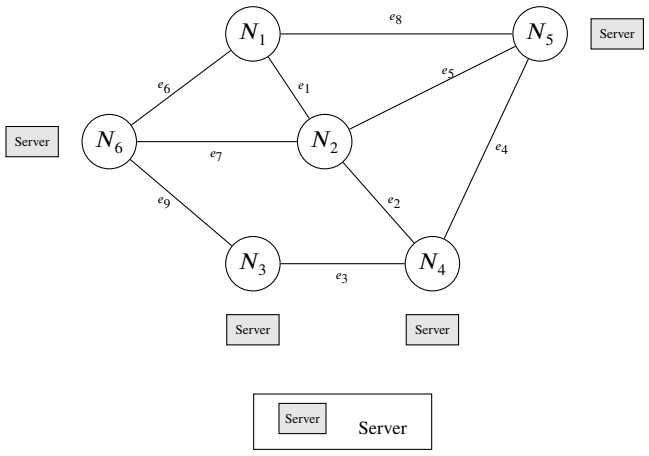


Figure 4: Original graph

Table 5: Available VNF Parameters

V	Ψ	ψC	λC	$\alpha\beta$	$M_{v_i\, r_1}$	$M_{v_i\, r_2}$	$M_{v_i\, r_3}$
v_1 = Firewall	20	10	50	1	30	4	8
v_2 = IDS	12	9	45	1,3	35	6	8
v_3 = Proxy	5	12	60	1.5	30	4	8
v_4 = NAT	8	8	55	0,8	30	4	6
v_5 = DPI	10	11	52	1,2	35	4	8

Table 6 presents the network link parameters. The rows display the bidirectional links, while the columns detail the set of links E, along with reference costs in generic units such as delay τ, distance L, bandwidth usage cost βC, and maximum link capacity B.

Table 7 presents the network node parameters. The columns display the set of nodes N, the set of servers S, and the energy cost τC.

Table 6: Network Link Parameters

E	τ	L	βC	B
e_1	7	690	0.14375	1000
e_8	9	1068	1,19E-05	1100
e_6	6	548	0.01223	1070
e_2	8	911	0,11931	1019
e_7	7	712	0,12741	1027
e_9	6	613	0,13651	1036
e_3	5	568	0,14531	1045
e_5	6	648	0,14561	1054
e_4	7	768	0,15471	1063

Table 7: Network Node Parameters

N	S	τC
n_1	-	1.3E-4
n_2	-	1.3E-4
n_3	s_3	1.9E-4
n_4	s_4	1.0E-4
n_5	s_5	1.1E-4
n_6	s_6	1.4E-4

Table 8 presents the parameters for traffic requests t_1, t_2, and t_3. The rows display each request t, while the columns outline their respective requirements. Each request is composed of a source node o, a destination node ω, the requested bandwidth β, the maximum allowable delay without penalty Δ, the penalty cost for exceeding the maximum delay ϱC, and the requested service chain W. Furthermore, Table 9 presents the specific VNF requirements for these requests.

Table 10 presents the parameters of the available servers. This table details the set of servers S alongside their reference costs in generic units, which include the license cost λC, the deployment cost δC, the maximum energy $maxP$, the idle energy P, the cost per core $\kappa C_{s_i\, r_1}$, the RAM cost $\kappa C_{s_i\, r_2}$, the storage cost $\kappa C_{s_i\, r_3}$, the number of available cores $Q_{s_i\, r_1}$, the available RAM capacity $Q_{s_i\, r_2}$, and the available external storage capacity $Q_{s_i\, r_3}$.

Table 8: Request Parameters

T	o	ω	β	Δ	ϱC	W
t_1	n_5	n_1	5	124	2	v1 → v2 → v4
t_2	n_5	n_3	3	116	1	v3 → v2 → v1
t_3	n_1	n_6	5	78	1	v4 → v5

Table 9: VNFs Requested per Traffic

VNF	t_1		t_2		t_3	
	RAM	**CPU**	**RAM**	**CPU**	**RAM**	**CPU**
v_1	2	3	2	4	-	-
v_2	2	4	2	3	-	-
v_3	-	-	2	3	-	-
v_4	1	2	-	-	2	3
v_5	-	-	-	-	1	3

Table 10: Server Parameters

S	λC	δC	$maxP$	P	$\kappa C_{s_i\, r_1}$	$\kappa C_{s_i\, r_2}$	$\kappa C_{s_i\, r_3}$	$Q_{s_i\, r_1}$	$Q_{s_i\, r_2}$	$Q_{s_i\, r_3}$
s_3	85	2	2735	80	0.06	0.03	0.01	12	32	100
s_4	100	3	3000	100	0.07	0.04	0.02	12	32	106
s_5	110	2	3500	85	0.08	0.35	0.02	12	32	124
s_6	90	1	2735	94	0.075	0.048	0.01	12	32	130

An example solution is provided in Table 11, where the rows indicate the requests and the columns display the routing and placement for each requested VNF type. Figure 5 illustrates the solution graphically, depicting both the routing and the placement of the VNFs. The solutions for requests t_1, t_2, and t_3 are the following, respectively:

- $n_5 \rightarrow n_3 \rightarrow v_1 \rightarrow v_2 \rightarrow n_6 \rightarrow v_4 \rightarrow n_1$,
- $n_5 \rightarrow n_4 \rightarrow v_3 \rightarrow v_2 \rightarrow n_2 \rightarrow n_6 \rightarrow v_1 \rightarrow n3$,
- $n_1 \rightarrow n_6 \rightarrow v_4 \rightarrow n_1 \rightarrow n_5 \rightarrow v_5 \rightarrow n_3 \rightarrow n_6$.

Note that server s_3 hosts functions v_1 and v_2, server s_4 hosts functions v_2 and v_3, server s_5 hosts function v_5, and server s_6 hosts functions v_1 and v_4, where v_4 is reused by traffic requests t_1 and t_3. As observed in the solution, depending on the placement of the VNFs, it is possible for the path connecting a source and destination node pair to exhibit a cyclic structure. This implies resource overutilization in both links and nodes, as well as delays in traffic transfer. Consequently, this structure is significantly more complex than conventional routing.

Based on the example in Figure 5, the following values can be observed:

- Energy Cost (f_1): For practical purposes, this function was divided into two operations. Operation $f_{1.1}$ computes the sum of the energy costs utilized by passing through each node. Request t_1 adds an energy cost of 0.00057 units for passing through the nodes, t_2 adds 0.00068 units, and t_3 adds 0.00084 units. In total, the energy cost for utilizing the nodes across all three requests is 0.00209 units. Operation $f_{1.2}$ calculates the costs related to energy utilization in the servers. The amount of energy utilized is related to the number of active cores in the server multiplied by the energy cost. The sum of energy costs in servers s_3, s_4, s_5, and s_6 is 1.204715. The total energy cost ($f_{1.1} + f_{1.2}$) amounts to 1.206798332 units.

- Bandwidth Consumption (f_2): The total bandwidth utilized across all links is 59.85 units.

- Latency (f_3): The accumulated delays for t_1, generated within the links E_t traversed by the traffic in the solution, are calculated by summing the following latency values: $e_5 = 6$, $e_9 = 6$, and $e_6 = 6$, yielding a total of 18 units. Similarly, the processing delay for each VNF requested by t_1 is summed: the delay for v_1 is 20 time units, for v_2 is 12 time units, and for v_4 is 8 time units. Summing these yields a total of 40 time units for processing. Thus, the total delay for t_1 is 18 (links) + 40 (processing), resulting in a total of 58 time units. The same procedure applies to the other traffic requests: for t_2, the accumulated delay is 65 time units, and for t_3, it is 51 time units. The total accumulated delay for the set of traffic requests T is 174 time units.

- Traffic Load (f_4): The traffic load is calculated by summing the product of the delay time and the bandwidth for each link between two servers hosting VNFs. The load for t_1 is calculated from node n_3 (which hosts v_1 and v_2) to node n_6 (which hosts v_4), presenting a delay of 6 time units and a bandwidth on link e_9 equal to 6.5 units (considering the modification factor $\alpha\beta$). This results in 6×6.5, equaling 39 units. For t_2 and t_3, the result is 87.75 and 60 load units, respectively. Finally, the total sum of the products is 186.75 load units.

- Installation Cost (f_5): The sum of installation costs for the utilized servers s_3, s_4, s_5, and s_6 totals 8 units. On server s_3, VNFs v_1 and v_2 were installed at a cost of 10 and 9 units, respectively. On server s_4, the installed functions v_2 and v_3 sum a cost of 21 units. On server s_5, v_5 cost 11 units, and lastly, on s_6, v_1 and v_4 were installed at a cost of 18 units. The sum of all VNF installation costs is 69 units, plus the server installation costs, yielding a total of 77 units.

- Total Resource Cost (f_6): On server s_3, 10 cores, 16 RAM units, and 65 storage units were utilized. Multiplying these by their respective costs yields a total cost of 1.73 units for s_3. For s_4, s_5, and s_6, the respective costs are 2.64, 1.3, and 2.76 units. The total cost of the utilized resources is 8.43 units.

- Resource Fragmentation (f_7): On server s_3, 2 cores, 16 RAM units, and 35 storage units remained idle. Each unutilized resource is multiplied by its respective penalty cost, which are 0.0001 for cores, 0.0002 for RAM, and 0.0004 for storage. In summary, the penalty cost on s_3 is 0.0174 units. For s_4, s_5, and s_6, the penalty costs are 0.0198, 0.0412, and 0.0184 respectively, totaling 0.0968 units. This is subsequently added to the cost for not fully utilizing the bandwidth on each link, where a penalty of 0.0003 per unit applies. The total link penalty cost is 2.1927 units. The overall cost of the solution, summing the penalty costs across servers and links, is 2.289545 units.

- Utilization cost of all network links (f_8): For this function, the unitary bandwidth cost is multiplied by the bandwidth utilized on each link. The final result is 0.000104952 units.

- Maximum Network Link Utilized (f_9): This value represents the maximum bandwidth utilized among all network links, resulting in 17.15002 bandwidth units.

- License Cost (f_{10}): Two types of VNFs were installed on servers s_3, s_4, and s_6. The license cost for each server plus the license cost for each installed VNF yields a total of 180, 205, and 195 units, respectively. On s_5, a single VNF was installed, with a VNF license cost of 52 units, which, added to the server license cost, results in 162 units. The total cost for all licenses is 742 units.
- SLO Cost (f_{11}): Each request is associated with a maximum allowable delay. If the request's delay in the solution exceeds this defined maximum, the specific penalty cost for that request is applied. For the solution in this example, no request exceeds the maximum delay; hence, this cost is 0.
- Traveled Distance (f_{12}): The sum of the distances traveled by each request is computed: t_1 utilizes a route distance of 1.809, t_2 utilizes 3.004, and finally, t_3 totals 3.525 units. The total distance for the solution is 8.238 units.
- Number of Hops (f_{13}): The total number of hops performed by each request across the links amounts to 12 units.
- Number of Servers (f_{14}): 4 nodes were utilized to install the servers, resulting in a cost of 4 units.
- Number of VNF Instances (f_{15}): The total number of VNF instances deployed on the servers reaches 7 units.
- Effective Network Throughput (f_{16}): In this example, all functions were successfully placed, resulting in a 100% effectiveness rate.
- Percentage of Allocated VNFs (f_{17}): In this example, all functions were successfully placed, yielding a rate of 100%.

In summary, by grouping the objective functions by type, a total OPEX cost of 8349.926 units, a total CAPEX cost of 742 units, and a total QoS cost of 379.900 units are obtained.

Table 11: Numerical example solution

T	Routing	VNF Placement					
		s_1	s_2	s_3	s_4	s_5	s_6
t_1	n_5, n_3, n_6, n_1	-	-	v_1, v_2	-	-	v_4
t_2	n_5, n_4, n_2, n_6, n_3	-	-	-	v_2, v_3	-	v_1
t_3	$n_1, n_6, n_1, n_5, n_3, n_6$	-	-	-	-	v_5	v_4

5. Multi-objective Evolutionary Algorithms

To address the formulation of the VNF-P problem based on the selected objectives, this work relies on Evolutionary Algorithms (EAs). Traditional Multi-Objective Evolutionary Algorithms (MOEAs) [45] are well-suited for problems with two or three objectives [47]. However, they suffer from significant scalability issues—such as loss of selection pressure, exponential increase in the proportion of non-dominated solutions, and computational overhead in diversity preservation—when the number of objectives grows [12]. This performance degradation motivates the use of Many-Objective Evolutionary Algorithms (MaOEAs) [48] for our VNF-P model considering the chosen set of objectives.

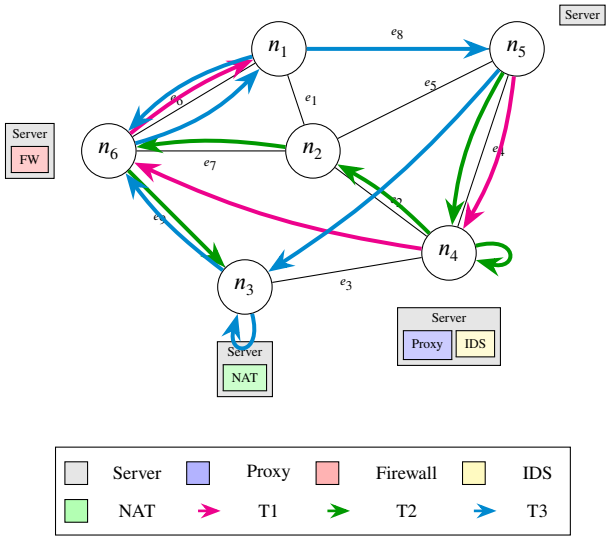


Figure 5: Final resulting graph

To manage the high dimensionality of the objective space, MaOEAs are generally grouped into families based on their diversity and selection mechanisms: decomposition-based approaches that optimize scalarizing sub-problems, relaxed dominance strategies (e.g., ϵ- or α-dominance), indicator-based methods (e.g., hypervolume), and objective reduction techniques. Algorithm 1 outlines the general evolutionary scheme shared by these algorithms.

Algorithm 1 General Scheme of MOEA/MaOEA

```
Inputs: Evolutionary parameters and problems to solve
Output: Pareto Set
Begin:
    Initialize population P with N individuals
    Evaluate the objective functions and indicators for each individual in P
    While the stopping criterion is not met do:
        Select subsets of mating individuals
        Crossover mating individuals to generate new offspring
        Mutate the individuals generated by crossover
        Evaluate the objective functions and indicators for each individual
        Combine population P with the generated offspring
        Select the individuals that survive to the next generation
        Update population P
    End While
    Obtain the Pareto Set from P
    Return Pareto Set
End
```

6. Evolutionary VNF-P

For the application of MOEAs, we define the chromosomal representation of the individual, the method for initializing the population, the processes corresponding to the evolutionary operators (evaluation, selection, and crossover), and the input parameters of the algorithms. Figure 6 presents an example that correlates a solution, a chromosome structure, and the evaluation process.

