future internet

Article

A Novel Strategy for VNF Placement in Edge Computing Environments

Anselmo Luiz Éden Battisti [1,*], Evandro Luiz Cardoso Macedo [2,*], Marina Ivanov Pereira Josué [1], Hugo Barbalho [3], Flávia C. Delicato [1], Débora Christina Muchaluat-Saade [1], Paulo F. Pires [1], Douglas Paulo de Mattos [1] and Ana Cristina Bernardo de Oliveira [3]

[1] MidiaCom Laboratory, Institute of Computing, Universidade Federal Fluminense (UFF), Niterói 24210-240, Brazil
[2] High-Speed Networks Laboratory, Systems Engineering and Computer Science Program (PESC), Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro 21941-914, Brazil
[3] Dell EMC Research Center, Rua Antônio Barros de Castro, 119, Cidade Universitária, Rio de Janeiro 21941-615, Brazil
* Correspondence: anselmo@midiacom.uff.br (A.L.É.B.); evandro@ravel.ufrj.br (E.L.C.M.)

Abstract: Network function virtualization (NFV) is a novel technology that virtualizes computing, network, and storage resources to decouple the network functions from the underlying hardware, thus allowing the software implementation of such functions to run on commodity hardware. By doing this, NFV provides the necessary flexibility to enable agile, cost-effective, and on-demand service delivery models combined with automated management. Different management and orchestration challenges arise in such virtualized and distributed environments. A major challenge in the selection of the most suitable edge nodes is that of deploying virtual network functions (VNFs) to meet requests from multiple users. This article addresses the VNF placement problem by providing a novel integer linear programming (ILP) optimization model and a novel VNF placement algorithm. In our definition, the multi-objective optimization problem aims to (i) minimize the energy consumption in the edge nodes; (ii) minimize the total latency; and (iii) reducing the total cost of the infrastructure. Our new solution formulates the VNF placement problem by taking these three objectives into account simultaneously. In addition, the novel VNF placement algorithm leverages VNF sharing, which reuses VNF instances already placed to potentially reduce computational resource usage. Such a feature is still little explored in the community. Through simulation, numerical results show that our approach can perform better than other approaches found in the literature regarding resource consumption and the number of SFC requests met.

Keywords: 5G; edge computing; VNF placement

Citation: Battisti, A.L.É.; Macedo, E.L.C.; Josué, M.I.P.; Barbalho, H.; Delicato, F.C.; Muchaluat-Saade, D.C.; Pires, P.F.; Mattos, D.P.d.; Oliveira, A.C.B.d. A Novel Strategy for VNF Placement in Edge Computing Environments. *Future Internet* **2022**, *14*, 361. https://doi.org/10.3390/fi14120361

Academic Editors: Xu Wang, Bin Shi and Yili Fang

Received: 26 October 2022
Accepted: 28 November 2022
Published: 30 November 2022

1. Introduction

The demand for flexibility and quality of service (QoS) has been continuously growing for network services in edge computing environments. One emerging technology that plays a crucial role in delivering such requirements is network function virtualization (NFV) [1]. NFV aims to leverage state-of-the-art technologies that virtualize computing, network, and storage resources to decouple the network functions from the underlying hardware, thus allowing the software implementation of such network functions to run on commodity hardware. By doing this, NFV provides the necessary flexibility to enable agile, cost-effective, and on-demand service delivery models combined with automated management. This software implementation, namely virtual network function (VNF), runs on an NFV infrastructure (NFVI) that encompasses the necessary software and hardware components to provide the expected functionality.

In many cases, the services provided through the virtualization of network functions require that end-to-end data flows go through a sequence of such functions (typically

in defined order). Hence, the concept of service function chaining (SFC) [2] emerges to denote a chain of services, i.e., a sequence of VNFs designed to replace a chain of physical equipment that provides a complete service requested by a user.

Other technologies are commonly used to reap the benefits introduced by NFV. One of them is cloud computing, which allows the deployment of VNFs on data centers far from end-users. NFV and cloud computing can work with other technologies to provide a service-centric architecture. For instance, the edge computing paradigm extends the cloud computing paradigm by exploiting available resources, data, and computing at the network edge. Such an emergent paradigm preserves the key benefits of using clouds as support infrastructure while moving part of the processing and storage capabilities to the edge of the network, therefore repositioning services closer to end-users and data sources.

The significant benefits of edge computing include reduced communication delay, the use of less network bandwidth, control decisions at the network edge, higher security, and data privacy, as presented in [3]. Moreover, edge computing can utilize edge node collaboration to compensate for their lower resource capacity. When integrating NFV and edge computing, VNF instances are deployed in hosts called edge nodes. Such a strategy facilitates the QoS fulfillment of time-critical applications, given the proximity to end-users, while taking advantage of the often idle capabilities of the various computing nodes deployed at the network edge. According to [4], the notion of edge/fog nodes is wide, with any equipment with processing power and storage capacity acting as an edge node.

Different management and orchestration challenges arise in such virtualized and distributed environment. One of these challenges concerns the placement of VNFs in edge nodes, which consists of selecting the most suitable edge nodes to deploy VNFs to meet service requests from multiple applications and users [5]. Typically, the component responsible for the placement is the network function virtualization orchestrator (NFVO) [6]. This must be performed in such a way that the QoS requirements of the applications are met without service-level agreement (SLA) violations, at the same time that the use of service providers' resources at the edge tier is optimized. Note that no single optimal solution exists given that this is a multi-objective optimization problem, i.e., an optimization problem with multiple and conflicting objectives, as defined in [7,8].

Several authors proposed different optimization models and heuristic solutions for managing the VNF placement problem [9–13]. In this paper, we address such a problem by providing a novel integer linear programming (ILP) optimization model and a novel heuristic method to provide more scalable solutions in a viable time. Our approach considers users with heterogeneous demands regarding network latency and bandwidth accessing a system. The edge nodes also have heterogeneous CPU, memory, latency, and bandwidth capacities. Additionally, each edge node has a different subset of VNF types (e.g., virtual machine images or container images) that can be offered on demand. Furthermore, each VNF request has a set of requirements that must be addressed. Some requirements are intrinsically related to the VNF, for example, the minimum amount of CPU available in the edge node. In contrast, others are application-dependent, for example, the maximum tolerable delay. Our formulation also defines that the placement occurs after the arrival of multiple users' requests. All resources of the edge node are available for running new VNF instances before the placement decision. Therefore, VNF instance sharing only occurs among those already-placed VNF instances.

Our solution presents the following contributions:

1. We formulate the VNF placement as a multi-objective optimization problem with the following objectives (i) minimization of the energy consumption in the edge nodes; (ii) minimization of the total latency; and (iii) minimization of the total cost of the infrastructure.
2. We also explore the VNF sharing in a new strategy, combining the VNF sharing with the three previously mentioned objectives.

The remainder of this paper is organized as follows. In Section 2, we review the related work according to the research field's state of the art. Section 3 introduces the problem statement and our proposed model for the VNF placement problem. Section 4 presents the

current challenges in edge environment contexts, and Section 5 presents our novel Smart VNF placement algorithm. We discuss our implementation architecture in Section 6. The experimental results and discussions are presented in Section 7. Finally, Section 8 concludes the paper and points out future work.

2. Related Work

The VNF placement problem is one of the most challenging issues in NFV and edge computing development. The literature presents proposals addressing such problems based on mono-objective or multi-objective approaches. As follows, we discuss such proposals. In Table 1, we depict a comparison among the VNF placement approaches. The last row of the table presents the features of our smart placement algorithm.

Table 1. Comparison of the related work with our approach.

Study	Objective	Approach	Model	Environment	VNF Sharing
Ruiz et al. [14]	Mono	Genetic algorithm	No	Cloud	No
Kim et al. [15]	Mono	Conventional Light chain algorithm	No	N/D	No
Emu et al. [16]	Mono	Machine learning	Yes	Edge	No
Garrich et al. [17]	Mono	Greed	Yes	Edge	No
Li et al. [18]	Mono	Greed	Yes	Edge	No
Nguyen et al. [19]	Multi	Markov chain	Yes	Edge and cloud	No
Reyhanian et al. [20]	Multi	Alternating direction Method of multipliers	Yes	Edge	No
Kiran et al. [21]	Multi	Genetic algorithm	Yes	Edge	No
Pei et al. [22]	Multi	Machine learning	Yes	N/D	No
Bunyakitanon et al. [23]	Multi	Q-learning scheme	Yes	Edge	No
Pham et al. [24]	Multi	Markov chain	Yes	Cloud	No
Li et al. [25]	Multi	Deep reinforcement Learning	Yes	Cloud	No
Niu et al. [26]	Multi	Graph-based particle Swarm optimization	Yes	Edge	No
Alahmad et al. [27]	Multi	Greed	Yes	Cloud	No
Gao et al. [28]	Multi	Steiner Tree and Markov Decision	Yes	Cloud	Yes
Mohamad et al. [29]	Multi	Greed	Yes	Edge	Yes
Our approach	**Multi**	**Smart Placement Algorithm**	**Yes**	**Edge**	**Yes**

Mono-objective approaches [14–18,30] solve the VNF placement problem by using only one characteristic of the environment, such as the resource consumption in edge nodes and, in this case, they try to minimize it. For instance, authors in [30] studied the VNF sharing problem aiming to minimize the cost for the mobile operator, subject to requirements on end-to-end service performance, e.g., total delay. Emu et al. [16] proposed a solution to the VNF placement problem using an ILP model that guarantees the minimum end-to-end latency while ensuring QoS by not overstepping beyond an acceptable limit of latency violation. The authors used the ILP solutions to train an artificial neural network (ANN) to cope with the NP-hardness of such a problem. The authors in [18] also proposed an ILP model aiming to minimizing the total resource consumption.