6.1. Chromosome Structure

The chromosome structure is a permutation encoded in a vector whose size is equal to the number of requests to be processed. Each i-th position of the vector indicates the i-th traffic request for which the placement and routing must be calculated via a heuristic. A traffic request is represented by $t = (o_t, \omega_t, \beta_t, \Delta_t, \rho C_t, W_t)$,

while the set of requests t is stored in a matrix. Figure 7 depicts a table comprising four requests, where each row stores the data corresponding to the requests t.

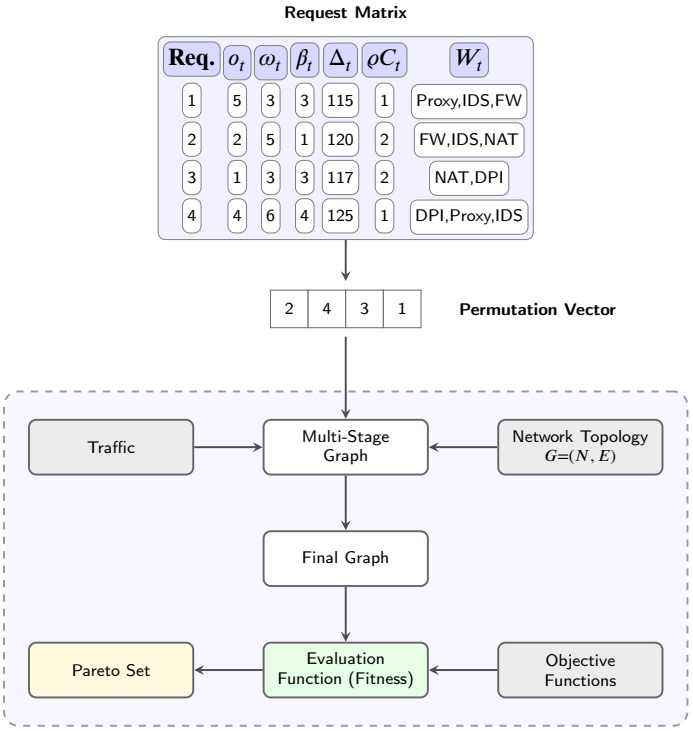


Figure 6: Operational scheme of an evolutionary VNF-P.

Furthermore, the i-th gene of the chromosome vector points to a row in the table where the i-th request to be processed is located. In the example provided in Figure 6, the chromosome vector [2,4,3,1] is observed, indicating that the requests defined in the table will be processed in that specific sequence.

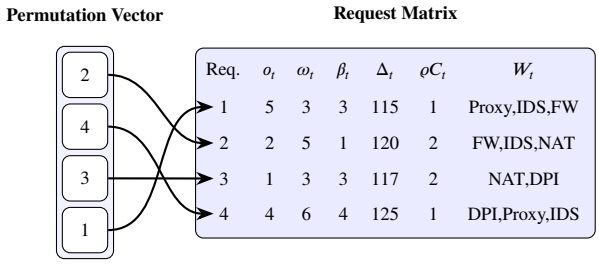


Figure 7: Permutation Vector.

6.2. Multi-stage Graph Modeling

In this work, we extend the approach described in [49], where it is suggested that a traffic request $t = (o_t, \omega_t, \beta_t, \Delta_t, \rho C_t, W_t)$ be mapped onto a virtual directed multi-stage graph. This virtual graph consists of $|W_t| + 2$ stages, where $|W_t|$ denotes the length of the service chain W_t. The first stage E_1 comprises the source node o_t, whereas the final stage consists of the destination node ω_t, i.e., $E_1 = \{o_t\}$, $E_{|W_t|+2} = \{\omega_t\}$.

Each intermediate stage E_i ($i \in \{2, ..., (|W_t| + 1)\}$), is composed of physical nodes featuring candidate servers for placing the j-th VNF, with $j = i$ - 1. Evidently, these servers must possess available computational resources to host the VNF. For each pair of nodes $n_j \in E_i$ and $n_k \in E_{i+1}$, a virtual link (n_j,n_k) is added between them. Finally, once the multi-stage graph is constructed, it is verified whether at least one valid path exists from the source node o_t to the destination node ω_t. A valid path will not exist if at least one of the intermediate stages is an empty set. Figure 8 presents the final structure of the multi-stage graph corresponding to the original network topology and traffic t_2.

Formally, for a given traffic request t, the multi-stage graph is defined as a directed graph $\hat{G}_t = (\hat{V}_t, \hat{A}_t)$. The set of vertices $\hat{V}_t$ is the union of all stages, i.e., $\hat{V}_t = \bigcup_{i=1}^{|W_t|+2} E_i$. The set of arcs $\hat{A}_t$ consists of directed virtual links connecting nodes in adjacent stages. An arc $(n_j, n_k) \in \hat{A}_t$ exists if and only if $n_j \in E_i$, $n_k \in E_{i+1}$ (for some $1 \leq i \leq |W_t| + 1$), and there is a feasible physical path from n_j to n_k in the underlying network G satisfying the bandwidth requirement β_t.

Algorithm 2 formalizes the multi-stage graph construction procedure described above.

Algorithm 2 Multi-Stage Graph Construction

Require: Traffic request $t = (o_t, \omega_t, \beta_t, \Delta_t, \rho C_t, W_t)$, physical graph $G = (N, E)$, resource state $\mathcal{A}$
Ensure: Multi-stage graph $\hat{G}_t = (\hat{V}_t, \hat{A}_t)$, or INFEASIBLE

$E_1 \leftarrow \{o_t\}$; $E_{|W_t|+2} \leftarrow \{\omega_t\}$
for $i \leftarrow 2$ **to** $|W_t| + 1$ **do**
  $j \leftarrow i - 1$ {j-th VNF to host: $W_t[j]$}
  $E_i \leftarrow \{n \in N :$ server at n has sufficient residual resources for $W_t[j]\}$
  if $E_i = \emptyset$ **then**
    return INFEASIBLE
  end if
end for
$\hat{A}_t \leftarrow \emptyset$
for $i \leftarrow 1$ **to** $|W_t| + 1$ **do**
  for each $n_j \in E_i$, $n_k \in E_{i+1}$ **do**
    if $\exists$ path $n_j \rightarrow n_k$ in G with residual bandwidth $\geq \beta_t$ **then**
      $\hat{A}_t \leftarrow \hat{A}_t \cup \{(n_j, n_k)\}$
    end if
  end for
end for
return $\hat{G}_t = \left(\bigcup_i E_i,\ \hat{A}_t\right)$

6.3. VNF Placement

A solution is defined as the placement of a VNF in each of the intermediate stages of the multi-stage graph. It should be recalled that there are as many intermediate stages as there are VNFs required by the corresponding traffic request.

To install the j-th VNF on a server in stage i (where $j = i$-1), the impact of its installation is evaluated across each of the servers in that stage. This impact is associated with a metric calculated after installing the VNF on the server. Ultimately, the server exhibiting the lowest impact measure is selected.

To compute the impact following the installation of the j-th

VNF on the servers of stage i, the values of the objective functions are first calculated. Subsequently, each of these values is normalized. Afterwards, the normalized values are summed and divided by the total number of objective functions, thereby yielding an average cost for each server. Finally, the servers in stage i are sorted from lowest to highest based on this cost and stored in a list according to this order.

The procedure for server selection at each stage constitutes a greedy constructive heuristic. At each step, it chooses the server that presents the lowest impact cost, provided that both the server and the physical link to this stage are available. If both resources are enabled, the VNF is installed on the selected server, and the process advances to the next stage of the multi-stage graph. It is important to note that an exact shortest-path algorithm (e.g., Dijkstra's algorithm) cannot be directly applied to the multi-stage graph because the arc costs (impact scores) are dynamically dependent on the continuously updating residual resource state $\mathcal{A}$.

In the event that the required server or link is unavailable, alternatives are considered by evaluating the next server with the lowest cost on the ranked list. If no viable options are found in the current stage, a backtracking mechanism retreats to the previous stage in order to explore alternative routes.

If, after exhausting all possible local combinations via backtracking, a valid route cannot be established, the heuristic terminates and it is concluded that a feasible solution for the corresponding traffic request cannot be found.

Physical routing between selected servers is constructed sequentially from the source to the destination, traversing the nodes of each stage and the nodes of the selected shortest-path link in the underlying physical topology.

Algorithms 3 and 4 formalize these two procedures. Finally, the solution obtained in the multi-stage graph for traffic 2 is depicted in Figure 9, while its representation in the original graph is illustrated in Figure 10. The final solution, which reflects the result of the entire process after placing traffic 1, traffic 2, and traffic 3, is captured in Figure 11. The routing of the final solution is determined as follows:

- Traffic 1: n_5, n_3, n_6, n_1
- Traffic 2: n_5, n_4, n_2, n_6, n_3
- Traffic 3: $n_1, n_6, n_1, n_5, n_3, n_6$

It is worth noting that the apparent cycle in the routing of Traffic 3 (e.g., visiting node n_1 and n_6 twice) is intentional and strictly valid within the multi-stage NFV model. A physical node can host VNFs that structure cycles or self-loops, allowing it to be visited repeatedly to satisfy complex intermediate routing and placement needs. The model independently accounts for computational resources during VNF placement and link bandwidth consumption during physical traversal, natively supporting non-simple paths when they represent the most cost-effective or feasible global placement.

Algorithm 3 VNF Impact Function

Require: Server s, VNF index j, resource state $\mathcal{A}$, objectives $f_1, \ldots, f_M$, bounds $f_k^{\min}, f_k^{\max}$
Ensure: Impact score $\text{impact}(s) \in [0, 1]$

Hypothetically install $W_t[j]$ on s (temporarily update $\mathcal{A}$)
for $k \leftarrow 1$ **to** M **do**
  Evaluate $f_k(s)$ under the hypothetical placement
  $\hat{f}_k(s) \leftarrow \left(f_k(s) - f_k^{\min}\right) / \left(f_k^{\max} - f_k^{\min}\right)$
end for
Restore $\mathcal{A}$ (undo hypothetical installation)
return $(1/M) \sum_{k=1}^{M} \hat{f}_k(s)$

Algorithm 4 Greedy VNF Placement with Backtracking

Require: Multi-stage graph $\hat{G}_t$, physical graph G, resource state $\mathcal{A}$
Ensure: Placement $P = (s_1, \ldots, s_{|W_t|})$ and route R, or UNPLACED

$i \leftarrow 2$; $P \leftarrow \langle\rangle$; $R \leftarrow \langle o_t \rangle$
$L[k] \leftarrow []$, $\pi[k] \leftarrow 1$ for all stages k {Ranked server lists and pointers}
while $i \leq |W_t| + 1$ **do**
  $j \leftarrow i - 1$
  if $L[i]$ is empty **then**
    Compute impact(s) for each $s \in E_i$ (Algorithm 3)
    $L[i] \leftarrow$ sort E_i by impact$(\cdot)$ ascending; $\pi[i] \leftarrow 1$
  end if
  $found \leftarrow$ **false**
  while $\pi[i] \leq |L[i]|$ **do**
    $s \leftarrow L[i][\pi[i]]$; $\pi[i] \leftarrow \pi[i] + 1$
    if s has resources for $W_t[j]$ **and** $\exists$ path R.last() $\rightarrow s$ with bw $\geq \beta_t$ **then**
      Install $W_t[j]$ on s (update $\mathcal{A}$); $P \leftarrow P \,\|\, \langle s \rangle$
      Append shortest path R.last() $\rightarrow s$ to R
      $found \leftarrow$ **true**; **break**
    end if
  end while
  if not $found$ **then**
    if $i = 2$ **then**
      return UNPLACED {Cannot backtrack beyond first stage}
    end if
    Undo installation of $W_t[j-1]$ from P.last() (restore $\mathcal{A}$)
    P.pop(); Remove last segment from R; $L[i] \leftarrow []$
    $i \leftarrow i - 1$ {Backtrack to previous stage}
  else
    $i \leftarrow i + 1$
  end if
end while
if $\exists$ path R.last() $\rightarrow \omega_t$ in G with bandwidth $\geq \beta_t$ **then**
  Append shortest path R.last() $\rightarrow \omega_t$ to R
  return (P, R)
end if
return UNPLACED

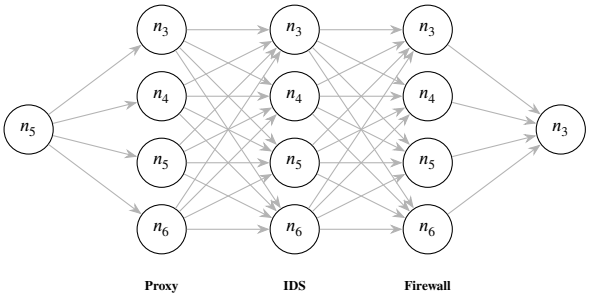


Figure 8: Multi-stage graph of traffic 2.

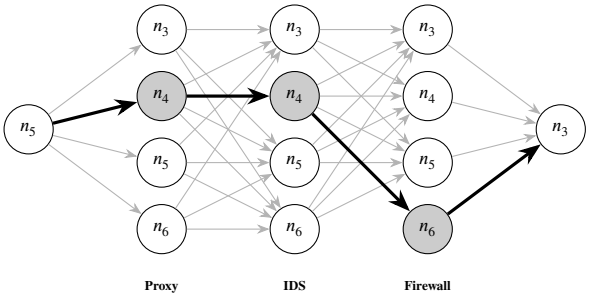


Figure 9: Multi-stage graph solution for traffic 2.

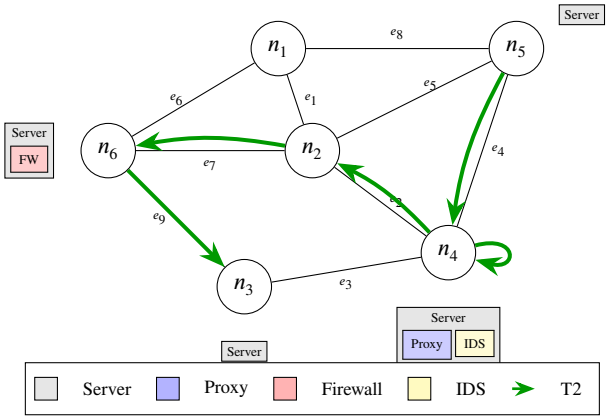


Figure 10: Traffic 2 in the original graph.

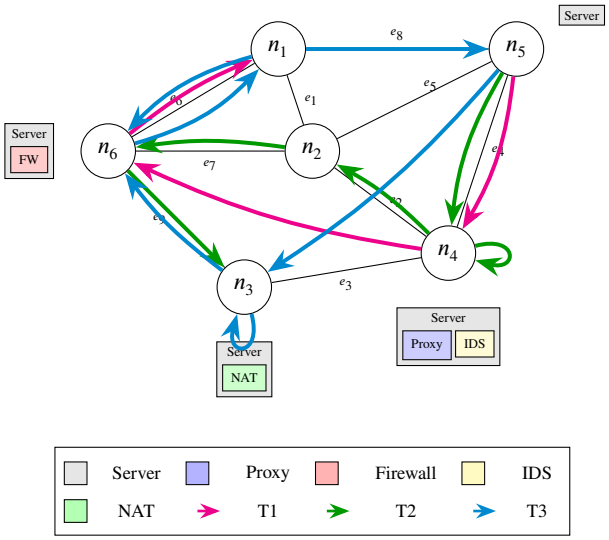


Figure 11: Final resulting graph.

6.4. Constraint Handling and Feasibility

A critical aspect of many-objective optimization is the treatment of constraints, which in the VNF-P problem are governed by Equations (23) through (29) (covering physical node and link capacity constraints such as CPU, memory, storage, bandwidth, and maximum latency limits). Rather than employing traditional exterior penalty functions—which can severely distort the objective space and hinder convergence in many-objective environments—this work implements a hybrid of *Constrained Encoding* and an active *Repair Mechanism* integrated directly within the multi-stage decoding process.

Specifically, the following three mechanisms are applied:

1. **Constrained Encoding (Active Filtering):** During the construction of the virtual multi-stage graph for each traffic request (defined in Section 6-B), candidate physical nodes and links are filtered actively. A physical server is only included as a candidate in stage E_i if it has sufficient residual CPU, memory, and storage capacity to host the j-th VNF. Similarly, physical links are only included as candidate transitions if their available bandwidth meets the request demand. This prevents the generation of structurally invalid individuals.

2. **Active Repair and Backtracking:** When selecting a path across the multi-stage graph, if a stage is reached where no candidate servers can satisfy the resource constraints, or if the linking physical paths violate the accumulated latency limit, the algorithm triggers a backtracking step. This mechanism undoes the placement in the previous stage E_{i-1} and attempts to route through the next best-ranked feasible candidate.

3. **Feasible Space Projection:** If all combinations are exhausted and a traffic request cannot be routed under the strict constraints of physical nodes and links, the request is flagged as unplaced (contributing to a penalty in the throughput and Service Level Objective (SLO) cost objectives), but the chromosome's structure remains mathematically valid. For example, in highly congested scenarios such as the `INDIA35` topology with `Traffic_600` or the `Zib54` topology with `Traffic_3000` (Saturated loads), empirical observations indicate an infeasibility (unplaced request) rate of approximately 10% to 25% even after the repair mechanism is exhausted. By simply dropping these requests rather than creating invalid configurations, the structure guarantees that the decoded physical mapping strictly complies with network capacity.

Through this hybrid approach, every chromosome decoded always maps to a feasible execution state in the physical network without requiring external penalty factors, ensuring that the evolutionary search is restricted to the strictly feasible region of the solution space.

7. Pareto Performance Metrics

In this study, metrics were utilized to measure the characteristics of the known Pareto front (*PFK*) relative to a true Pareto front (*PFT*). Since no single metric can capture all characteristics, they are briefly explained below [45, 50, 51]:

- **Generational Distance (GD):** GD is an indicator that reports how far, on average, *PFK* is from *PFT*. This indicator can be mathematically defined as:

$$GD \triangleq \frac{1}{H} \cdot \left(\sum_{f \in PFK} (dist(f, PFT))^p \right)^{\frac{1}{p}} \tag{30}$$

where:

  - H is the number of vectors in *PFK*,
  - *dist(f, PFT)* is the phenotypic distance between the vector $f \in PFK$ and *PFT*, that is:

$$\text{dist}(f, PFT) = \min_{f^* \in PFT} \{dist(f, f^*)\} \tag{31}$$

If GD = 0, then $PFK = PFT$. In this study, we consider $p = 2$ in accordance with [52], thus the distance is Euclidean.

- **Inverted Generational Distance (IGD):** This indicator measures how far, on average, *PFT* is from *PFK*. In this study, the following indicators were utilized, mathematically defined as:

$$IGD \triangleq \frac{1}{C} \cdot \left(\sum_{f^* \in PFT} \left(dist(f^*, PFK)\right)^p \right)^{\frac{1}{p}} \tag{32}$$

where C is the number of vectors in *PFT* and $p = 1$ in accordance with [52]. If IGD = 0, then $PFK = PFT$.

- **Epsilon Indicator (I_ϵ):** Given *PFK* and *PFT*, the additive I_ϵ defines the minimum value $\epsilon \in \mathbb{R}$ such that any vector in *PFT* is ϵ-dominated by at least one solution in *PFK*.

$$\begin{aligned} I_\epsilon = min\{\epsilon \in \mathbb{R} | \forall f^* \in FPT, \exists f \in FPK \\ \texttt{such that } \forall i \in \{1, \dots, \alpha\}, f_i^* \leq f_i + \epsilon\} \end{aligned} \tag{33}$$

where α indicates the number of objective functions of the problem. If $I_\epsilon > 0$, then *PFK* must be moved by at least ϵ units to completely dominate *PFT*. If $I_\epsilon = 0$, then every solution in *PFT* is dominated or equaled by some solution in *PFK*.

- **Maximum Pareto Front Error (ME):** This metric measures how well one set of vectors compares with another. This metric can be mathematically expressed as:

$$ME \triangleq \max_{f \in PFK} \left\{ \min_{f^* \in PFT} \left\{ dist(f, f^*) \right\} \right\} \tag{34}$$

More specifically, it measures the largest minimum distance between each vector in *PFK* and the corresponding closest vector in *PFT*.

- **Spacing (S):** This metric numerically describes the spread of the vectors in PFK, which can be mathematically described as:

$$S \triangleq \left(\frac{1}{H-1} \sum_{f \in PFK} \left(dist(f, PFK) - \overline{dist} \right)^p \right)^{\frac{1}{p}} \tag{35}$$

where

  - $dist(f, PFK)$ is the closest distance from vector f to the set *PFK*-$\{f\}$ whose structure is identical to Equation (31).
  - $\overline{dist}$ is the average distance of the vectors, that is: $\overline{dist} = \frac{1}{H} \sum_{f \in PFK} dist(f, PFK)$.

When $S = 0$, all vectors are evenly spaced. As the uniformity deteriorates, S increases. In this study $p = 2$ in accordance with [52].

- R_k *Indicator*: Given that the exact hypervolume calculation is #P-hard in general [53], and even practical state-of-the-art sweeping algorithms (such as WFG or those based on Klee's measure problem [54, 55]) exhibit a worst-case time complexity bounded by $\mathcal{O}(\eta \log \eta + \eta^{\alpha/2})$ for $\alpha \geq 3$ (where η is the number of solutions and α is the number of objective functions), R_k indicators emerge as alternatives. These indicators constitute a family of scalar metrics. The R_k indicators have linear complexity in the number of directions and the size of the PFK set ($\mathcal{O}(K \cdot |PFK| \cdot \alpha)$).

  These indicators are based on the concept of aggregate utility functions applied to a collection of preference directions uniformly distributed over the simplex. Each variant represents a different way of evaluating the utility of the set, allowing for flexible and computationally efficient assessment. The R_k indicator can be mathematically expressed as:

$$R_k = \frac{1}{|PFK|} \cdot \sum_{\lambda^j \in \Lambda} \min_{f \in PFK} u_k(f, \lambda^j) \tag{36}$$

where:

  - Λ is a set of weight vectors uniformly generated over the simplex $\Delta^{\alpha-1}$, that is: $\Lambda = \{\lambda^{(1)}, \dots, \lambda^{(K)}\}$ with $\lambda^{(j)} \in \mathbb{R}^\alpha$, $\sum_{i=1}^{\alpha} \lambda_i^{(j)} = 1$, $\lambda_i^{(j)} \geq 1$.
  - $u_k(f, \lambda^j)$ is a scalarizing function that reduces the vector f to a numerical value.

In this context, the literature proposes three R_k-type indicators: R_1, R_2, and R_3. These quality indicators are explained below.

  - R_1 *Indicator:* This indicator utilizes an additive utility function (Linear Weighted Sum) that averages the performance of the PFK under multiple linear preferences, making it suitable for convex problems. This utility can be mathematically expressed as:

$$u_1(f, \lambda^j) = \sum_{i=1}^{\alpha} \lambda_i^j \cdot f_i \tag{37}$$

  - R_2 *Indicator:* Introduces a Chebyshev-type utility function (Weighted Tchebycheff) relative to an ideal point, emphasizing the worst weighted deviation. Mathematically, this utility can be expressed as:

$$u_2(f, \lambda^j) = \max_{1 \leq i \leq \alpha} \{\lambda_i^j \cdot |f_i - z_i^*|\} \tag{38}$$

where $z^* = \{z_1^*, \ldots, z_\alpha^*\}$ is a reference or ideal point. This indicator is more sensitive to vectors with poor performance in at least one objective, and is more robust against non-convex fronts.

– *R3 Indicator:* This indicator evaluates the degree of coverage of the set with respect to each preference direction, utilizing an inverse and normalized function. This is expressed mathematically as:

$$u(\lambda^j, PFK) = \max_{1 \le i \le \alpha} \left[\frac{1}{\lambda_i^j} \cdot \frac{f_i - z_i^*}{z_i^{nad} - z_i^*} \right] \tag{39}$$

where $z^{nad} = (z_1^{nad}, \ldots, z_\alpha^{nad})$ is the nadir point. This variant is particularly useful for capturing both the convergence and diversity of the vectors.

Table 12: Traffic Load Classification

Traffic Load	Effective Transfer Rate
Low	75 - 100%
Medium	45 - 74%
High	20 - 44%
Saturated	0 - 19%

8. Computational Experiments

The simulations conducted in this section seek to answer two questions: (1) Which objective functions should be considered in the problem as a MaOP, and (2) What is the performance of the implemented MaOEAs? In this context, we initially present the computational environment utilized for the tests; subsequently, tests are conducted to determine the correlation between objective functions and consequently discard redundant ones. Finally, simulations are performed in order to compare the performance of the implemented MaOEAs.

8.1. Computational Environment

The computational simulations were carried out in a controlled environment, utilizing a Lenovo Legion Y540-15IRH laptop, equipped with an Intel Core i7-9750H processor (6-core, 12-thread architecture, with a 2.6 GHz base frequency and 4.5 GHz maximum turbo frequency), 16 GB of DDR4 RAM, and a 1 TB Solid State Drive (SSD) as primary storage. The operating system employed was Microsoft Windows 10. The evolutionary algorithms were implemented using the MOEAFramework library in Java [52]. Each execution was performed sequentially on a single thread, without employing multi-core parallelism, to guarantee result reproducibility and avoid variations associated with concurrency. These specifications allowed for the efficient execution of multiple runs of the evolutionary algorithms without incurring significant hardware limitations or performance penalties during the evaluation, selection, and solution generation processes. It is worth noting that the same evolutionary parameters previously defined in the corresponding section were maintained, with the aim of preserving methodological homogeneity and ensuring the comparability of the results obtained across the different experimental scenarios.

8.2. Objective Function Selection

Feature selection, as a dimensionality reduction technique, aims to select a subset of relevant features from the original ones by eliminating redundant features. This generally leads to an improved performance of the evolutionary process, that is, better Pareto front quality and lower computational cost [56]. In this study, the elimination of redundant objective functions was considered given that all are deemed relevant.

To construct the vector set F, we first evaluate the solutions obtained across preliminary simulations utilizing NSGA-III over the 12 scenarios. Data vectors of the 17 metrics are collected at the final generation, obtaining a substantial sample of 2400 observations (100 solutions per 24 scenarios). The first step calculates the Pearson correlation (r) between pairs of functions. Additionally, the coefficient of determination (r^2) is utilized to quantify the level of redundancy, indicating what percentage of the variance of one function is explained by the other. If two pairs of functions f_a and f_b have a correlation equal to or greater than 80% ($r^2 > 0.64$), one of them is a candidate to be eliminated. A sensitivity analysis revealed that thresholds of 75%, 80%, and 85% produce exactly the same subset of $M = 11$ objective functions, demonstrating robustness. Lowering the threshold to 70% is too aggressive, discarding additional functions where more than half of their variance is independent information, whereas raising it to 90% retains too many redundant objectives, failing to alleviate the many-objective complexity [57, 12]. Furthermore, discarding decisions are not merely statistical; domain knowledge plays a crucial role. For instance, evaluating performance metrics and navigating the search space in 17 dimensions is mathematically complex and computationally expensive, often exacerbating the curse of dimensionality. Specifically, in such high-dimensional spaces, the proportion of non-dominated solutions grows exponentially, leading to *dominance resistance*, where traditional Pareto-based selection completely loses selection pressure. While the many-objective algorithms utilized in this work (e.g., MOEA/D and NSGA-III) are explicitly designed to cope with dominance resistance by relying on reference-based decomposition and scalarization instead of pure Pareto dominance, discarding highly correlated, domain-redundant objectives still critically alleviates computational bottlenecks and preserves the physical significance of the optimization process.

Tables 13 and 14 summarize the sensitivity analysis for objective reduction. The thresholds 75%, 80%, and 85% yield an identical subset of $M = 11$ objective functions, indicating stability within a range of ±5 percentage points around the 80% threshold. Lowering the threshold to 70% discards f_{10} and f_{12}, corresponding to the CAPEX and distance coverage. On the other hand, a threshold of 90% retains 14 objectives, providing an insufficient reduction. Therefore, the 80% threshold provides an adequate balance between removing redundant functions and preserving the fundamental criteria of the problem.

8.2.1. Test Instances and Experimental Scheme

The objective function selection procedure was carried out through the following steps:

- A set of $N_{esc} = 24$ scenarios is considered.