Multi-objective approaches [19–27,31] solve the VNF placement problem considering multiple, and sometimes conflicting, environment aspects. For instance, the authors in [31] demonstrated the gain of VNF sharing-based service function chaining (SFC) requests placement, as a way of satisfying more requests with average-less resources per request. They formulated the sharing-based SFC placement as an integer linear program (ILP) to minimize the overall deployment cost, but did not take the energy consumption into consideration. The authors in [20] formulated the joint problem of routing, and VNF placement cost minimization subject to flow demands where processing flows in local subnetworks is encouraged. Kiran et al. [21] proposed an SDN-NFV infrastructure to

tackle the VNF placement problem in wireless MEC networks by minimizing the overall placement and resourcing allocation costs. Unlike the already proposed approaches, our strategy tackles the VNF placement problem considering a set of aspects in the multi-objective approach that simultaneously minimize the energy consumption in the edge nodes, minimize the total latency, and reduce the total cost of the infrastructure. In [19], the authors considered the bandwidth capacity and link delay of the network connection between clouds in which VNFs are allocated to minimize resource usage costs.

Pham et al. [24] presented an approach to the VNF placement problem with energy and traffic-aware cost minimization, focusing on minimizing network latency and operational cost. The energy consumption of edge nodes is essential to guarantee QoS due to their power constraints. Energy is spent on tasks such as data generation, transmission, aggregation, storage, and processing. Our proposed energy model, discussed in detail in Section 3, considers the energy consumed for data propagation along with communication links and VNF packet processing in edge nodes. As such, our approach selects edge nodes with low energy consumption according to the SFC priority (latency, capacity, and energy).

Methods derived from Greedy approaches are also widely adopted to solve the VNF placement problem [17,25,27,28]. Such approaches quickly converge to a solution but tend to fall into local minimum or maximum. Other approaches consider machine learning-based strategies [16,19,23,24], which typically require datasets in training and testing phases, sometimes difficult to obtain, hindering their practical deployment in dynamic environments.

Some papers have recently addressed the VNF instance sharing mechanism, as in [32], where authors restrict the number of SFC that can share the same VNF instances based on the resources each SFC will use in the VNF instance. In their approach, each SFC request needs to define how many resources it will consume in each shared VNF instance, which is not an easy task in a dynamic environment. Mohamed and Hassanein [29] computed the number of SFCs that can share the same VNF instances utilizing the number of SDN flows that ingress in the VNF instance. Unlike previous studies, our approach defines the limit to which SFCs can share the same VNF instances based on a value configured by the infrastructure owner during the VNF setup.

Unlike previously mentioned VNF placement solutions, we consider several aspects of formulating our solution for the VNF placement problem. We consider four communication delays in our modeling discussed in Section 3, namely transmission delay, propagation delay, processing delay, and queuing delay. Our approach also considers the energy consumption to select edge nodes to allocate VNFs. Additionally, we formulate the problem using the ILP model and present our smart VNF placement algorithm that includes all previously commented upon aspects.

3. Problem Statement and Modeling

This section presents the formulation of the VNF placement problem and our optimization model as a proposal to solve it.

3.1. Edge Environment Model

We propose a system model for the VNF placement problem that aims to minimize:

- The total latency from the moment of the user's request until the moment the user receives the result of the requested service;
- The energy consumption at the edge tier;
- The total cost of computing and network resources required to run the VNFs.

The model assumes that the VNF placement problem is performed at the edge tier. It also considers that each user is associated with only one edge node and the service demands come from users directly connected to an edge node. In addition, in our model, each VNF instance can handle a maximum number of requests. Such instances are prohibited from consuming more resources than the allocation defined during the VNF instance creation.

The system model represents the following entities and attributes. The physical infrastructure is represented as an undirected graph, $G = (H, L)$, where H is the set of

edge nodes that host virtual machines where VNF instances can be deployed; and L is the set of links between edge nodes. Each link between two edge nodes $h_i \in H$ and $h_j \in H$, represented by $l_{ij} \in L$, has a bandwidth $b(l_{ij})$, a delay $d(l_{ij})$, and a loss rate $w(l_{ij})$. The edge nodes comprehend a set of resources $\mathcal{A} = \{CPU, Memory\}$. Each resource is identified by the variable $R^{\alpha}(h)$ with $\alpha \in \mathcal{A}$.

The set of users is represented by U. According to our assumption, each user $u \in U$ is associated with only one edge node $h \in H$. This association is characterized by the following properties: bandwidth $b(l_{uh})$, delay $d(l_{uh})$, and loss rate $w(l_{uh})$ between user $u \in U$, and host $h \in H$.

The set of VNF types is represented by V. Each VNF type $v \in V$ has attributes $r^{\alpha}(v)$ which represents the CPU and memory allocated in host h during the creation of a VNF instance. We also define the set of VNF instances of a given type v as I_v. The attribute $max_{v_{inst}}$ defines the maximum number of SFC requests that the VNF instance v_{inst} can accept. The VNF instance is non-shareable if $max_{v_{inst}} = 1$.

The entity SFC type $s \in S$ is composed of a chain of VNF types, where S is the set of SFC types. Accordingly, we have instances of SFC types, which are represented by the set I_s of SFC instances of type s. Table 2 summarizes the entities and attributes of our approach.

Table 2. Entities and attributes notation.

Symbol	Description
	Entities
H	Set of edge nodes (hosts)
L	Set of links between edge nodes
U	Set of Users
V	Set of VNF types
I_v	Set of VNF instance of type v
S	Set of SFC types
I_s	Set of SFC instances of type s
	Edge Node (Host) Attributes
$\mathcal{A}$	Set of resource types
α	A specific type of resource in $\mathcal{A}$
$R^{\alpha}(h)$	The amount of resource type α of host h
TR_h	Total resources of host h
	User Attributes
$b(l_{uh})$	Bandwidth between user u and host h
$d(l_{uh})$	Delay between user u and host h
$w(l_{uh})$	Loss rate between user u and host h
S_u	Set of SFCs requested by user u
	VNF Type Attributes
v_{inst}	VNF instance of type v
IM_v	Set of hosts with the image of VNF type v
$r^{\alpha}(v)$	The demanded resource α by VNF type v
TR_v	Total required resources of VNF v
	SFC Attributes
d_{tol}	End-to-end delay tolerance of an SFC type
$b(s)$	Min bandwidth of SFC s
$jit(s)$	Jitter of SFC s
$SLA(s)$	SLA of SFC s

In the following discussion, we present the delay, energy, and QoS models that are considered in our VNF placement problem solution.

3.1.1. Delay Model

Our model considers the following delays: (i) propagation delay $d_p(l_{ij})$ between two edge nodes $h_i, h_j \in H$; (ii) transmission delay $d_{tm}(h_i, h_j)$, which represents the time needed to transfer packets from edge node h_i to h_j; (iii) and processing delay $d_{prc}(v_{inst}, h)$, which refers to the time needed to process the arriving packets in the VNF instance v_{inst}; and (iv) queuing delay $d_q(v_{inst}, h)$ of the VNF instance v_{inst}, which represents the time spent by packets at an edge node waiting for being sent to the next edge node after being processed by v_{inst}. Table 3 summarizes those different delays and their notations.

Table 3. Types of delay.