Table 13: Discard Table - Complete Sensitivity Analysis

Selected Function	Discarded Function	r	Minimum Threshold
$\|r\| \geq 90\%$			
f_7 Resource fragmentation	f_{14} Number of servers	99%	90%
f_2 Bandwidth consumption	f_8 Link utilization cost	99%	90%
f_{16} Effective transfer rate	f_{17} Percentage of allocated VNFs	94%	90%
$85\% \leq \|r\| < 90\%$			
f_{10} License cost	f_5 Installation cost	88%	85%
f_1 Energy cost	f_{14} Number of servers	87%	85%
f_{12} Distance traveled	f_{13} Number of hops	86%	85%
f_1 Energy cost	f_6 Total resource cost	86%	85%
$80\% \leq \|r\| < 85\%$ **– chosen threshold in the thesis**			
f_{10} License cost	f_6 Total resource cost	84%	80%
$75\% \leq \|r\| < 80\%$ **– no new pair with both functions active**			
f_3 Latency	f_{13} *Number of hops (already discarded)*	79%	–
f_5 *Installation cost (already discarded)*	f_6 *Total resource cost (already discarded)*	78%	–
$70\% \leq \|r\| < 75\%$ **– pairs with both functions active at 80%**			
f_1 Energy cost	f_{10} License cost	74%	70%
f_3 Latency	f_{12} Distance traveled	73%	70%
f_2 Bandwidth consumption	f_{13} *Number of hops (already discarded)*	72%	–
f_8 *Link utilization cost (already discarded)*	f_{13} *Number of hops (already discarded)*	71%	–
f_{10} License cost	f_{14} *Number of servers (already discarded)*	71%	–

Table 14: Sensitivity Analysis Summary for Objective Reduction

Threshold	Discarded	Retained (M)	Discarded Functions
90%	3	14	f_8, f_{14}, f_{17}
85%	6	11	$f_5, f_6, f_8, f_{12}, f_{14}, f_{17}$
80% – chosen	**6**	**11**	$f_5, f_6, f_8, f_{12}, f_{14}, f_{17}$
75%	6	11	$f_5, f_6, f_8, f_{12}, f_{14}, f_{17}$
70%	8	9	$f_5, f_6, f_8, f_{10}, f_{12}, f_{13}, f_{14}, f_{17}$

- Each scenario i is configured according to a network topology, a traffic load (with a set of requests in an all-to-all scheme), and the blocking type.
- For each scenario i, $N_{sol} = 100$ consistent solutions $\Theta_i = \{\Theta_{i1}, \ldots, \Theta_{iN_{sol}}\}$ are calculated.
- For each solution Θ_{ij}, the objective function vector $F = \{f_1, \ldots, f_M\}$ is calculated.
- With the $N_{esc} \cdot N_{sol}$ solutions Θ_{ij}, the Pearson correlation is computed for each pair (f_a, f_b) of objective functions.

The experimental tests were conducted on three network topologies with distinct characteristics regarding the number of nodes and links: NSF [58], depicted in Figure 12; EON-RT [59], in Figure 13; and INDIA35 [60], in Figure 14. The links in these topologies present an average bandwidth varying between 1000 and 1200 Mbps, while the servers feature, on average, 64 CPU cores, 128 GB of RAM, and 1200 GB of storage.

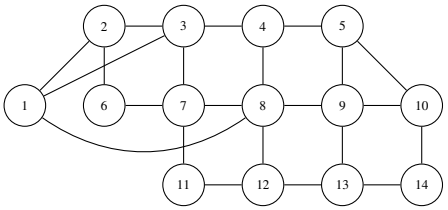


Figure 12: NSF Topology

In each topology, multiple executions were performed utilizing different traffic request loads, classified into four levels: Low, Medium, High, and Saturated. These loads were designed to force scenarios with blockages in links and/or servers, allowing the evaluation of the system's behavior under congestion conditions.

In the experiment, two types of blocking were considered:

- Node blocking: the available resources in the links were considerably increased to prevent rejections due to lack of capacity in them, ensuring that request blockages were exclusively caused by insufficient server resources.
- Link blocking: the computational resources of the servers (CPU, RAM, and storage) were increased so that any request rejection was due solely to the lack of capacity in the links.

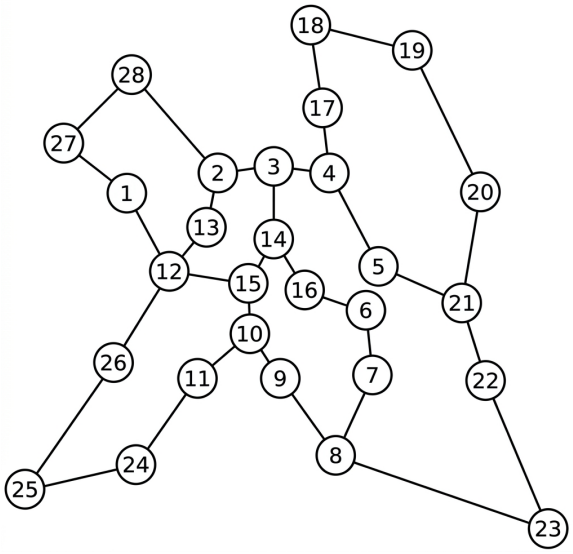


Figure 13: EON-RT Topology

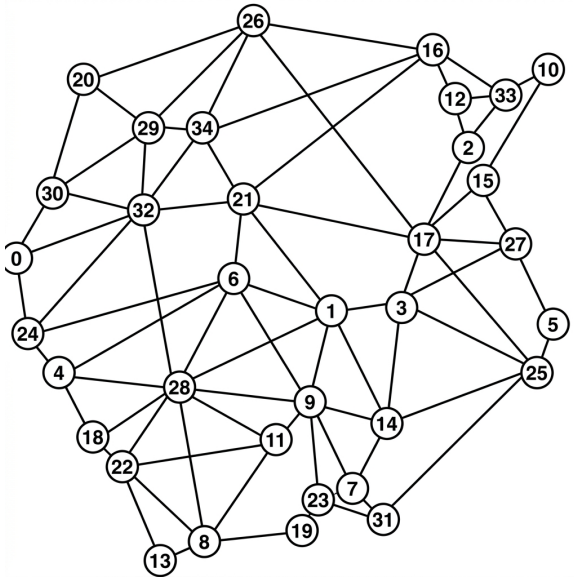


Figure 14: INDIA35 Topology

Table 15 summarizes all the test scenarios, classified according to the blocking type, the topology utilized, the load level applied in each execution, and the average effective transfer rate (throughput) corresponding to each load for each blockage. The bandwidth and number of VNFs columns are explained below. To adequately configure the test conditions, the following adjustments were made to the traffic requests:

- In the case of link blocking, the initial required bandwidth was modified in order to induce an average rejection rate that permitted classifying the load into the aforementioned levels.
- In the case of node blocking, the number of VNFs per request was adjusted to cause rejections due to a lack of resources in the servers, also allowing their classification into Low, Medium, High, and Saturated.

For each of the topologies, a set of traffic requests was generated following an all-to-all scheme, that is, from all nodes to all other nodes in the network. Under this scheme, the NSF topology generated 182 requests, while EON-RT and INDIA35 produced 756 and 1190 requests, respectively. These experiments identified the loads corresponding to the four congestion levels for both types of blocking.

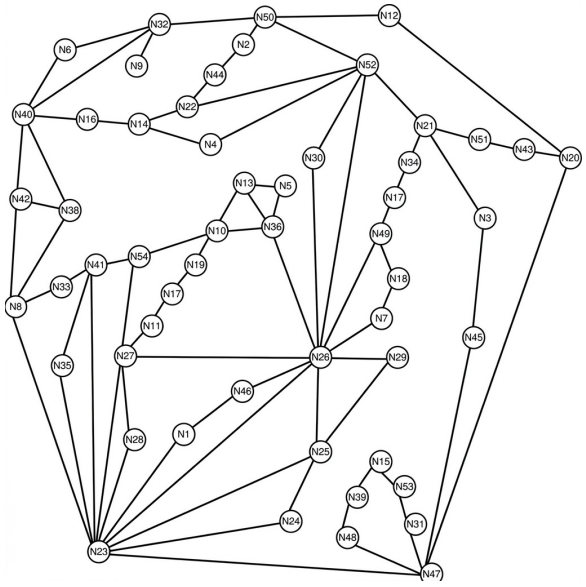


Figure 15: Zib54 Topology

8.2.2. Results and Discussions

Table 16 presents the results of the correlation analysis between pairs of objective functions. If the correlation is greater than 80% between two objective functions, it implies low contradiction, whereby one of the functions is eliminated to avoid redundancy.

Finally, Table 17 displays the comparisons made between highly correlated objective functions.

8.3. Objective Functions Discarding Justification

To reduce the dimensionality of the Many-Objective Problem (MaOP) and avoid redundancy, highly correlated objective functions (over 80% correlation) were analyzed to determine which should be retained and which discarded, as detailed in Table 17. The elimination decisions were strictly guided by the physical meaning and operational relevance of each objective within the NFV environment:

- **Installation Cost (f_5):** It presented an 88% correlation with License Cost (f_{10}). f_{10} was retained because license fees are recurring and more representative of the continuous software-based nature of VNFs, whereas the installation is usually a one-time deployment expense.
- **Total Resource Cost (f_6):** It was highly correlated with both Energy Cost (f_1, 86%) and License Cost (f_{10}, 84%). Since f_1 and f_{10} already provide a detailed account of the operational and software expenditures respectively, maintaining f_6

Table 15: Test Scenario Configuration Table

		BANDWIDTH BLOCKING		NODE BLOCKING	
Network	**Load**	**Throughput**	**Bandwidth (Mb/s)**	**Throughput**	**Number of VNFs**
NSF	Low	89.005	25	89.65	2
	Medium	52.19	65	46.115	4
	High	19.5	200	18.385	10
	Saturated	7.965	600	6.455	25
EON-RT	Low	93.645	4	93.675	1
	Medium	61.7	9	49.765	2
	High	19.77	50	19.315	5
	Saturated	6.415	150	7.62	11
INDIA35	Low	82.835	15	79.505	1
	Medium	57.515	25	41.74	2
	High	15.625	150	16.345	5
	Saturated	7.775	300	7.605	10

Table 16: Pearson correlations between objective functions.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}
f_1		41%	57%	42%	66%	86%	-8%	41%	7%	74%	3%	41%	48%	87%	18%	-52%	-53%
f_2			61%	43%	19%	30%	-52%	99%	26%	19%	12%	62%	72%	29%	10%	-44%	-43%
f_3				65%	28%	55%	-34%	61%	17%	28%	14%	73%	79%	40%	19%	-83%	-88%
f_4					25%	42%	-19%	43%	13%	24%	22%	33%	39%	35%	17%	-69%	-73%
f_5						78%	-2%	19%	1%	88%	-4%	17%	18%	47%	32%	-25%	-27%
f_6							-13%	30%	7%	84%	0%	32%	36%	48%	28%	-53%	-56%
f_7								-51%	-15%	5%	-4%	-32%	-39%	99%	15%	23%	23%
f_8									25%	19%	12%	61%	71%	28%	10%	-44%	-43%
f_9										1%	2%	14%	18%	7%	2%	-11%	-11%
f_{10}											-2%	20%	21%	71%	32%	-27%	-29%
f_{11}												14%	15%	3%	1%	-10%	-11%
f_{12}													86%	24%	12%	-43%	-45%
f_{13}														27%	11%	-52%	-54%
f_{14}															48%	-41%	-41%
f_{15}																-18%	-20%
f_{16}																	94%
f_{17}																	

would introduce redundancy in OPEX evaluation. Therefore, f_6 was discarded.

- **Utilization Cost of All Network Links (f_8):** It showed an almost perfect correlation (99%) with Bandwidth Consumption (f_2). f_2 provides a more direct and universally understood metric of network traffic volume in NFV, making f_8 expendable.
- **Number of Hops (f_{13}):** It exhibited an 86% correlation with Distance Traveled (f_{12}). While hops represent logical distances, f_{12} reflects the physical geographic propagation distance which has a more direct impact on signal latency. Thus, f_{13} was discarded.
- **Number of Servers (f_{14}):** It showed high correlations with Energy Cost (f_1, 87%) and Resource Fragmentation (f_7, 99%). Energy consumption (f_1) is inherently driven by the number of active servers, making f_{14} redundant. Retaining f_1 and f_7 better captures both the environmental impact and the efficiency of server utilization.
- **Percentage of Placed VNFs (f_{17}):** It correlated strongly (94%) with Network Effective Transfer Rate (f_{16}, Throughput). In service provider environments, throughput (f_{16}) is a much more critical performance indicator for user satisfaction and Quality of Service (QoS) than the mere percentage of successfully instantiated VNFs. Consequently, f_{17} was eliminated.

As a result of this analysis, 11 relevant objective functions were selected, namely: Energy Cost (f_1), Bandwidth Consumption (f_2), Latency (f_3), Load Traffic (f_4), Resource Fragmentation (f_7), Maximum utilized network link (f_9), License Cost (f_{10}), SLO Cost (f_{11}), Distance Traveled (f_{12}), Number of VNF instances (f_{15}), and Network effective transfer rate (f_{16}).

Table 17: Discard Table

O.F.1	O.F.2	Correlation %	Discarded O.F.
f_{10} License cost	f_5 Installation cost	88%	f_5 Installation cost
f_1 Energy cost	f_{14} Number of servers	87%	f_{14} Number of servers
f_7 Resource fragmentation	f_{14} Number of servers	99%	f_{14} Number of servers
f_2 Bandwidth consumption	f_8 Utilization cost of all network links	99%	f_8 Utilization cost of all network links
f_{12} Distance traveled	f_{13} Number of hops	86%	f_{13} Number of hops
f_{16} Network effective transfer rate	f_{17} Percentage of placed VNFs	94%	f_{17} Percentage of placed VNFs
f_1 Energy cost	f_6 Total resource cost	86%	f_6 Total resource cost
f_{10} License cost	f_6 Total resource cost	84%	f_6 Total resource cost

8.4. MaOEA Performance Analysis

These simulations were conducted considering the eleven objective functions selected in the previous section, assessing three MaOEAs: R-VEA [61], MOEA/D [62], and NSGA-III [63]. These MaOEAs are widely recognized for their efficacy in MaOPs.

8.4.1. Evolutionary Parameters

The configuration applied to the MaOEAs included the type of genetic operators, population size, and stopping criteria, selected to ensure a fair comparison among them.

Genetic Operators:
All algorithms employed Polynomial Mutation [47] as the mutation operator, with a mutation probability of $p_m = 1/L$ (where L is the number of decision variables) and a distribution index of $\eta_m = 20.0$. Regarding the crossover operator, R-VEA and NSGA-III applied the Simulated Binary Crossover (SBX) [64] operator with a crossover probability of $p_c = 1.0$ and a distribution index of $\eta_c = 30.0$, whereas MOEA/D utilized the Differential Evolution (DE) [65] operator with a crossover rate of $CR = 1.0$ and a scaling factor of $F = 0.5$. The relatively high SBX distribution index ($\eta_c = 30$) controls the dispersion of offspring, generating children closer to their parents to favor finer local search [63].
In all three algorithms, parent selection was performed randomly, aiming to preserve the genetic diversity of the population during the evolutionary process.

Population Size:

To maintain homogeneous experimental conditions, the population size was fixed at 90 individuals for the three considered algorithms (R-VEA, MOEA/D, and NSGA-III). In the cases of R-VEA and NSGA-III, this value emerges naturally from the generation of reference points utilizing outer divisions = 2 and inner divisions = 1, according to the MOEAFramework implementation and following the provisions of their respective original works [61], [63].

Let α be the number of objectives and H the number of divisions. The number of reference points generated by a layer is obtained by:

$$(\eta(\alpha, H) = \binom{\alpha + H - 1}{H} = \binom{\alpha + H - 1}{\alpha - 1}). \tag{40}$$

When two layers are employed ($H_{outer} = 2$ and $H_{inner} = 1$), the target population size is calculated as:

$$(\eta = \eta(\alpha, H_{outer}) + \eta(\alpha, H_{inner})) \tag{41}$$

$$(\eta = \binom{\alpha + H_{outer} - 1}{H_{outer}} + \binom{\alpha + H_{inner} - 1}{H_{inner}}). \tag{42}$$

Particularly, for $\alpha = 12$ objectives, we have:

$$(\eta_1 = \binom{12 + 2 - 1}{2} = \binom{13}{2} = 78), \tag{43}$$

$$(\eta_2 = \binom{12 + 1 - 1}{1} = \binom{12}{1} = 12), \tag{44}$$

which results in a total population size of $\eta = 90$.

Both R-VEA and NSGA-III employ this same combinatorial scheme, although in R-VEA it is often expressed in terms of $\alpha - 1$ instead of H, an equivalent notation in the literature. In the case of MOEA/D, the MOEAFramework defaults to a population of 100 individuals. However, in this work, it was adjusted to 90 to ensure a fair comparison.

Stopping Criterion:
To control the execution duration and allow a fair comparison among the MaOEAs, the simulation adopts the maximum number of generations fixed at $MaxGen = 300$ as the stopping criterion, which is equivalent to $NFE = P_o \times MaxGen = 90 \times 300 = 27,000$ Number of Function Evaluations per execution. This was determined from a preliminary experiment aimed at analyzing the convergence behavior of the NSGA-III algorithm. In that experiment, the Zib54 network topology with 2000 traffic requests was utilized, along with Generational Distance (GD) as the metric recorded every 50 generations. The results showed that starting from generation 250-300, the GD value stabilized without significant improvements.

8.4.2. Test Instances

For the test instances, the network topologies Zib54 [66] depicted in Figure 15, EON-RT [59] in Figure 13, and INDIA35 [60] in Figure 14 were utilized. The number of requests corresponding to the Low, Medium, High, and Saturated load levels was calculated previously for each topology, following the scale defined in Table 12. These values are detailed in Table 18. In total, twelve experimental scenarios were defined, in which traffic requests were generated randomly. Recall that each request is composed of the

following tuple $t = (o_t, \omega_t, \beta_t, \Delta_t, \rho C_t, W_t)$. In this regard, each component was selected according to the following schemes:

- o_t = traffic origin is randomly generated among all network nodes.
- ω_t = traffic destination is randomly generated among all network nodes.
- β_t = minimum traffic bandwidth at the source node is generated in a range between two and five Mbps [40].
- Δ_t = maximum allowable delay is equal to twice the delay from source to destination under low traffic load.
- ρC_t = penalty cost in dollars for exceeding the delay declared in the SLA is between one and two dollars.
- W_t = number of VNFs per traffic request is generated between two and four VNFs.

Considering these criteria, we proceed to perform the simulations explained in the following subsection.

Table 18: Test Scenarios

Topology	Load	# Requests
Zib54	Low	600
Zib54	Medium	1000
Zib54	High	2000
Zib54	Saturated	3000
India35	Low	400
India35	Medium	600
India35	High	1000
India35	Saturated	2000
EON	Low	300
EON	Medium	500
EON	High	980
EON	Saturated	1500

8.4.3. Experimental Scheme

The experimental scheme is the overall process for performing evolutionary simulations, obtaining Pareto fronts, and calculating performance metrics. The scheme is explained below:

- A set of $N_{esc} = 12$ scenarios was defined, each determined by a combination of a network topology and a traffic load level.
- For each scenario, a number $\mathcal{T}$ of traffic requests was generated according to the corresponding load, see Table 18.
- Each MaOEA was executed independently for each scenario.
- To mitigate the effect of stochastic variability inherent to evolutionary algorithms and obtain statistically representative estimations, each algorithm was executed $N_{ej} = 30$ times per scenario independently using different random seeds, as recommended in the literature for reliable estimation of average performance and variability. Consequently, the total number of executions performed in the study was $N_{ej} \times N_{alg} \times N_{esc} = 30 \times 3 \times 12 = 1,080$ independent executions.
- As a result, $N_{ej} = 30$ Pareto fronts were obtained for each scenario and algorithm combination.
- To evaluate algorithm performance, the quality metrics defined in Section 7 were applied to the Pareto fronts obtained in each execution.

At the end of these steps, statistical calculations were performed presenting the performance of the MaOEAs according to the traffic scenario and quality metrics. The figures and their discussions are provided in the following subsection.

8.4.4. Results and Discussion

Before presenting Figure 16, recall that the value of each cell, for each topology and load combination (*low, medium, high, and saturated*), corresponds to the average over $N_{ej} = 30$ Pareto fronts calculated in that scenario, as mentioned in the previous section.

Figure 16 summarizes the performance of the multi-objective algorithms by topology and load, reporting quality metrics (GD, IGD, I_ϵ, ME, S, R_1, R_2, R_3). The horizontal blocks correspond to the topologies and the vertical blocks to the different traffic load levels (low, medium, high, and saturated). Each block displays subcolumns corresponding to the evaluated algorithms (NSGA-III, MOEA/D, and R-VEA), and the rows indicate the performance metrics (GD, IGD, ME, S, R1, R2, and R3) employed to compare the results.

Each cell shows the *average value* of the metric from $N_{ej} = 30$ Pareto fronts generated in that same scenario (same topology and same load).

Except for R_1, which is maximized, all metrics (GD, IGD, I_ϵ, ME, S, R_2, and R_3) are minimized: lower values are better.

The shaded cells mark the *best* value within the corresponding scenario (same topology and load), comparing algorithms. For instance, if in the `ZIB54` topology and *saturated load* the IGD cell for `NSGA-III` displays $3.234E-01$ and is shaded, it indicates that the average (over $N_{ej} = 30$ fronts) IGD obtained by `NSGA-III` in that scenario is $3.234E-01$, and that this value is the best (lowest) compared to the other evaluated algorithms in `ZIB54` with *saturated load*.

Table 19 presents a detailed numerical comparison of the three evaluated MaOEAs—NSGA-III, MOEA/D, and R-VEA—across the three network topologies (ZIB54, INDIA35, and EON-RT) and four traffic load levels (Low, Medium, High, and Saturated). Shaded cells indicate the best-performing algorithm for each topology-load combination according to each metric.

Several patterns emerge from this analysis. First, **R-VEA dominates GD and ME** across virtually all scenarios: it achieves a GD of 0 (perfect convergence) in nearly every ZIB54 and INDIA35 scenario, indicating that its approximated Pareto front contains at least one solution that coincides with the reference front. A similar

Table 19: Performance metrics comparison across topologies and traffic loads. Shaded cells indicate the best-performing algorithm per scenario.

Topology	Metric	Low Load			Medium Load			High Load			Saturated Load		
		NSGA-III	MOEA/D	R-VEA	NSGA-III	MOEA/D	R-VEA	NSGA-III	MOEA/D	R-VEA	NSGA-III	MOEA/D	R-VEA
ZIB54	GD	0.0138	0.0134	0	0.0156	0.0174	0	0.0157	0.0166	0	0.0156	0.0172	0
	IGD	0.2976	0.3345	0.6642	0.3210	0.3628	0.7401	0.3140	0.3613	0.6967	0.3234	0.3769	0.7815
	I_ϵ	0.3505	0.3528	0.6948	0.3515	0.4059	0.7052	0.3554	0.3974	0.6969	0.3352	0.4383	0.7717
	ME	0.3111	0.2644	0	0.3512	0.3056	0	0.3375	0.2870	0	0.3339	0.3087	0
	S	30622	38850	0	39744	45838	0	30087	55247	2344	31652	44088	0
	R_1	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03
	R_2	1518702	1490606	1492719	1513905	1490978	1490271	1500451	1472352	1461464	1498792	1472926	1471190
INDIA35	GD	0.0174	0.0163	0	0.0170	0.0175	0	0.0174	0.0185	0	0.0162	0.0182	0
	IGD	0.3204	0.3518	0.6753	0.3203	0.3528	0.6696	0.3404	0.3812	0.6919	0.3323	0.3739	0.7143
	I_ϵ	0.3429	0.3903	0.7628	0.4030	0.4283	0.7944	0.3820	0.4223	0.9288	0.3649	0.4709	0.7787
	ME	0.3445	0.2952	0	0.3457	0.3062	0	0.3776	0.3168	0	0.3593	0.3136	0
	S	4166	4357	4658	3941	5519	0	3380	4903	511.9540	3923	5319	1199
	R_1	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03
	R_2	104874	103873	104331	103026	101338	101567	104731	102245	102837	105164	102937	103254
EON-RT	GD	1.69e-03	0	0	3.14e-03	0	0	2.65e-03	0	0	0	0	0
	IGD	0.2753	0.3154	0.6516	0.2743	0.3180	0.6945	0.2811	0.3286	0.6300	0.2910	0.3449	0.7130
	I_ϵ	0.3848	0.5127	0.7789	0.3193	0.4171	0.6997	0.3867	0.4000	0.6494	0.3320	0.4255	0.7621
	ME	0.1446	0	0	0.2078	0	0	0.1906	0	0	0	0	0
	S	2111	2943	1493	2255	3069	2853	2467	3502	1853	2634	2838	1264
	R_1	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03	1.37e-03
	R_2	59650	58344	58709	57086	55200	55418	58089	56528	57564	59824	58029	58261

trend is observed for the Maximum Pareto Error (ME), where R-VEA systematically achieves ME= 0, confirming that no member of its front lies significantly far from the reference. This behavior is expected, as R-VEA's angle-penalized distance mechanism is specifically designed to maintain reference-point convergence.

Second, **NSGA-III consistently achieves the best IGD and I_ϵ** values across all topologies and loads. For example, in ZIB54 under low load, NSGA-III obtains IGD= 0.2976 compared to MOEA/D's 0.3345 and R-VEA's 0.6642. This advantage holds across all four load levels and all three topologies, indicating that NSGA-III generates a more uniformly spread approximation front that covers the objective space more comprehensively. The consistently higher IGD and I_ϵ of R-VEA suggests that, while it achieves good pointwise convergence (low GD), its front is less diverse and covers fewer regions of the Pareto front.

Third, **behavior under the Spacing (S) metric is topology-dependent**. In ZIB54 (a large backbone network), R-VEA achieves the best spacing in most scenarios, producing a more evenly distributed front. In INDIA35 (a medium-sized national topology), results are more balanced: NSGA-III leads under low load, while R-VEA leads under medium and saturated conditions. In EON-RT (a compact optical network), R-VEA and NSGA-III alternate as the best performers, suggesting that topology size and connectivity density influence which algorithm achieves better solution spacing.

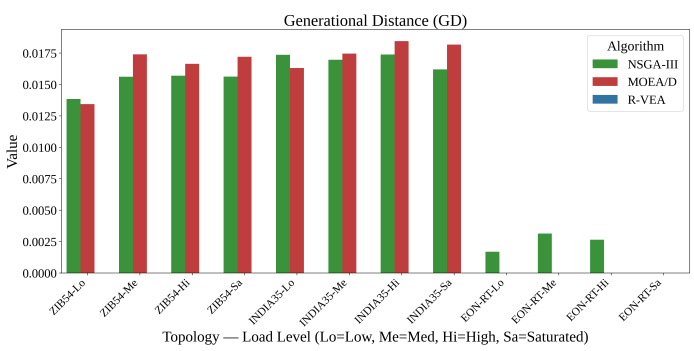
Fourth, **the R_2 indicator—a weighted utility metric—favors MOEA/D or R-VEA** depending on the topology. In ZIB54, MOEA/D obtains the best R_2 under low load, while R-VEA leads under high and saturated loads. In INDIA35 and EON-RT, MOEA/D consistently obtains the lowest R_2 values, suggesting better alignment with the reference weight vectors in smaller, more structured topologies. The R_1 indicator, which is identical across all algorithms in all scenarios ($\approx 1.37 \times 10^{-3}$), reflects that all three algorithms produce fronts of comparable quality when measured by this particular weighted indicator, and therefore does not discriminate between them.

Overall, the results in Table 19 confirm the complementary nature of the three algorithms: R-VEA excels in convergence (GD, ME), NSGA-III excels in coverage and diversity (IGD, I_ϵ), and MOEA/D achieves competitive R_2 performance especially on smaller topologies. These findings are consistent with the aggregate best-count analysis presented in subsequent figures.

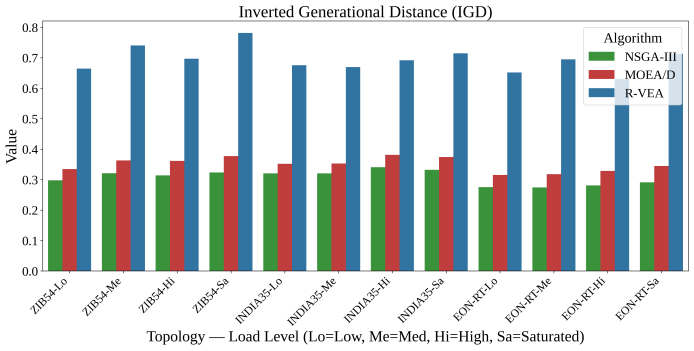
Figure 17 summarizes what is detailed in Figure 16. Figure 17 takes those values cell-by-cell and, for each metric, counts in how many scenarios each algorithm achieved the best result. It is observed that R-VEA maintains the best values in GD and ME, confirming better convergence capacity, whereas NSGA-III achieves the best results in IGD and Iε, evidencing superior convergence toward the optimal Pareto front and a broader coverage/diversity of solutions. MOEA/D places with an intermediate performance, standing out for its global consistency across most metrics. This figure allows for clearly identifying the particular strengths of each algorithm: R-VEA in convergence, NSGA-III in diversity, and MOEA/D as a balanced solution.