Symbol	Description
$d_p(l_{ij})$	Propagation delay over link l_{ij}
$d_{tm}(l_{ij})$	Transmission delay over link l_{ij}
$d_{prc}(v_{inst}, h)$	Processing delay of VNF instance v_i in edge node $h \in H$
$d_q(v_{inst}, h)$	Queuing delay in node $h \in H$ + queuing delay for a VNF instance v_i

3.1.2. Energy Model

In the energy model, we consider the energy consumed with communication among edge nodes and with VNF processing. Concerning the energy consumption with communication, we used the propagation delay with the power for edge nodes to send (or receive) data to (from) other edge nodes. Thus, we denote the communication energy consumption between two edge nodes $h_i, h_j \in H$ in Equation (1), where p_{ij} is the power consumption in the communication between h_i and h_j, and $d_p(l_{ij})$ is the propagation delay between the edge nodes.

$$E_{ij}^{Cm} = p_{ij} \times d_p(l_{ij}) \tag{1}$$

The energy consumption regarding the VNFs processing at the instant τ by an edge node h is defined in Equation (2), where p_h^{IDLE} is the consumed energy in the idle state of an edge node h at time τ, p_h^{MAX} is the consumed energy in the full state, and $u_h(\tau)$ is the utilization of edge node h at τ. Therefore, the energy consumed with processing in $h \in H$ at τ is defined in Equation (3).

$$p_h(\tau) = p_h^{IDLE} + (p_h^{MAX} - p_h^{IDLE}) \times u_h(\tau) \tag{2}$$

$$E_{s,u}^{prc} = \sum_{v_{inst}^{a,b} \in v_{inst}} \sum_{h \in H} p_h(\tau) \times \chi_h^{suv_{inst}}, \forall v \in V, s \in S_u, \forall u \in U \tag{3}$$

We compute the energy consumption regarding the propagation delay of a user u and service s at time τ, as described in Equation (4).

$$E_{s,u}^{p} = \sum_{v_{inst}^{a,b} \in I_v} \sum_{l_{ij} \in L} d_p(l_{ij}) \times p_{l_{ij}} \times y_{l_{ij}}^{v_{inst}^{a,b}}, \forall v \in V, s \in S_u, \forall u \in U \tag{4}$$

3.2. VNF Placement Problem Statement

The entities involved in the VNF placement problem are: (i) the users requesting new SFCs; (ii) the SFCs; (iii) the VNF instances; and (iv) the edge nodes.

The number of entities and their high resource heterogeneity and the dynamism of their availability make this problem even more challenging. Moreover, an edge node can execute multiple instances of one or more VNFs, and a single VNF instance can be shared by multiple applications. Additionally, every application request needs to traverse the VNF instances of the corresponding SFC.

Each user is associated with one or more SFC requests. An SFC request represents the request for a service, issued by a user and represents a chain of VNFs to be placed and executed in an edge environment.

For simplicity, in the remainder of the article, we use the term VNF to explicitly refer to a VNF type (e.g., firewall, proxy, etc.), and the term instance is always referred to the VNF type instantiated and running on an edge node.

3.3. ILP Formulation

This section describes the ILP formulation to the VNF placement problem in the edge tier upon the system model defined in Section 3.1. We propose an ILP formulation with three goals:

1. Minimizes the computational and communication resource usage;
2. Minimizes the end-to-end delay;
3. Minimizes the energy consumption.

Therefore, our formulation defines a multi-objective function that considers the delay, resource, and energy parameters. Table 4 presents the decision variables of the proposed ILP formulation.

Table 4. ILP Symbols and notation.

Symbol	Description
	Decision Variables
$\chi_h^{suv_{inst}}$	Binary variable set to 1 if VNF instance $v_{inst} \in I_v$ of service s from user u is executed in host h
$k_h^{v_{inst}}$	Binary variable set to 1 if VNF instance v_{inst} is executed in host h
	Network Traffic
$y_{l_{ij}}^{v_{inst}^{a,b}}$	The amount of traffic between VNF instances v_{inst}^a and v_{inst}^b that is routed over link $l_{ij} \in L$
	Resource
$R_{s,u}^{Cp}$	Total resource usage in the edge nodes by each VNF instance of all SFCs of user u
$R_{s,u}^{Cm}$	Total communication resources spent among VNF instances for each user SFC
	Delay
$D_{s,u}^{prc}$	Time spent for processing all VNF instances of all SFCs of user u
$D_{s,u}^{q}$	Time spent in the edge node queue for starting to process the VNF instances
$D_{s,u}^{p}$	Time spent for propagating data between the VNF instances that compose each SFC of users
$D_{s,u}^{tm}$	Time spent for transmitting data between the VNF instances that compose each SFC of users
	Energy consumption
E_{ij}^{Cm}	Energy consumed on communication
$E_{s,u}^{p}$	Energy consumed on transferring data
$E_{s,u}^{prc}$	Energy consumed on processing data

Based on our delay model and the mentioned decision variables, consider the computational resource $R_{s,u}^{Cp}$ defined in Equation (5), which sums the total resource usage on edge nodes for all VNF instances and all the SFCs of users. The communication resource $R_{s,u}^{Cm}$, defined in Equation (6), computes the total communication resources spent among all VNF instances of each SFC of the users. The processing delay $D_{s,u}^{prc}$, defined in Equation (7), calculates the time spent for processing VNF instances for each SFC of the users. The queuing delay $D_{s,u}^{q}$, defined in Equation (8), calculates the time spent on an edge node queue to start processing the VNF instances. The propagation delay $D_{s,u}^{p}$, defined in Equation (9), calculates the time spent by a packet data to traverse all VNF instances that compose each SFC of the users. The transmission delay $D_{s,u}^{tm}$, defined in Equation (10), calculates the time spent for transmitting data through all VNF instances that compose each SFC of the users.

$$R_{s,u}^{Cp} = \sum_{h \in H} \sum_{v_{inst}^h \in I_v} \frac{R_{v_{inst}^h}}{TR_h} \times k_h^{v_{inst}}, \forall v \in V, \forall s \in S_u, \forall u \in U \quad (5)$$

$$R_{s,u}^{Cm} = \sum_{v_{inst}^{a,b} \in I_v} \sum_{l_{ij} \in L} \frac{y_{l_{ij}}^{v_{inst}^{a,b}}}{b(l_{ij})}, \forall v \in V, \forall s \in S_u, \forall u \in U \tag{6}$$

$$D_{s,u}^{prc} = \sum_{v_{inst} \in I_v} \sum_{h \in H} d_{prc}(v_{inst}, h) \times \chi_h^{suv_{inst}}, \forall v \in V, \forall s \in S_u, \forall u \in U \tag{7}$$

$$D_{s,u}^{q} = \sum_{v_{inst} \in I_v} \sum_{h \in H} d_{q}(v_{inst}, h) \times \chi_h^{suv_{inst}}, \forall v \in V, \forall s \in S_u, \forall u \in U \tag{8}$$

$$D_{s,u}^{p} = \sum_{v_{inst}^{a,b} \in I_v} \sum_{l_{ij} \in L} d_{p}(l_{ij}) \times y_{l_{ij}}^{v_{inst}^{a,b}}, \forall v \in V, \forall s \in S_u, \forall u \in U \tag{9}$$

$$D_{s,u}^{tm} = \sum_{v_{inst}^{a,b} \in I_v} \sum_{l_{ij} \in L} d_{tm}(l_{ij}) \times y_{l_{ij}}^{v_{inst}^{a,b}}, \forall v \in V, \forall s \in S_u, \forall u \in U \tag{10}$$

Finally, the multi-objective function is defined in Equation (11), which aims to minimize the sum of the queuing and transmission delays, the energy consumption of processing VNFs, and transmitting VNF data from edge nodes to their outgoing links. The processing and propagation delays are omitted, since they are embedded in the energy consumption calculation.

$$minimize \sum_{u \in U} \sum_{s \in S_u} D_{s,u}^{q} + D_{s,u}^{tm} + E_{s,u}^{p} + E_{s,u}^{prc} + R^{Cp} + R^{Cm} \tag{11}$$

Equation (11) is subject to the following constraints:

- VNF instances $v_{inst} \in I_v$ will be placed only on a node h that can offer the VNF type $v \in V$, as described in Equation (12):

$$\sum_{h \notin IM_v} \chi_h^{suv_{inst}} = 0, \forall v_{inst} \in I_v, v \in V, s \in S, u \in U \tag{12}$$

- The resources of edge nodes cannot be over-allocated, as defined in Equation (13);

$$\sum_{v \in V} TR_v \times \chi_h^{suv_{inst}} \leq TR_h, \forall h \in H \tag{13}$$

The proposed ILP model for the VNF placement is an NP-hard problem. Therefore, to solve this problem, we propose a heuristic-based algorithm, as described in the following section.

4. Challenges of VNF Placement in Edge Environments

Placing different types of VNFs at the edge tier should consider SFC requirements, users' SLA, and resources available in the infrastructure. Therefore, VNF placement must deal with the heterogeneity of edge nodes' computational capacity and SFC requirements. Moreover, the available resources in edge nodes are dynamic since the CPU and memory utilization increases or decreases when VNF instances are allocated or deallocated. In addition, VNFs consume bandwidth to communicate with other VNFs that make part of an SFC when placed in different edge nodes.

Another concern in the VNF placement problem is the optimization of resource usage. This is essential to serve the maximum number of users without violating their SLAs. Therefore, the following challenges are highlighted: the decision-making for deploying VNFs considering the VNF requirement and SLAs, the heterogeneity and dynamism of available resources in edge nodes, and the heterogeneity of SFC requirements. We discuss each of them in more detail in the following sections.