Figure 18 analyzes the behavior of the algorithms under different network load conditions (low, medium, high, and saturated). Similarly to the section on Figure 17, the number of times an algorithm had the best performance according to the load level is counted. In low load scenarios, differences between algorithms are minor, given that the network has sufficient resources and VNF placement does not present significant conflicts. As the load increases, differences become more noticeable: R-VEA maintains superior stability, preserving a well-defined Pareto front even under saturation conditions. NSGA-III, while retaining good solution dispersion, shows greater variability and lower precision in convergence. MOEA/D, for its part, achieves competitive results at low loads, but its performance decreases slightly in saturation scenarios due to its lower search space exploration capacity.

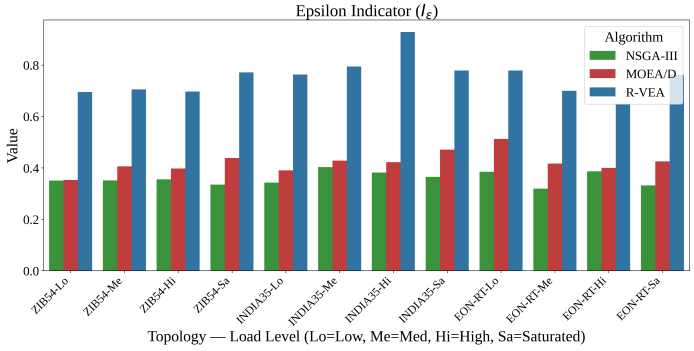
Figure 19 summarizes the global performance count of the number of times the three algorithms (NSGA-III, MOEA/D, and R-VEA) achieved the best result, considering the ZIB54, INDIA, and EON topologies. It is observed that algorithm behavior varies notably depending on the evaluated network structure.



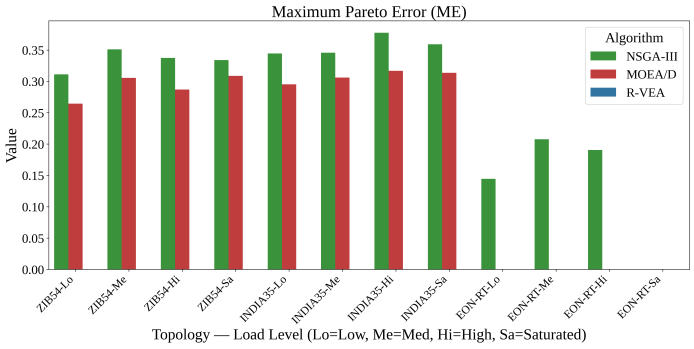
(a) Generational Distance (GD)



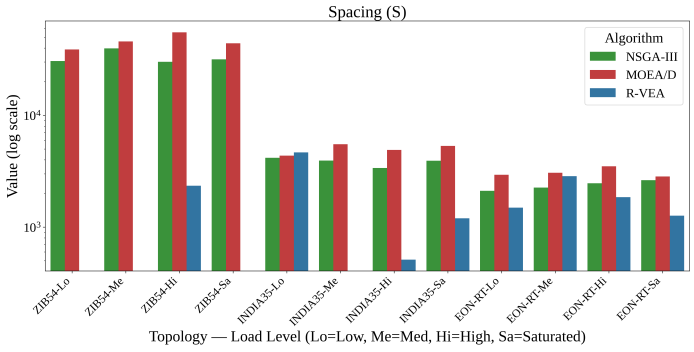
(b) Inverted Generational Distance (IGD)



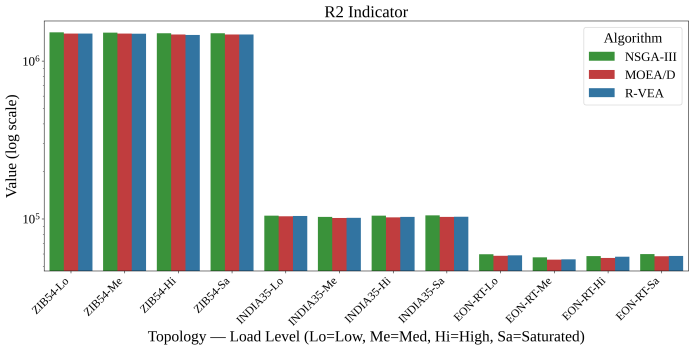
(c) Epsilon Indicator (I_ϵ)



(d) Maximum Pareto Error (ME)



(e) Spacing (S)



(f) R Indicators (R_1, R_2, R_3)

Figure 16: MaOEA Performance

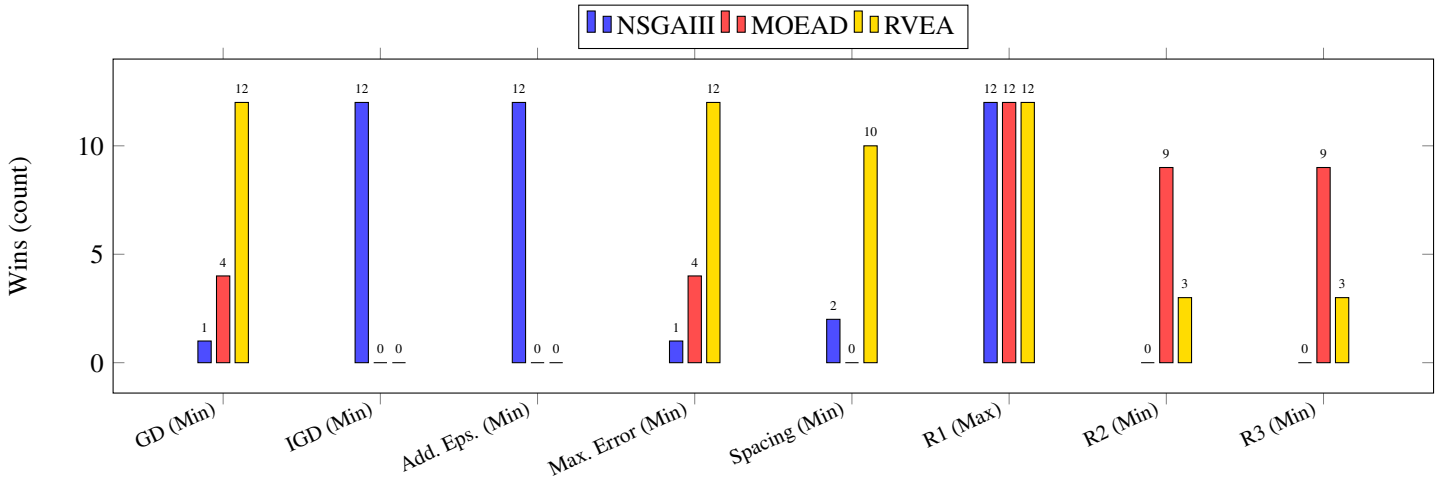


Figure 17: Performance summary by metric

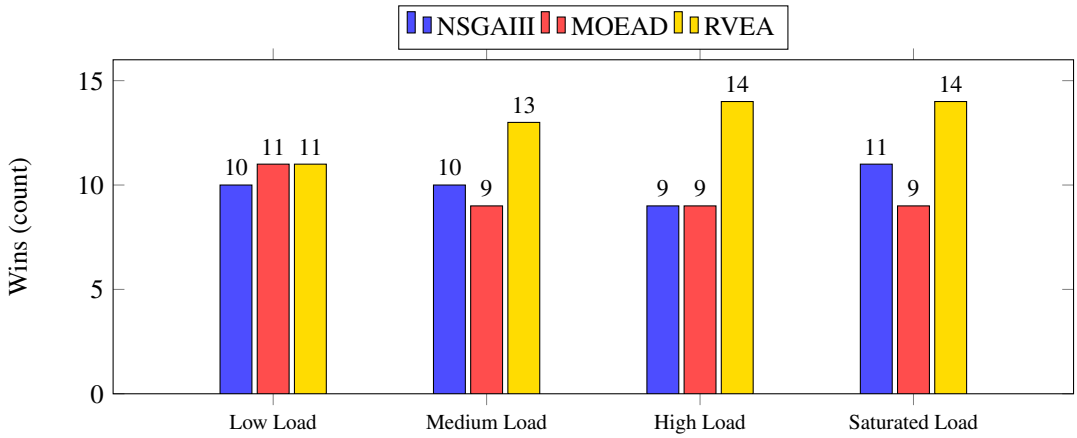


Figure 18: Performance summary by load

In the ZIB54 topology (large network), the R-VEA algorithm demonstrates clearly superior performance, reaching the highest number of favorable metrics (22), whereas NSGA-III and MOEA/D obtain more modest performances (12 and 6, respectively). In the case of the INDIA topology (medium size), all three algorithms present a more balanced behavior: NSGA-III obtains 13 metrics, alongside R-VEA with 15 and MOEA/D with 12, suggesting a tighter competition. Finally, in the EON topology (small network), the best performance corresponds to MOEA/D with 20 metrics, followed by NSGA-III and R-VEA, both with 15.

These results reflect that there is no single dominant algorithm across all topologies, but rather their effectiveness depends on the structural characteristics of each network. In particular, R-VEA excels in larger networks like ZIB54, while MOEA/D benefits in more compact topologies such as EON.

Figure 20 presents a comparative performance summary by algorithm, integrating the average values of all analyzed metrics and scenarios. The graph allows for clearly visualizing the overall superiority of R-VEA, which combines precise convergence and balanced diversity, outperforming the other approaches in most evaluated contexts. R-VEA accumulates 52 cases (≈ 40%), followed by NSGA-III with 40 (≈ 30.8%) and MOEA/D with 38 (≈ 29.2%). These results consolidate the trend shown in previous figures: R-VEA is the most consistent method by obtaining the highest number of victories across metrics and topologies, while NSGA-III and MOEA/D exhibit a competitive and proximate performance to each other.

8.5. Execution Time Analysis

The execution times recorded for each network topology and traffic load scenario are summarized in Table 20. The results reveal two consistent trends. First, execution time is strongly influenced by the topology size: the Zib54 topology (54 nodes) required between 6 and 7 times more computation than EON-RT (28 nodes). Second, within the same topology, variations across traffic load levels are moderate (generally below 20%), indicating that structural network size has a larger impact on execution time than traffic volume.

R-VEA was consistently the most computationally efficient algorithm across all topologies and load levels. On average, NSGA-III required approximately 6–11% more time than R-VEA, making both algorithms comparable and suitable for practical deployment scenarios. MOEA/D, in contrast, required between 3.5 and 3.9 times the execution time of R-VEA across all scenarios. For example, in

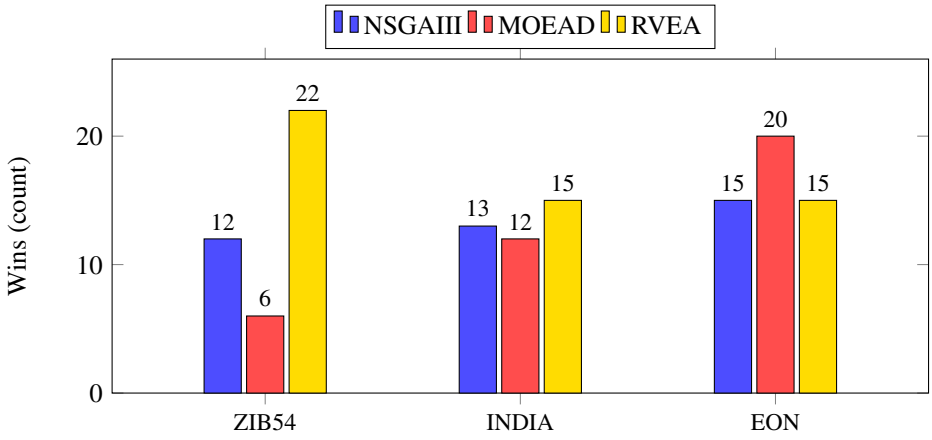


Figure 19: Performance summary by topology

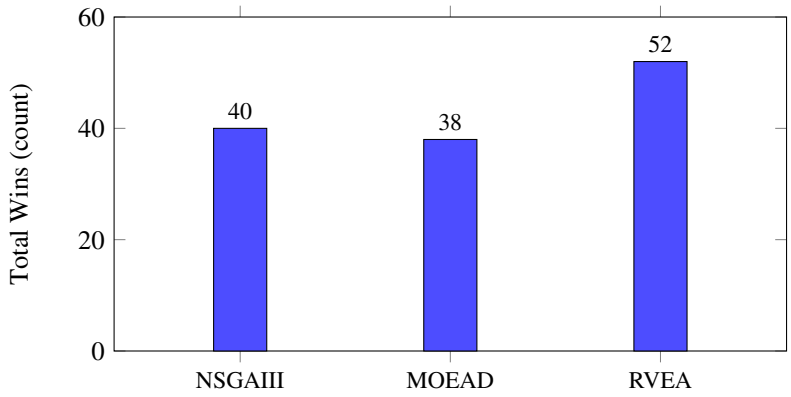


Figure 20: Results summary by algorithm

NET-zib54 with Traffic_2000, NSGA-III ran for 09:49:56, R-VEA for 09:18:53, and MOEA/D for 34:10:27—a ratio of approximately 3.7×.

Table 20: Wall-clock execution time per topology, traffic load, and algorithm (30 independent runs)

Topology	Traffic	NSGA-III	R-VEA	MOEA/D
NET-EON-RT	Traffic_1500	01:34:54	01:26:26	05:01:52
NET-EON-RT	Traffic_300	01:25:39	01:18:37	04:35:48
NET-EON-RT	Traffic_340	01:25:10	01:17:00	04:36:52
NET-EON-RT	Traffic_500	01:21:44	01:14:00	04:30:17
NET-EON-RT	Traffic_980	01:26:28	01:16:02	04:43:55
NET-INDIA35	Traffic_1000	02:39:09	02:22:33	09:03:58
NET-INDIA35	Traffic_2000	02:53:21	02:38:50	09:47:48
NET-INDIA35	Traffic_400	02:38:29	02:24:02	09:08:40
NET-INDIA35	Traffic_600	02:31:04	02:21:17	08:31:29
NET-zib54	Traffic_1000	08:59:49	08:30:55	31:41:03
NET-zib54	Traffic_2000	09:49:56	09:18:53	34:10:27
NET-zib54	Traffic_3000	09:44:24	09:30:33	34:15:20
NET-zib54	Traffic_600	08:22:31	08:04:00	29:17:26

The higher execution cost of MOEA/D can be attributed to its decomposition strategy: in an 11-objective space, the algorithm must repeatedly evaluate Tchebycheff scalarizing functions and update subproblem neighborhoods for a large number of weight vectors, generating substantial overhead compared to the non-dominated sorting of NSGA-III or the angle-penalized distance of R-VEA. Notably, within each topology, the differences across traffic loads are minor (generally <20%), which confirms that network size—rather than request volume—is the primary driver of execution time.

8.6. Trade-offs Interpretation (QoS vs. OPEX/CAPEX)

One of the primary advantages of modeling VNF placement as a MaOP is the ability to expose the underlying conflicts between Quality of Service (QoS) metrics and Capital/Operational Expenditures (CAPEX/OPEX). In traditional single-objective approaches, minimizing costs often implicitly degrades service quality. However, the Pareto fronts obtained by the MaOEAs allow network operators to quantify this degradation.

For example, a solution that prioritizes OPEX reduction by consolidating VNFs into the minimum number of active servers (minimizing Energy Cost f_1 and License Cost f_{10}) inevitably forces traffic to traverse longer paths, thereby increasing Latency (f_3) and potentially decreasing the Network Effective Transfer Rate (f_{16}, QoS). Conversely, distributing VNFs close to the edge to maximize QoS requires provisioning more servers and software licenses, driving up OPEX.

The extracted Pareto fronts consistently demonstrate that algorithms like R-VEA can provide solutions capable of maintaining high Throughput without disproportionately escalating Energy

Costs or Distance, offering a balanced operational middle-ground for service providers across varying topologies such as EON-RT and zib54.

8.7. Statistical Validation

To statistically validate the performance among the MaOEAs, we employ the non-parametric Friedman test [67] alongside the Vargha-Delaney (A_{12}) effect size [68], and pairwise Wilcoxon rank-sum tests [69] over the Inverted Generational Distance (IGD) indicator. Because multiple pairwise comparisons are performed across various algorithms and scenarios, the Holm-Bonferroni procedure [70] is applied to adjust the p-values, thereby controlling the family-wise error rate (FWER) and preventing false positives (Type I errors). Table 21 presents a summary of the statistical tests across 30 independent runs per topology and traffic scenario. The results confirm that R-VEA is the dominant algorithm, outperforming MOEA/D and NSGA-III in the vast majority of cases. Specifically, against NSGA-III, the results show strong statistical significance (adjusted $p < 0.05$) and large effect sizes ($A_{12} \approx 0.99 - 1.0$). When compared to MOEA/D, R-VEA maintains a clear advantage in terms of effect size ($A_{12} \geq 0.77$); however, in certain specific scenarios the adjusted p-value approaches 0.5, indicating a lack of statistical significance exclusively in those particular cases.

Table 21: Statistical validation (Friedman and Wilcoxon test, $p < 0.05$) and Vargha-Delaney A_{12} for IGD.

Topology	Traffic	Best Algo	Compared against	A_{12}
NET-EON-RT	300, 500, 980, 1500	R-VEA	MOEA/D, NSGA-III	1.00 (+)
NET-INDIA35	400, 600, 1000, 2000	R-VEA	MOEA/D, NSGA-III	1.00 (+)
NET-zib54	600	R-VEA	MOEA/D, NSGA-III	0.86, 0.93 (+)
NET-zib54	1000, 2000	R-VEA	MOEA/D, NSGA-III	0.61 - 1.00 (+)
NET-zib54	3000	MOEA/D	NSGA-III, R-VEA	1.00 (+), 0.00 (=)

The analysis of the statistical tests indicates a clear superiority of R-VEA in most configurations. The high Vargha-Delaney effect sizes (A_{12} ranging from 0.77 to 1.0) combined with the extremely low p-values ($p < 10^{-11}$ for the Friedman tests) provide strong empirical evidence that R-VEA consistently converges closer to the true Pareto front compared to NSGA-III and MOEA/D in the majority of evaluated scenarios.

R-VEA's dominant performance in this 11-objective space can be attributed to its reference vector guided approach and its Angle-Penalized Distance (APD) criterion. By dynamically adjusting its reference vectors based on the objective space geometry and balancing convergence and diversity through the APD, R-VEA prevents the population from clustering in sub-optimal local regions—a frequent pathology in highly constrained combinatorial problems such as VNF placement. In contrast, NSGA-III's reliance on Pareto dominance weakens as the number of objectives increases, and MOEA/D suffers from the computational overhead and rigidity of its scalarizing functions across so many dimensions.

However, an interesting exception occurs in the `NET-zib54` topology at the highest traffic load (`Traffic_3000`). Under this extreme congestion, the network's resources are saturated, likely causing the true Pareto front to become highly disconnected, sparse, or severely distorted. In this specific boundary condition, the fixed reference vectors of R-VEA appear to struggle to adapt to the irregular front. Meanwhile, MOEA/D managed to find slightly better localized solutions through its weight-vector decomposition ($A_{12} = 1.0$ against NSGA-III). This suggests that while R-VEA is generally the most robust and efficient algorithm for typical and moderately-high loads, decomposition-based strategies like MOEA/D might offer slight advantages in ultra-congested, highly discontinuous boundary scenarios.

8.8. Robustness Analysis

Due to the stochastic nature of evolutionary algorithms, analyzing their robustness across independent runs is critical. Figure 21 presents the distribution of the final Inverted Generational Distance (IGD) values achieved by each MaOEA across 30 independent executions under the highest traffic load for each topology. The boxplots reveal that NSGA-III and MOEA/D consistently attain lower IGD scores with very tight distributions (low variance), indicating highly stable and robust performance across different random initializations. Conversely, R-VEA exhibits a wider interquartile range and higher median IGD values, demonstrating greater sensitivity to the initial population and stochastic variations during the search process. This indicates that while R-VEA can find competitive individual solutions (as shown by other metrics), NSGA-III and MOEA/D provide more reliable and consistent overall approximations of the Pareto front.

8.9. MANO Integration and Practical Edge Deployment

In a practical NFV environment, the deployment of these solutions would be integrated into the NFV Management and Orchestration (MANO) framework. The MaOEA acts as an intelligent placement engine within the NFV Orchestrator (NFVO). Once the Pareto front is generated, a Multi-Criteria Decision Making (MCDM) method is required to select a single actionable solution based on the service provider's dynamic priorities.

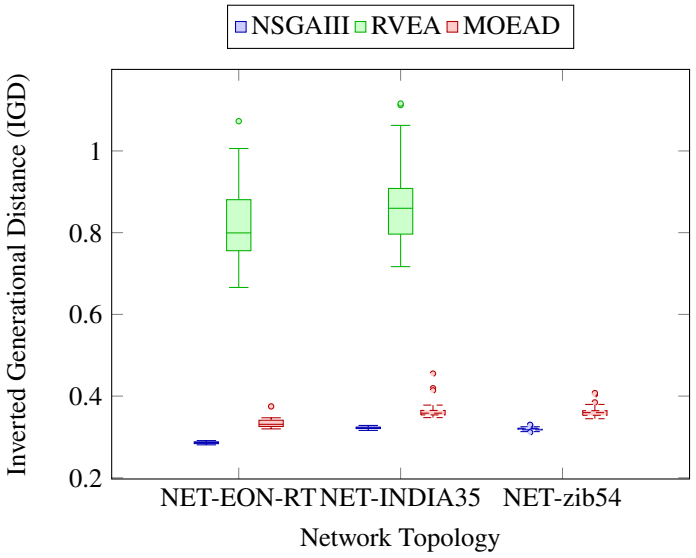


Figure 21: Robustness of MaOEAs: Final IGD distribution across 30 independent runs

For instance, applying the TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) method to our resulting

Pareto fronts with equal weights for all 11 retained objectives would yield a single solution characterized by a highly balanced trade-off. As an illustrative example, a representative placement configuration selected by TOPSIS could exhibit values of the order of: an Energy Cost around 9–10 units, a Delay Cost of approximately 17000 units, around 90–100 instantiated VNF instances, and a Throughput in the range of -30 to -25 units, achieving a TOPSIS composite score close to 0.86. This automated MCDM selection mechanism allows the MANO system to seamlessly bridge the gap between many-objective mathematical optimization and the physical instantiation of VNFs at the network edge.

8.10. Limitations and Baseline Considerations

It is important to acknowledge that this study focuses exclusively on comparing Many-Objective Evolutionary Algorithms (MaOEAs). A comparison against an exact baseline such as a Mixed Integer Linear Programming (MILP) solver or classical heuristic algorithms (e.g., Greedy First-Fit) was not included in our experimental setup. The VNF Placement problem is inherently NP-Hard, and as widely demonstrated in the literature, exact MILP models scale exponentially with both network size and constraint complexity. Consequently, solving a MILP formulation that simultaneously incorporates 11 conflicting objectives is computationally intractable for the large topologies evaluated in our study (such as `NET-INDIA35` and `NET-zib54`). Furthermore, adapting traditional Greedy heuristics to navigate an 11-dimensional objective space without introducing significant scalarization bias is highly complex and remains outside the scope of this work. Therefore, our reliance on MaOEAs is fundamentally motivated by the critical need to discover acceptable, near-optimal multi-objective trade-offs within practical computational timeframes, specifically in large-scale scenarios where exact methods are structurally incapable of scaling.

9. Conclusions and Future Work

Based on the key contributions of this work, which include the correlation analysis within the state of the art and the performance comparison of evolutionary algorithms (NSGA-III, R-VEA, and MOEA/D) for optimizing VNF placement to guarantee QoS and reduce CAPEX and OPEX, the following conclusions can be drawn:

- **Correlation Analysis:** The first major contribution of this work consisted of performing a correlation analysis of the state of the art, which is a crucial step to simplify and reduce the complexity of the optimization problem. By identifying the relationships among the objectives, the number of objectives to be optimized can be reduced, thereby facilitating the search for more effective solutions. This can simplify the search process, render the problem more manageable, and lead to faster convergence of the optimization algorithms as well as the obtainment of more practical and efficient solutions.
- **Algorithm Comparison:** The second contribution focuses on comparing the performance of evolutionary algorithms under rigorous statistical scrutiny. Through non-parametric Friedman testing and pairwise Wilcoxon tests (adjusted $p < 0.05$ via Holm-Bonferroni) and Vargha-Delaney effect sizes ($A_{12} \geq 0.86$), the results indicate that R-VEA consistently demonstrates superior performance across most topologies and load scenarios. This robustness is attributed to its Angle-Penalized Distance (APD) metric, which adeptly balances convergence and diversity in highly dimensional Pareto fronts. This performance consistency suggests that R-VEA is a reliable and effective option for addressing the VNF placement problem with multiple objectives, except in highly saturated network edge cases where MOEA/D occasionally exhibited resilience due to Pareto front distortion.
- **Practical Implications for Network Operators:** Finally, the formulated many-objective optimization framework offers tangible benefits for telecommunication operators migrating to NFV infrastructures. By demonstrating that algorithms like R-VEA can successfully balance conflicting operational dimensions—such as minimizing energy consumption (OPEX) and server installations (CAPEX) while adhering to SLA delay bounds (QoS)—network administrators are provided with a robust decision-support tool. This capability enables operators to select deployment configurations tailored to specific business policies, effectively reducing infrastructure footprint without sacrificing user experience.

In summary, this work has made significant contributions by addressing the VNF placement optimization problem in networks. The simplification through correlation analysis and the effective selection of the R-VEA algorithm have led to promising results in terms of generating high-performance solutions on the Pareto front. These conclusions support the utility and applicability of this approach to improve network resource management concerning QoS, CAPEX, and OPEX. In addition to the current contributions, several research areas for future work are proposed, which could expand and enrich the field:

- **Dynamic adaptability of the algorithms:** Extending the VNF placement problem from a static scenario to a dynamic one, where the topology, traffic load, and service chains vary over time. In this context, it is relevant to study adaptive variants of R-VEA, MOEA/D, and NSGA-III that incorporate memory mechanisms, change detection, and population reheating in order to maintain the quality of the Pareto front in the face of events such as link failures, traffic spikes, or the deployment of new services.
- **Impact of objective reduction on decision-making:** Conducting a systematic study comparing the solutions obtained when utilizing all objective functions versus the case where only the subset reduced via Pearson correlations is employed. The analysis should evaluate how VNF placement, link utilization, node consolidation, and QoS levels change, as well as the shape of the approximated Pareto front. A key aspect is to quantify how much is lost (or gained) in terms of solution quality, robustness, and variety of alternatives for the operator when working with a reduced but computationally and interpretatively lighter model.

- **Comparison and improvement of objective reduction schemes:** Delving into the comparison between objective function selection based on Pearson correlations and other dimensionality reduction approaches (e.g., PCA, mutual information analysis, or objective clustering). An interesting future work is to systematically analyze how each reduction scheme impacts the convergence, Pareto front diversity, and computational cost of many-objective evolutionary algorithms applied to the VNF placement problem.
- **Evaluation in large-scale scenarios and with real traces:** Replicating and extending the conducted experiments to larger and more heterogeneous topologies (e.g., operator WAN networks, 5G/edge scenarios, or multi-domain networks), as well as utilizing real traffic traces and demand patterns. This would allow validating the robustness of the algorithms and the reduced set of objective functions in operational contexts closer to production deployments.
- **Hybridization with machine learning techniques:** Investigating the integration of machine learning models to assist the evolutionary process, whether through traffic pattern prediction, construction of surrogate models to estimate expensive objective functions, or dynamic adjustment of evolutionary parameters. This hybridization could reduce execution times and improve the system's response in online scenarios.
- **Thorough baseline comparison:** Conducting a comprehensive comparison against exact methods (e.g., Mixed Integer Linear Programming formulations) or specialized state-of-the-art single-objective heuristics. While a full MILP evaluation is computationally intractable at the scale of our current experiments due to the combination of 11 objectives and large network instances, establishing these baselines on smaller-scale scenarios or using simplified models constitutes an important direction to quantify the optimality gap and further validate the relative effectiveness of the many-objective evolutionary approach.

These lines of future work seek to delve deeper into the treatment of the dynamic nature of the VNF placement problem, improve objective reduction strategies, and bring evolutionary proposals closer to realistic operation scenarios, thereby strengthening the practical applicability of many-objective optimization in NFV-based networks.

Conflict of Interest The authors declare no conflict of interest.