4.1. Decision Making for Deploying VNFs Considering VNF Requirements and SLA

When deploying VNF instances into the edge tier, one should select the most suitable edge nodes so that SFC packets can traverse VNFs to meet the users' requests without violating the established SLAs. An SLA is defined per SFC and comprises the maximum end-to-end packet delay. The packet delay is the difference between the time a packet enters the first VNF and its processing in the last VNF of the corresponding SFC.

Additionally, given the dynamism of the workload and the potentially high number of edge nodes in the network, the placement decision needs to be done automatically. Furthermore, this selection should consider the VNF requirements and optimize the use of service provider resources. Such requirements include CPU and memory for processing VNFs in edge nodes and bandwidth regarding the traffic between VNFs that compose an SFC. Several VNF placement solutions face these challenges [33–38] and aim to overcome them to provide an optimal or sub-optimal approach.

4.2. Heterogeneity and Dynamism of Resources in Edge Environments

Edge nodes must provide the computational resources required by VNFs to host them successfully. In other words, edge nodes must offer sufficient CPU and memory capacity to provide the minimum resources required by VNFs. Additionally, edge nodes need to be able to provide the VNF type that is being requested. At the edge tier, there are edge nodes with different computational resource capacities that can provide different types of VNFs. Therefore, the edge environment has highly heterogeneous available resources (CPU and memory) among nodes.

Furthermore, VNFs must deal with various link bandwidth capacities among those nodes. Link bandwidth is an essential resource to be provided to SFCs since they comprise a set of chained VNFs, which must be processed in a predefined order for the success of the SFC processing. Thus, not only the computational resources of edge nodes must be capable of meeting VNF requirements, but also link bandwidth among them must be able to forward the traffic to the chained VNFs.

Moreover, edge nodes vary their computational capacity as they host new instances and free up resources when some instances are removed. Consequently, the link bandwidth varies as well. Such dynamism of computational resources and link bandwidth, added to the heterogeneity of nodes and links, makes the VNF placement a complex task.

4.3. Heterogeneity of SFC Requirements

In addition to the variety of computational resource capacities that edge nodes provide, SFCs and their VNFs also present different requirements. SFCs can demand different end-to-end delay tolerance, bandwidth, jitter, and SLA according to their VNFs and service types. Furthermore, the same SFC may have different requirements depending on the user requesting it. Moreover, VNFs of the same SFC may require different CPU and/or memory capacities according to their roles in the SFC.

SFCs also have multiple and conflicting priorities that should be addressed in the VNF placement solution. These priorities regard latency, capacity, and energy consumption. If an SFC prioritizes latency, the VNF placement approach should be aware of selecting edge nodes that better satisfy the VNF requirements regarding the delay for executing the SFC. Then, the other priorities should also be verified to serve the SFC according to the priority order. Therefore, these different priorities further contribute to increasing the heterogeneity of SFCs.

5. A Novel Smart VNF Placement Algorithm

The proposed heuristic-based algorithm for VNF placement determines to which VNF instance, hosted by an edge node, the VNFs associated with a user SFC Request should be mapped and which link between the nodes that host such VNFs should be used. Finding a solution for the VNF placement problem by considering all SFCs that arrived within a time window in the decision increases the complexity of the problem due to the variety of

combinations of possible placement plans. Thus, adopting an optimal approach, such as that based on ILP, is unfeasible due to the time taken to find the placement plan.

Once the SFC to be placed is chosen, the heuristic needs to select the chain of VNF instances that will be considered to meet the SFC request. Then, following the VNF order in the SFC, the heuristic either reuses a VNF instance that is already running in an edge node or creates a new one. In other words, the heuristic algorithm has two phases: phase 1, which exploits already-placed VNF instances, and phase 2, which places new VNF instances. Figure 1 illustrates an overview of each step of the proposed heuristic.

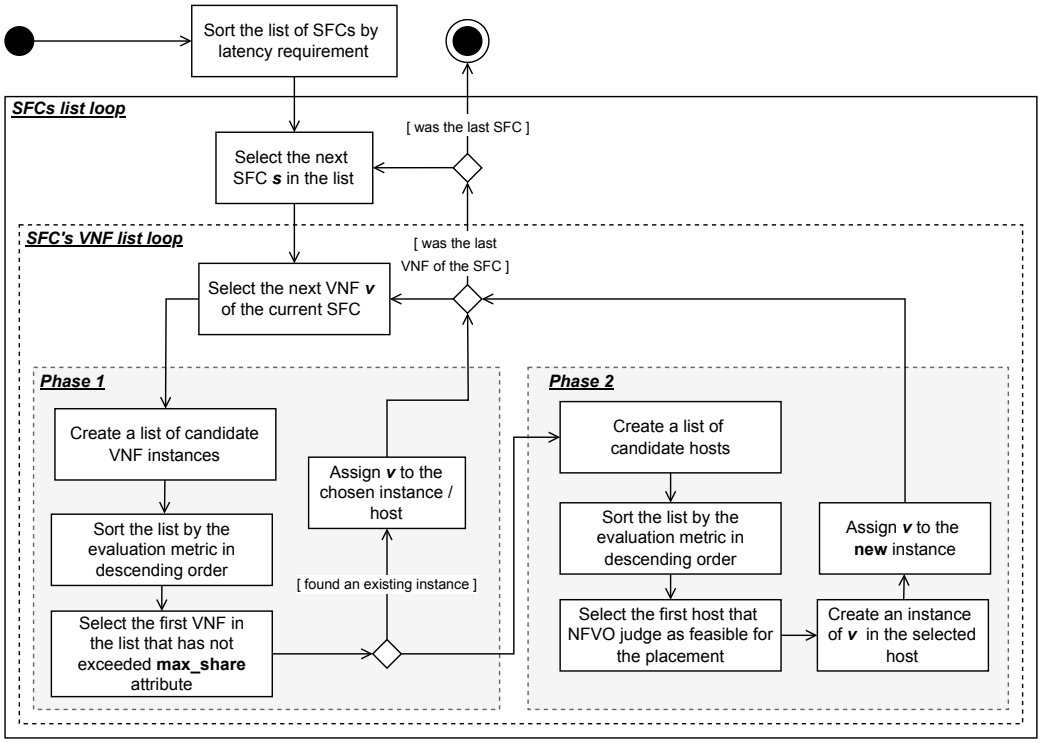

Figure 1. Smart VNF placement algorithm.

Both phases rely on an evaluation metric to sort a list of "candidate entities" with which the current VNF will associate. These "candidate entities" are defined according to the execution phase of the smart VNF placement algorithm. In ***Phase 1*** , a candidate entity is an existing VNF instance that has not exceeded the maximum share. In ***Phase 2***, a candidate entity is an edge node that the NFVO has verified as feasible to place the requested VNF.

The evaluation metric proposed in this paper is a convex combination of the following attributes of the entity e: latency delay (L_e), resource capacity availability (R_e), and energy availability (E_e), as defined in Equation (14), where $w_1 + w_2 + w_3 = 1$ and $w_i \geq 0$ with $i = 1, 2, 3$. Note that the weights are assigned by the network administrator to set up the heuristic to prioritize one attribute over the others. Alternatively, this setup can be performed by an autonomous agent, but it is out of the scope of this article.

$$Eval(e) = w_1 \times E_e + w_2 \times R_e + w_3 \times L_e \tag{14}$$

The novel VNF placement algorithm allows resource sharing among multiple VNF instances by exploiting already-placed VNF instances before creating new ones. This resource sharing among VNFs can reduce the computational resource usage since the request uses the resource allocated to the instance serving it. Furthermore, this novel aspect can optimize the use of well-positioned nodes by concentrating several requests in the same node. Figure 2 depicts a scenario composed by users u_1, u_2, and u_3, requesting SFCs s_1 and s_2. The placement plan for request (u_1, s_1) has decided to create new VNF instances

of VNFs ○ and △ in node h_1, and VNF □ in node h_3. The instance of VNF □ is reused in the placement plan for request (u_2, s_2), since it can be shared by two requests at most, while VNF ○ is created in node h_2. Once the instance of VNF □ in node h_3 reaches its maximum share, the placement plan for request (u_3, s_2) has to create a new instance of VNF □ in node h_4.

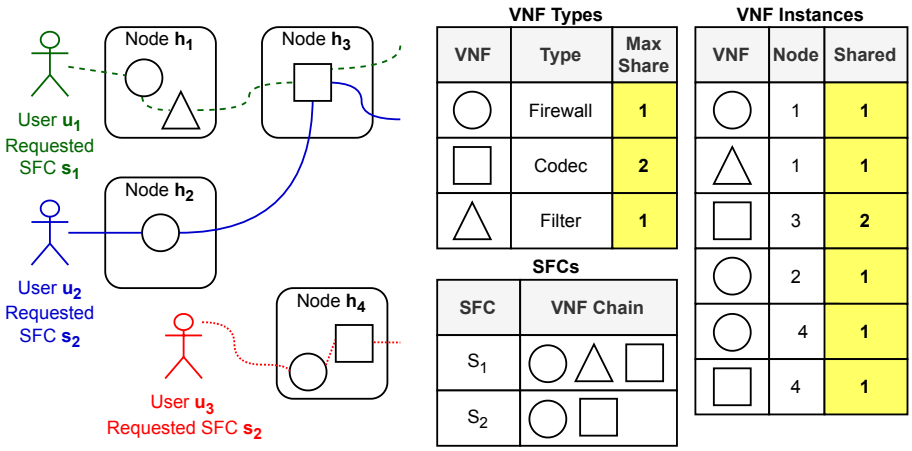

Figure 2. VNF sharing during phase 1 of the smart VNF placement algorithm.

The proposed algorithm also targets a different application QoS while decreasing the cost for the infrastructure provider. To this end, the algorithm is aware of latency, computational resource capacity, and energy consumption. Therefore, it selects the edge nodes for placing VNF requests by verifying those characteristics in the candidate edge nodes. Each SFC has a list in our model that defines the priority order considering latency, capacity, and energy. Thus, the algorithm searches for candidate nodes according to the highest priority requirement (among energy consumption, latency, and resource consumption) for each SFC request. For instance, if an SFC first prioritizes energy consumption, the heuristic algorithm searches for nodes that consume less energy than the other nodes. SFCs have different QoS requirements related to end-to-end delay tolerance and bandwidth. Thus, the balance of those multiple QoS requirements is crucial for the VNF placement problem. The proposed heuristic algorithm allows this balance by prioritizing one of the requirements according to SFC demand. Moreover, the solution can prioritize applications according to the users' payment plan.

6. Implementation Architecture

Many architectures were proposed to manage VNFs in edge environments [39]. In particular, the European Telecommunications Standards Institute (ETSI) [40] proposed the open source NFV management and orchestration (OSM) that is a reference architecture for multi-access edge computing (MEC). The document describes an MEC system that enables MEC applications to run efficiently and seamlessly in a multi-access network. We consider SFCs as MEC applications.

The architecture proposed in this study complies with [40]. Figure 3 shows a complete view of the physical devices and software components of the proposed architecture, with the MEC orchestrator in charge of defining in which edge node each VNF of an SFC must be placed.

Leveraging the standard ETSI architecture, we change the component *VNF placement* and propose a new component called *VNF mapping*. These components work within the MEC orchestrator module that already exists in the ETSI architecture. The *VNF mapping* component is used to map the VNFs demanded by the requested SFC to already created VNF instances during the VNF placement. The VNF placement component will implement our proposed VNF placement heuristic described in Section 5.

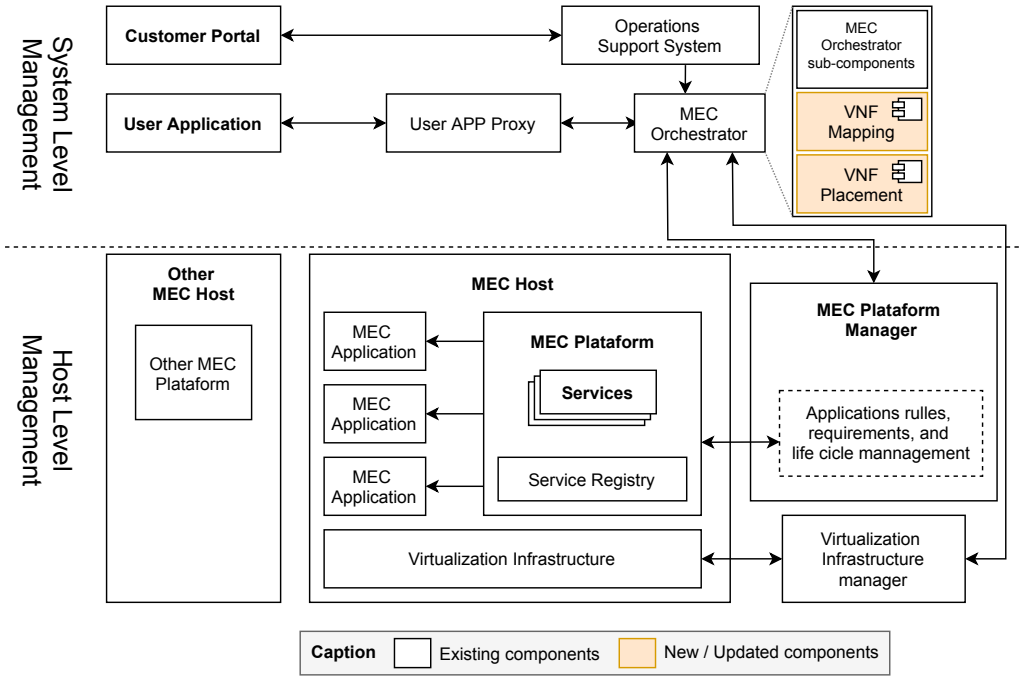

Figure 3. Architecture components of our approach.

7. Experimental Results

In this section, we cover the simulation environment and describe the performed experiments. Then, we analyze the obtained results, comparing our heuristic-based algorithm for VNF placement with three approaches, namely (i) the solution provided by the ILP solver; (ii) a greedy; and (iii) a random algorithm.

The objective of the evaluation was to compare the performance of our heuristic-based algorithm against the other previously mentioned proposals concerning the results produced by each proposal in terms of the amount of energy and resources (CPU and memory) consumed by the execution of the generated placement plan. In addition, we executed a set of tests to verify the percentage of SFCs served by our smart VNF placement algorithm and the greedy and random algorithms.

7.1. Simulation Environment

To perform the experiments, we implemented environment entities, such as edge nodes, VNFs, SFCs, links, packets, users, and data sources, as described in the model presented in Section 3.1. Similar to [41,42], we implemented our own simulation tool since other simulation frameworks do not support the granularity level (dynamic packet size) and do not offer support for new environment features, such as VNF sharing required for our experiments. Simulations were executed in a server equipped with an Intel(R) Xeon(R) Silver 2.20 GHz 48 cores processor and 500 GB of RAM. We used Python as a programming language to code our simulation, along with the simulation framework SimPy. The same programming stack was also used in the 5G simulation framework called 5GPy [43].

To define the edge environment entities' configuration, we considered a virtual scenario compatible with the 5G use case called *best effort*. In this 5G scenario, the resource requirements are more flexible and less restrictive in comparison with the other 5G 3GPP scenarios [44]. This represents mixed applications with a variety of QoS requirements. Besides that, in the experiments, we consider that the edge nodes are arranged in a full mesh network topology, i.e., each edge node is directly connected to all the others.

All the experiments were conducted using the parameters presented in Table 5. The CPU resource at the edge node and VNF type description is defined in terms of instructions per second (IPS). Each experiment was executed 10 times to increase the confidence interval, we

adopted a 5% error tolerance. We analyzed the number of SLA violations in terms of packet delay, the number of accepted SFC requests, as well as the energy and resource consumption.

Table 5. Simulation parameters.

Parameter	Value
Edge nodes	32
Edge node CPU	5000 (IPS)
Edge node Mem	10 GB
VNF types	3
VNF min CPU	1000 (IPS)
VNF min Mem	3000
SFC types	3
VNFs in SFC	2, 3
Number of users	1000
Number of SFC requests	1000
SFC instance timeout	2 min
VNF instance timeout	2 min
Packet size	50 IP
VNF max share	3 SFC Requests

Since the VNF placement problem is NP-hard, we created an ILP solver that implements the objective function defined in Equation (11) able to find feasible solutions only for small-scale edge systems. As the ILP solver presents poor scalability in terms of the number of nodes and SFC requests, we initially compare our proposed heuristic-based algorithm with the optimal solution in smaller scenarios.

Besides that, we compared our approach with two other algorithms widely used as benchmarks in the literature, namely, greedy and random [17,18,28], which are described below. The energy parameters for edge nodes consumption follow the specifications defined in [45].

- The **Greedy placement** first selected entities (edge nodes and links) with the higher resources available to compose the placement plan. This strategy tends to avoid selecting overloaded nodes and links.
- The **Random placement** randomly chooses a node among available nodes to create the new VNF instances.

7.2. ILP Solver vs. Smart Heuristic

This section presents the results obtained when comparing the ILP solver with the heuristic proposed in this work. Due to the model scalability and time constraints, it took approximately 50 min to obtain a solution considering a scenario with just a few requests, even executing the model in a server equipped with 48 cores and 500 GB of RAM. Hence, we only consider 10 SFC requests in the experiment, given that the time to find the best solution increases as the number of requests grows.

Figure 4 depicts the energy consumed in the nodes to execute the requested SFCs. Regarding energy consumption, the ILP tends to select nodes that consumed less energy to place the VNF instances in comparison with our heuristic.