Acknowledgment We would like to thank the Facultad Politécnica, Universidad Nacional de Asunción for supporting this research.

References

[1] ETSI Industry Specification Group for Network Functions Virtualisation, "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action," Technical report, ETSI, 2012.

[2] J. Billingsley, K. Li, W. Miao, G. Min, N. Georgalas, "A Formal Model for Multi-objective Optimisation of Network Function Virtualisation Placement," in International Conference on Evolutionary Multi-Criterion Optimization, 529–540, Springer, 2019, doi:10.1007/978-3-030-12598-1_42.

[3] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, F. Huici, "ClickOS and the Art of Network Function Virtualization," in Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), 459–473, 2014.

[4] B. Yi, X. Wang, K. Li, M. Huang, et al., "A comprehensive survey of network function virtualization," Computer Networks, **133**, 212–262, 2018, doi:10.1016/j.comnet.2018.01.021.

[5] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, "On orchestrating virtual network functions," in 2015 11th International Conference on Network and Service Management (CNSM), 50–56, IEEE, 2015, doi:10.1109/cnsm.2015.7367338.

[6] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, O. C. M. B. Duarte, "Orchestrating virtualized network functions," IEEE Transactions on Network and Service Management, **13**(4), 725–739, 2016, doi:10.1109/tnsm.2016.2569020.

[7] C. A. C. Coello, "Computación Evolutiva," Incomplete source details in the original BibTeX file.

[8] B. Addis, D. Belabed, M. Bouet, S. Secci, "Virtual network functions placement and routing optimization," in 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), 171–177, IEEE, 2015, doi:10.1109/cloudnet.2015.7335301.

[9] M. Gao, B. Addis, M. Bouet, S. Secci, "Optimal orchestration of virtual network functions," Computer Networks, **142**, 108–127, 2018, doi:10.1016/j.comnet.2018.06.006.

[10] S. Lange, A. Grigorjew, T. Zinner, P. Tran-Gia, M. Jarschel, "A multi-objective heuristic for the optimization of virtual network function chain placement," in 2017 29th International Teletraffic Congress (ITC 29), volume 1, 152–160, IEEE, 2017, doi:10.23919/itc.2017.8064351.

[11] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE transactions on evolutionary computation, **6**(2), 182–197, 2002, doi:10.1109/4235.996017.

[12] C. Von Lücken, B. Barán, C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," Computational optimization and applications, **58**(3), 707–756, 2014, doi:10.1007/s10589-014-9644-1.

[13] S. Chand, M. Wagner, "Evolutionary Many-Objective Optimization: A Quick-Start Guide," Surveys in Operations Research and Management Science, **20**(2), 35–42, 2015, doi:10.1016/j.sorms.2015.08.001.

[14] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," IEEE Communications Surveys & Tutorials, **17**(1), 14–76, 2015, doi:10.1109/SURV.2014.032014.00182.

[15] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization," in ACM SIGOPS Operating Systems Review, volume 37, 164–177, 2003, doi:10.1145/1165389.945462.

[16] A. Tomassilli, F. Giroire, N. Huin, S. Pérennes, "Provably efficient algorithms for placement of service function chains with ordering constraints," in IEEE INFOCOM 2018-IEEE Conference on Computer Communications, 774–782, IEEE, 2018, doi:10.1109/infocom.2018.8486275.

[17] O. Soualah, M. Mechtri, C. Ghribi, D. Zeghlache, "Online and batch algorithms for VNFs placement and chaining," Computer Networks, **158**, 98–113, 2019, doi:10.1016/j.comnet.2019.01.041.

[18] O. Soualah, M. Mechtri, C. Ghribi, D. Zeghlache, "Energy efficient algorithm for VNF placement and chaining," in 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 579–588, IEEE, 2017, doi:10.1109/ccgrid.2017.84.

[19] M. A. Raayatpanah, T. Weise, "Virtual network function placement for service function chaining with minimum energy consumption," in 2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET), 198–202, IEEE, 2018, doi:10.1109/ccet.2018.8542223.

[20] D. Amaya, Y. Sumi, S. Homma, T. Okugawa, T. Tachibana, "VNF placement with optimization problem based on data throughput for service chaining," in 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), 1–3, IEEE, 2018, doi:10.1109/cloudnet.2018.8549543.

[21] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, R. Boutaba, "Delay-aware VNF placement and chaining based on a flexible resource allocation approach," in 2017 13th International Conference on Network and Service Management (CNSM), 1–7, ieee, 2017, doi:10.23919/cnsm.2017.8255993.

[22] M. Savi, M. Tornatore, G. Verticale, "Impact of processing-resource sharing on the placement of chained virtual network functions," IEEE Transactions on Cloud Computing, 2019, doi:10.1109/tcc.2019.2914387.

[23] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, B. Mukherjee, "Joint virtual network function placement and routing of traffic in operator networks," UC Davis, Davis, CA, USA, Tech. Rep, 2015.

[24] O. Soualah, M. Mechtri, C. Ghribi, D. Zeghlache, "A green VNFs placement and chaining algorithm," in NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, 1–5, IEEE, 2018, doi:10.1109/noms.2018.8406183.

[25] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, 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), 98–106, IEEE, 2015, doi:10.1109/inm.2015.7140281.

[26] Y. Sang, B. Ji, G. R. Gupta, X. Du, L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in IEEE INFOCOM 2017-IEEE Conference on Computer Communications, 1–9, IEEE, 2017, doi:10.1109/infocom.2017.8057036.

[27] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, L. P. Gaspary, "A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining," Computer Communications, **102**, 67–77, 2017, doi:10.1016/j.comcom.2016.11.002.

[28] H. Xing, X. Zhou, X. Wang, S. Luo, P. Dai, K. Li, H. Yang, "An integer encoding grey wolf optimizer for virtual network function placement," Applied Soft Computing, **76**, 575–594, 2019, doi:10.1016/j.asoc.2018.12.037.

[29] R. Cziva, C. Anagnostopoulos, D. P. Pezaros, "Dynamic, latency-optimal vNF placement at the network edge," in Ieee infocom 2018-ieee conference on computer communications, 693–701, IEEE, 2018, doi:10.1109/infocom.2018.8486021.

[30] D. B. Oljira, K.-J. Grinnemo, J. Taheri, A. Brunstrom, "A model for QoS-aware VNF placement and provisioning," in 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 1–7, IEEE, 2017, doi:10.1109/nfv-sdn.2017.8169829.

[31] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," Networks, **71**(2), 97–106, 2018, doi:10.1002/net.21768.

[32] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, R. Boutaba, "Elastic virtual network function placement," in 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), 255–260, IEEE, 2015, doi:10.1109/cloudnet.2015.7335318.

[33] C. Pham, N. H. Tran, S. Ren, W. Saad, C. S. Hong, "Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach," IEEE Transactions on Services Computing, 2017, doi:10.1109/tsc.2017.2671867.

[34] O. A. Wahab, N. Kara, C. Edstrom, Y. Lemieux, "MAPLE: A Machine Learning Approach for Efficient Placement and Adjustment of Virtual Network Functions," Journal of Network and Computer Applications, **142**, 37–50, 2019, doi:10.1016/j.jnca.2019.06.003.

[35] D. Qi, S. Shen, G. Wang, "Towards an efficient VNF placement in network function virtualization," Computer Communications, **138**, 81–89, 2019, doi:10.1016/j.comcom.2019.03.005.

[36] H. Moens, F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in 10th International Conference on Network and Service Management (CNSM) and Workshop, 418–423, IEEE, 2014, doi:10.1109/cnsm.2014.7014205.

[37] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi, W. Kellerer, "QoS-driven function placement reducing expenditures in NFV deployments," in 2017 IEEE International Conference on Communications (ICC), 1–7, IEEE, 2017, doi:10.1109/icc.2017.7996513.

[38] S. Mehraghdam, M. Keller, H. Karl, "Specifying and placing chains of virtual network functions," in 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), 7–13, IEEE, 2014, doi:10.1109/cloudnet.2014.6968961.

[39] S. Ahvar, H. P. Phyu, S. M. Buddhacharya, E. Ahvar, N. Crespi, R. Glitho, "CCVP: Cost-efficient centrality-based VNF placement and chaining algorithm for network service provisioning," in 2017 IEEE Conference on Network Softwarization (NetSoft), 1–9, IEEE, 2017, doi:10.1109/netsoft.2017.8004104.

[40] X. Song, X. Zhang, S. Yu, S. Jiao, Z. Xu, "Resource-efficient virtual network function placement in operator networks," in GLOBECOM 2017-2017 IEEE Global Communications Conference, 1–7, IEEE, 2017, doi:10.1109/glocom.2017.8254492.

[41] T.-H. Nguyen, J. Lee, M. Yoo, "A Practical Model for Optimal Placement of Virtual Network Functions," in 2019 International Conference on Information Networking (ICOIN), 239–241, IEEE, 2019, doi:10.1109/icoin.2019.8717979.

[42] R. Cohen, L. Lewin-Eytan, J. S. Naor, D. Raz, "Near optimal placement of virtual network functions," in 2015 IEEE Conference on Computer Communications (INFOCOM), 1346–1354, IEEE, 2015, doi:10.1109/infocom.2015.7218511.

[43] S. Tavakoli-Someh, M. H. Rezvani, "Multi-objective virtual network function placement using NSGA-II meta-heuristic approach," The Journal of Supercomputing, 1–37, 2019, doi:10.1007/s11227-019-02849-y.

[44] S. Khebbache, M. Hadji, D. Zeghlache, "A Multi-Objective Non-Dominated Sorting Genetic Algorithm for VNF Chains Placement," in 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), 1–4, IEEE, 2018, doi:10.1109/CCNC.2018.8319250.

[45] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, et al., Evolutionary algorithms for solving multi-objective problems, volume 5, Springer, 2007, doi:10.1007/978-1-4757-5184-0.

[46] K. Deb, "Multi-objective optimisation using evolutionary algorithms: an introduction," in Multi-objective evolutionary optimisation for product design and manufacturing, 3–34, Springer, 2011, doi:10.1007/978-0-85729-652-8_1.

[47] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, 2001.

[48] M. Farina, P. Amato, "On the Optimal Solution Definition for Many-Criteria Optimization Problems," in 2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings, NAFIPS-FLINT 2002, 233–238, IEEE, 2002, doi:10.1109/NAFIPS.2002.1018061.

[49] S. Khebbache, M. Hadji, D. Zeghlache, "Scalable and cost-efficient algorithms for VNF chaining and placement problem," in 2017 20th conference on innovations in clouds, internet and networks (ICIN), 92–99, IEEE, 2017, doi:10.1109/icin.2017.7899395.

[50] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, V. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," IEEE Transactions on Evolutionary Computation, **7**(2), 117–132, 2003, doi:10.1109/TEVC.2003.810758.

[51] K. C. Tan, T. H. Lee, E. F. Khor, "Evolutionary Algorithms for Multi-Objective Optimization: Performance Assessments and Comparisons," Artificial Intelligence Review, **17**(4), 251–290, 2002, doi:10.1023/A:1015516501242.

[52] D. Hadka, "MOEA Framework: A Free and Open Source Java Framework for Multiobjective Optimization," 2025, version 5.1 [Computer software].

[53] K. Bringmann, T. Friedrich, "Computing the hypervolume indicator: exact algorithms, approximations, and lower bounds," Artificial Intelligence, **205**, 100–122, 2013.

[54] L. While, L. Bradstreet, L. Barone, "A fast way of calculating exact hypervolumes," IEEE Transactions on Evolutionary Computation, **16**(1), 86–95, 2012, doi:10.1109/tevc.2010.2077298.

[55] N. Beume, C. M. Fonseca, M. Lopez-Ibanez, L. Paquete, J. Vahrenhold, "The complexity of computing the hypervolume indicator," IEEE Transactions on Evolutionary Computation, **13**(5), 1075–1082, 2009, doi:10.1109/tevc.2009.2015575.

[56] J. Miao, L. Niu, "A Survey on Feature Selection," Procedia Computer Science, **91**, 919–926, 2016, doi:10.1016/j.procs.2016.07.111.

[57] N. Honest, K. Kotecha, "A survey on feature selection techniques," International Journal of Computer Applications, **975**(8887), 1–6, 2020.

[58] B. G. Bathula, J. M. Elmirghani, "Constraint-based anycasting over optical burst switched networks," Journal of Optical Communications and Networking, **1**(2), A35–A43, 2009, doi:10.1364/jocn.1.000a35.

[59] M. Yang, K. Guo, Y. Zhang, Y. Ji, "Routing, modulation level, spectrum and transceiver assignment in elastic optical networks," IEICE Transactions on Communications, **101**(5), 1197–1209, 2018, doi:10.1587/transcom.2017ebp3309.

[60] M. K. Awad, Y. Rafique, S. Alhadlaq, D. Hassoun, A. Alabdulhadi, S. Thani, "A greedy power-aware routing algorithm for software-defined networks," in 2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 268–273, IEEE, 2016, doi:10.1109/isspit.2016.7886047.

[61] R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, "A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization," IEEE Transactions on Evolutionary Computation, **20**(5), 773–791, 2016, doi:10.1109/TEVC.2016.2519378.

[62] Q. Zhang, H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," IEEE Transactions on Evolutionary Computation, **11**(6), 712–731, 2007, doi:10.1109/TEVC.2007.892759.

[63] K. Deb, H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems with Box Constraints," IEEE Transactions on Evolutionary Computation, **18**(4), 577–601, 2014, doi:10.1109/TEVC.2013.2281535.

[64] K. Deb, R. B. Agrawal, "Simulated binary crossover for continuous search space," Complex Systems, **9**(2), 115–148, 1995.

[65] R. Storn, K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, **11**(4), 341–359, 1997, doi:10.1023/a:1008202821328.

[66] Zuse Institute Berlin, "ZIB54 Network Topology," Benchmark network instance, network with 54 nodes.

[67] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," Journal of the american statistical association, **32**(200), 675–701, 1937, doi:10.1080/01621459.1937.10503522.

[68] A. Vargha, H. D. Delaney, "A critique and improvement of the CL common language effect size statistics of McGraw and Wong," Journal of Educational and Behavioral Statistics, **25**(2), 101–132, 2000, doi:10.3102/10769986025002101.

[69] F. Wilcoxon, "Individual comparisons by ranking methods," Biometrics bulletin, **1**(6), 80–83, 1945, doi:10.2307/3001968.

[70] S. Holm, "A simple sequentially rejective multiple test procedure," Scandinavian journal of statistics, 65–70, 1979, doi:10.2307/4615733.