In addition to energy consumption, we also compare the ILP solver and the heuristic approaches regarding resource consumption (i.e., CPU and memory). Analyzing Figures 5 and 6, we can see that the optimal solution and the heuristic approach performed similarly in terms of resource consumption. Although the resource consumption obtained by the ILP was slightly lower, the heuristic approach can provide a good (but not optimal) solution in a shorter time than the ILP, even in scenarios with a higher number of nodes. This fast response time is essential considering that one of the appeals of using edge computing is lower latency for applications.

Figure 4. Consumed energy in edge nodes.

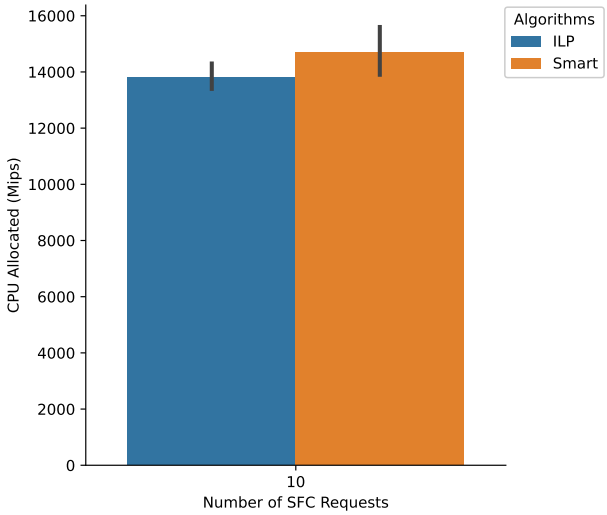

Figure 5. Allocated CPU Resources.

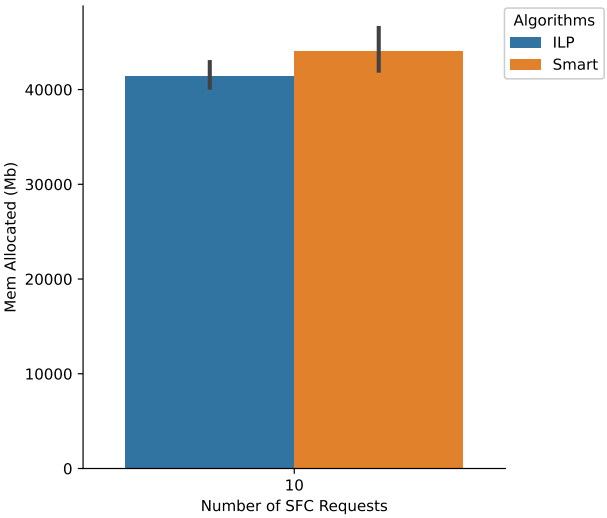

Figure 6. Allocated memory resources.

7.3. Number of Placed SFC Requests

The rationale behind such an experiment is to verify whether the proposed smart VNF placement algorithm increases the number of placed SFC requests while overloading less resources in comparison with random and greedy approaches.

We consider an SFC request as placed if the placement algorithm can create a feasible placement plan to allocate the VNFs of the related SFC.

In Figure 7, as the number of SFC requests increases in the edge environment, the percentage of requests that are placed decreases since the resources become depleted, given that the available resources of the environment are constant.

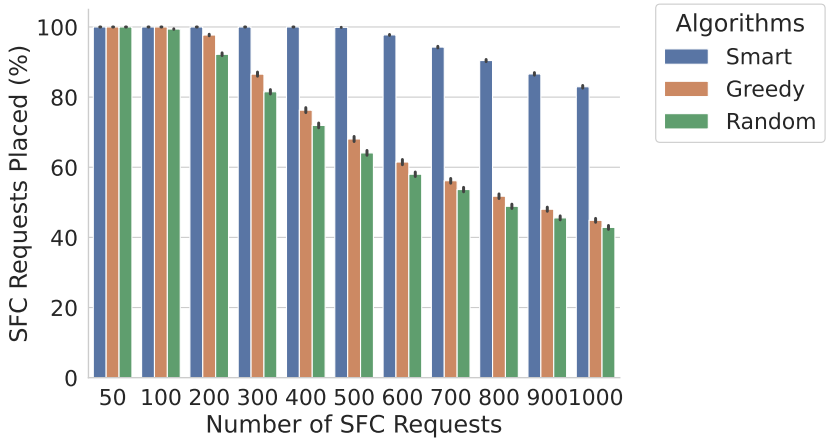

Figure 7. Percentage of placed SFC requests.

Our approach always performs better than the other approaches, even though it is also impacted by the resource shortage. For instance, in a highly demanding scenario with 1000 SFC requests in the edge environment, our placement strategy outperforms greedy and random by approximately 50%.

We also observe that both greedy and random support up to 100 SFC requests in the environment with 100% of placed requests, while our approach endures up to 500 requests with all requests being placed.

With such results, we observe that our Smart Placement algorithm is able to efficiently find placement plans that can meet SFCs, even in highly demanding situations. We will see in the following experiments that this is not the only benefit that our approach brings.

7.4. Energy Consumption

In this experiment, we aim to compare the total energy consumed by each placement algorithm and verify whether our approach can generate energy-efficient placement plans.

We observe in Figure 8 that our approach spends less energy than the other approaches, while it delivers more placed requests (see Figure 7). The VNF sharing mechanism helps to achieve such a result, since fewer edge nodes are demanded to meet a given number of requests, while greedy and random generally choose VNF instances spread in more edge nodes.

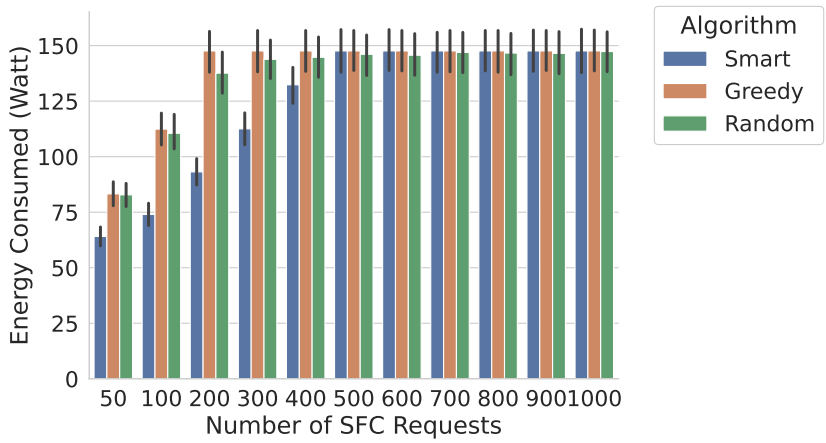

Figure 8. Consumed energy in edge nodes.

As the resources of the environment become exhausted with more than 400 requests, all placement strategies spend the same amount of energy, which is the expected behavior for the proposed extreme scenario.

7.5. Resource Consumption

In this experiment, we compare the total allocated resources by the placement algorithms in terms of CPU and memory.

Analyzing Figures 9 and 10, we realize that our approach outperforms the other approaches by allocating less CPU and memory to meet a given number of SFC requests. Again, the VNF sharing mechanism allows our approach to be a resource-allocation efficient.

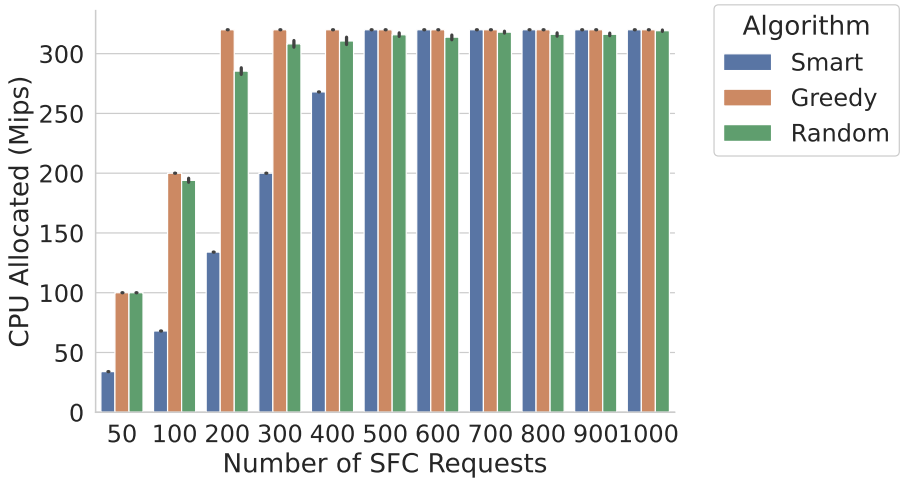

Figure 9. Allocated CPU resources.

This observed behavior of our proposed placement strategy also corroborates with the previously presented experiment. It shows that, besides being energy-efficient in terms of less nodes being used, our approach is also energy-efficient in terms of less resource consumption of edge nodes.

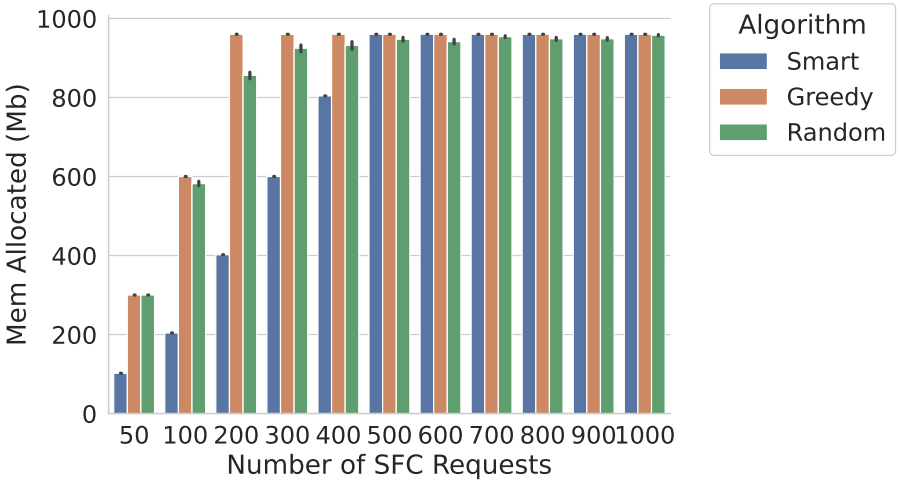

Figure 10. Allocated memory resources.

7.6. Packet Delay

The rationale behind such an experiment is to verify whether the proposed placement algorithm influences the total packet processing time throughout the VNFs of the SFCs in comparison with the other approaches. The goal is to verify whether the adoption of our

approach impacts the number of packets that do not violate the maximum tolerable packet delay, which we consider as the definition of the SLA of an SFC.

We see in Figure 11 that the percentage of the packets within the maximum tolerable delay can be considered constant for all three compared approaches. Although greedy and random presented a higher percentage than our approach, analyzing Figure 7, we observe that our approach is actually placing a greater number of requests than the other two algorithms, especially in extreme scenarios, such as with 1000 requests in the environment. Hence, our smart placement algorithm meets the demanded SFC requirements for a larger number of delivered requests.

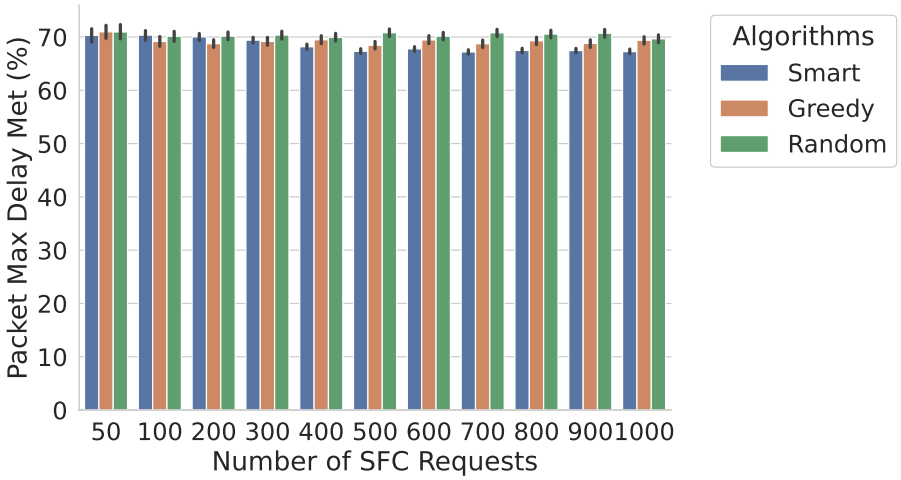

Figure 11. Percentage of packets that do not exceed the maximum tolerable delay.

7.7. Results Analysis

Throughout the results obtained in previously presented experiments, we can observe that our approach performs better than the Greedy and Random approaches.

As seen in Figure 7, the number of SFC requests placed is higher than the other approaches, even with less consumed resources in the edge node, as depicted in Figures 9 and 10.

We can also identify in Figures 7, 9 and 10 that, in an extreme edge environment considering the scenarios of depletion of all available resources, our approach maintains a higher rate of placed SFC requests, while both greedy and random approaches stop placing new SFC requests (see Figure 7).

Figure 11 illustrates that, besides sharing VNFs and consuming fewer resources and energy in the edge nodes, our proposal was capable of finding placement plans that were within the maximum tolerable delay specified by the SFCs.

8. Conclusions

In this article, we presented our solution for the VNF placement problem based on its formulation as an optimization problem, solved through ILP. The VNF placement problem is NP-hard, and then optimal approaches, such as ILP, can find feasible solutions only for small-scale edge systems and it presents poor scalability. Therefore, we also proposed a heuristic approach to address large-scale systems. The described ILP model acts as a solver for small-scale edge environments, and a benchmark for the heuristic solution.

The novel heuristic algorithm allows resource sharing by exploiting already-placed VNF instances before creating new ones. The algorithm is based on a system model, which defines an attribute to specify the maximum number of VNF requests an instance of the related VNF type can accept simultaneously. When a VNF instance accepts more than one VNF request, those requests share the computational resource allocated for that instance in the edge node. Such resource sharing among VNFs can reduce computational resource usage since the request takes advantage of the already allocated resources. Furthermore,

such a novel aspect can optimize the use of well-positioned nodes by concentrating VNFs from several requests in the same node.

Our proposal also targets different application QoS requirements while decreasing costs for the infrastructure provider. To this end, the algorithm is aware of latency, computational resource capacity, and energy consumption. Therefore, it selects edge nodes to place VNF requests by verifying those characteristics in the candidate edge nodes. In contrast with most VNF placement solutions, we consider four communication delays in our delay model, providing accurate decisions regarding latency. The energy consumption of edge nodes is essential for guaranteeing QoS due to their power constraints. This consumption includes, for example, data generation, transmission, aggregation, storage, and processing tasks.

As future work, we envision (i) leveraging our model to support the continuous arrivals of user SFC requests over time; (ii) enhancing the proposal with the feature of being aware of user mobility and with actions that might be taken as a result of such user movements; and (iii) to create a mechanism to dynamically change the limit of SFCs that can share the same VNF instance based on the current workload.

Author Contributions: Writing—original draft, methodology, software, validation, and formal analysis, A.L.É.B., E.L.C.M., M.I.P.J., H.B. and D.P.d.M.; Writing—review and editing F.C.D. and P.F.P.; Supervision, and project administration, A.C.B.d.O., F.C.D., P.F.P. and D.C.M.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by DELL EMC, and by the Brazilian funding agencies Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), CAPES-Print, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq—grant number 306747/2018-9), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP—grant number 2015/24144-7), and Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ). Débora Muchaluat-Saade, Flávia C. Delicato, and Paulo F. Pires are CNPq Fellows.

Data Availability Statement: Not Applicable, the study does not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gil Herrera, J.; Botero, J.F. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 518–532. [CrossRef]
2. Bhamare, D.; Jain, R.; Samaka, M.; Erbad, A. A survey on service function chaining. *J. Netw. Comput. Appl.* **2016**, *75*, 138–155. [CrossRef]
3. Yu, W.; Liang, F.; He, X.; Hatcher, W.G.; Lu, C.; Lin, J.; Yang, X. A Survey on the Edge Computing for the Internet of Things. *IEEE Access* **2017**, *6*, 6900–6919. [CrossRef]
4. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1657–1681. [CrossRef]
5. Zhang, C.; Joshi, H.P.; Riley, G.F.; Wright, S.A. Towards a virtual network function research agenda: A systematic literature review of VNF design considerations. *J. Netw. Comput. Appl.* **2019**, *146*, 102417. [CrossRef]
6. Yi, B.; Wang, X.; Li, K.; Das, S.k.; Huang, M. A comprehensive survey of Network Function Virtualization. *Comput. Netw.* **2018**, *133*, 212–262. [CrossRef]
7. Jaeggi, D.; Parks, G.; Kipouros, T.; Clarkson, P. The development of a multi-objective Tabu Search algorithm for continuous optimisation problems. *Eur. J. Oper. Res.* **2008**, *185*, 1192–1212. [CrossRef]
8. Zitzler, E.; Deb, K.; Thiele, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evol. Comput.* **2000**, *8*, 173–195. [CrossRef]
9. Benkacem, I.; Taleb, T.; Bagaa, M.; Flinck, H. Optimal VNFs Placement in CDN Slicing Over Multi-Cloud Environment. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 616–627. [CrossRef]
10. Zhao, D.; Liao, D.; Sun, G.; Xu, S. Towards resource-efficient service function chain deployment in cloud-fog computing. *IEEE Access* **2018**, *6*, 66754–66766. [CrossRef]
11. Leivadeas, A.; Falkner, M.; Lambadaris, I.; Kesidis, G. Optimal virtualized network function allocation for an SDN enabled cloud. *Comput. Stand. Interf.* **2017**, *54*, 266–278. [CrossRef]
12. Tseng, C.W.; Tseng, F.H.; Yang, Y.T.; Liu, C.C.; Chou, L.D. Task Scheduling for Edge Computing with Agile VNFs On-Demand Service Model toward 5G and Beyond. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 7802797. [CrossRef]

13. Abu-Lebdeh, M.; Naboulsi, D.; Glitho, R.; Tchouati, C.W. NFV orchestrator placement for geo-distributed systems. In Proceedings of the 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 30 October–1 November 2017; pp. 1–5. [CrossRef]
14. Ruiz, L.; Barroso, R.J.; De Miguel, I.; Merayo, N.; Aguado, J.C.; De La Rosa, R.; Fernández, P.; Lorenzo, R.M.; Abril, E.J. Genetic algorithm for holistic VNF-mapping and virtual topology design. *IEEE Access* **2020**, *8*, 55893–55904. [CrossRef]
15. Kim, S.I.; Sung Kim, H. A VNF Placement Method Considering Load and Hop count in NFV Environment. *Int. Conf. Inf. Netw.* **2020**, *2020*, 707–712. [CrossRef]
16. Emu, M.; Yan, P.; Choudhury, S. Latency aware VNF deployment at edge devices for IoT services: An artificial neural network based approach. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
17. Garrich, M.; Romero-Gazquez, J.L.; Moreno-Muro, F.J.; Hernandez-Bastida, M.; Delgado, M.V.B.; Bravalheri, A.; Uniyal, N.; Muqaddas, A.S.; Nejabati, R.; Casellas, R.; et al. IT and Multi-layer Online Resource Allocation and Offline Planning in Metropolitan Networks. *J. Light. Technol.* **2020**, *38*, 3190–3199. [CrossRef]
18. Li, D.; Hong, P.; Xue, K.; Pei, J. Virtual network function placement and resource optimization in NFV and edge computing enabled networks. *Comput. Netw.* **2019**, *152*, 12–24. [CrossRef]
19. Nguyen, D.T.; Pham, C.; Nguyen, K.K.; Cheriet, M. Placement and Chaining for Run-Time IoT Service Deployment in Edge-Cloud. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 459–472. [CrossRef]
20. Reyhanian, N.; Farmanbar, H.; Mohajer, S.; Luo, Z.Q. Joint Resource Allocation and Routing for Service Function Chaining with In-Subnetwork Processing. In Proceedings of the ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 4985–4989.
21. Kiran, N.; Liu, X.; Wang, S.; Yin, C. VNF Placement and Resource Allocation in SDN/NFV-Enabled MEC Networks. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Seoul, Republic of Korea, 6–9 April 2020; pp. 1–6. [CrossRef]
22. Pei, J.; Hong, P.; Pan, M.; Liu, J.; Zhou, J. Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 263–278. [CrossRef]
23. Bunyakitanon, M.; Vasilakos, X.; Nejabati, R.; Simeonidou, D. End-to-End Performance-Based Autonomous VNF Placement with Adopted Reinforcement Learning. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *6*, 534–547. . 2988486. [CrossRef]
24. Pham, C.; Tran, N.H.; Ren, S.; Saad, W.; Hong, C.S. Traffic-Aware and Energy-Efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach. *IEEE Trans. Serv. Comput.* **2020**, *13*, 172–185. [CrossRef]
25. Li, S.; Zhang, S.; Chen, L.; Chen, H.; Liu, X.; Lin, S. An Attention Based Deep Reinforcement Learning Method for Virtual Network Function Placement. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications, ICCC 2020, Chengdu, China, 11–14 December 2020; pp. 1005–1009. [CrossRef]
26. Niu, M.; Cheng, B.; Chen, J.L. GPSO: A graph-based heuristic algorithm for service function chain placement in data center networks. In Proceedings of the 2020 IEEE 13th International Conference on Services Computing, SCC 2020, Beijing, China, 7–11 November 2020; pp. 256–263. [CrossRef]
27. Alahmad, Y.; Agarwal, A.; Daradkeh, T. Cost and Availability-Aware VNF Selection and Placement for Network Services in NFV. In Proceedings of the 2020 International Symposium on Networks, Computers and Communications, ISNCC 2020, Montreal, QC, Canada, 20–22 October 2020; pp. 1–6. [CrossRef]
28. Gao, T.; Li, X.; Wu, Y.; Zou, W.; Huang, S.; Tornatore, M.; Mukherjee, B. Cost-Efficient VNF Placement and Scheduling in Public Cloud Networks. *IEEE Trans. Commun.* **2020**, *68*, 4946–4959. [CrossRef]
29. Mohamad, A.; Hassanein, H.S. On Demonstrating the Gain of SFC Placement with VNF Sharing at the Edge. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]
30. Malandrino, F.; Chiasserini, C.F.; Einziger, G.; Scalosub, G. Reducing Service Deployment Cost Through VNF Sharing. *IEEE/ACM Trans. Netw.* **2019**, *27*, 2363–2376. [CrossRef]
31. Mohamad, A.; Hassanein, H.S. PSVShare: A Priority-based SFC placement with VNF Sharing. In Proceedings of the 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2020, Leganes, Spain, 10–12 November 2020; pp. 25–30. ISBN: 9781728181592. [CrossRef]
32. Guo, H.; Wang, Y.; Li, Z.; Qiu, X.; An, H.; yu, P.; Yuan, N. Cost-aware Placement and Chaining of Service Function Chain with VNF Instance Sharing. In Proceedings of the NOMS 2020—2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 20–24 April 2020; pp. 1–8. [CrossRef]
33. Hawilo, H.; Jammal, M.; Shami, A. Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 643–655. [CrossRef]
34. Dinh-Xuan, L.; Seufert, M.; Wamser, F.; Tran-Gia, P.; Vassilakis, C.; Zafeiropoulos, A. Performance Evaluation of Service Functions Chain Placement Algorithms in Edge Cloud. In Proceedings of the 2018 30th International Teletraffic Congress (ITC 30), Vienna, Austria, 3–7 September 2018; Volume 1, pp. 227–235. [CrossRef]
35. Luizelli, M.C.; Bays, L.R.; Buriol, L.S.; Barcellos, M.P.; Gaspary, L.P. Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 98–106. [CrossRef]

36. Hmaity, A.; Savi, M.; Askari, L.; Musumeci, F.; Tornatore, M.; Pattavina, A. Latency- and capacity-aware placement of chained Virtual Network Functions in FMC metro networks. *Opt. Switch. Netw.* **2020**, *35*, 100536. [CrossRef]
37. Leivadeas, A.; Kesidis, G.; Ibnkahla, M.; Lambadaris, I. VNF Placement Optimization at the Edge and Cloud. *Future Internet* **2019**, *11*, 69. [CrossRef]
38. Castañeda, J.; Pomares Hernández, S.E.; Yangui, S.; Pérez Sansalvador, J.C.; Rodríguez Henríquez, L.M.; Drira, K. VNF-based network service consistent reconfiguration in multi-domain federations: A distributed approach. *J. Netw. Comput. Appl.* **2021**, *195*, 103226. [CrossRef]
39. Mamushiane, L.; Lysko, A.A.; Mukute, T.; Mwangama, J.; Toit, Z.D. Overview of 9 Open-Source Resource Orchestrating ETSI MANO Compliant Implementations: A Brief Survey. In Proceedings of the 2019 IEEE 2nd Wireless Africa Conference, WAC 2019, Pretoria, South Africa, 18–20 August 2019; pp. 1–7. [CrossRef]
40. ETSI. *MEC 003—V2.1.1–Multi-access Edge Computing (MEC); Framework and Reference Architecture*; Technical Report; ETSI: Valbonne, France, 2019.
41. Calheiros, R.N.; Ranjan, R.; De Rose, C.A.; Buyya, R. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv* **2009**. arXiv:0903.2525.
42. Lera, I.; Guerrero, C.; Juiz, C. YAFS: A Simulator for IoT Scenarios in Fog Computing. *IEEE Access* **2019**, *7*, 91745–91758. [CrossRef]
43. Tinini, R.I.; dos Santos, M.R.P.; Figueiredo, G.B.; Batista, D.M. 5GPy: A SimPy-based simulator for performance evaluations in 5G hybrid Cloud-Fog RAN architectures. *Simul. Model. Pract. Theory* **2020**, *101*, 102030. [CrossRef]
44. Ghosh, A.; Maeder, A.; Baker, M.; Chandramouli, D. 5G evolution: A view on 5G cellular technology beyond 3GPP release 15. *IEEE Access* **2019**, *7*, 127639–127651. [CrossRef]
45. Shehabi, A.; Smith, S.J.; Sartor, D.A.; Brown, R.E.; Herrlin, M.; Koomey, J.G.; Masanet, E.R.; Horner, N.; Azevedo, I.L.; Lintner, W. United States Data Center Energy Usage Report. Berkeley Lab. 2016; p. 65. Available online: https://www.osti.gov/servlets/purl/1372902 (accessed on 24 October 2022).