Energy-Efficient Virtual Network Function Placement Based on Metaheuristic Approaches

Fatemeh Mosaiyebzadeh

DISSERTATION PRESENTED
TO THE
INSTITUTE OF MATHEMATICS AND STATISTICS
OF THE
UNIVERSITY OF SÃO PAULO
TO
FULFILL THE REQUIREMENTS OF THE DEGREE
OF
MASTER IN SCIENCE

Program: Computer Science

Supervisor: Prof. Dr. Daniel Macêdo Batista

São Paulo, May 2020

Energy-Efficient Virtual Network Function Placement Based on Metaheuristic Approaches

This is the original version of dissertation text written by
the candidate Fatemeh Mosaiyebzadeh, as
submitted to the examining board.

Acknowledgments

First of all, I would like to appreciate my advisor, Professor Daniel Macêdo Batista, who has given me the opportunity of this MSc, and provided me with his guidance. I would also like to thank and give my heartfelt to my family and my friends for the support and encouragement.

Resumo

A preocupação em reduzir o consumo de energia no setor de tecnologias da informação e comunicação tem motivado cada vez mais a transição de serviços tradicionais dessa área para as nuvens. Nesse sentido, a virtualização de funções de rede (NFV – *Network Functions Virtualization*) surge como uma solução para migrar várias funções de rede, de dispositivos de hardware dedicados, para um ambiente virtual. Com essa virtualização, além da promessa de aumento da eficiência energética, espera-se reduzir o custo financeiro e aumentar a flexibilidade e a escalabilidade na rede. Nesta pesquisa, propomos o desenvolvimento de algoritmos baseados em diversas metaheurísticas para escalonar as funções de rede em máquinas físicas de *data centers* de nuvens observando não apenas as capacidades das máquinas mas também o consumo de energia de todos os dispositivos físicos envolvidos, incluindo *switches* e outros elementos de interconexão. Além disso, nesta dissertação, mostramos o consumo de energia dos data centers de três camadas e árvores de gordura, aplicando nosso algoritmo de posicionamento de três VNF. Comparando nossos algoritmos propostos com o algoritmo de melhor ajuste em um ambiente de simulação, o baseado no reconhecimento simulado economizou 55,44% do consumo de energia em um datacenter de três camadas e o baseado em um algoritmo memético economizou 49,18% do consumo de energia em um datacenter de árvore de gordura. Além disso, mostramos que, para o encadeamento de funções de serviço, o datacenter de três camadas é mais eficiente em termos de energia do que o datacenter de árvore de gordura. A segunda contribuição desta dissertação é desenvolver uma estrutura de código aberto com algoritmos metaheurísticos de colocação de VNF.

Palavras-chave: Virtualização de Funções de Rede, Encadeamento de Funções de Serviço, Computação em Nuvem, Eficiência Energética.

Abstract

Concerns about reducing energy consumption in the sector of information and communications technology have increasingly motivated the transition of traditional services of this area to the clouds. In this sense, Network Functions Virtualization (NFV) comes as a solution to migrate various network functions, from dedicated hardware devices, to a virtual environment. With this virtualization, besides the promise of increased energy efficiency, it is expected a reduction of the financial cost and an increase of the network flexibility and scalability. However, bad placement decisions of the Virtual Network Functions(VNFs) on the physical hosts can avoid the expected reductions in energy consumption. In this research, we proposed the development of three VNF placement algorithms based on metaheuristic approaches to place the network functions in physical machines of cloud data centers. The first aim of our approaches is declining the energy consumption of all the physical devices involved, including switches and other interconnection elements. Moreover, in this dissertation, we showed the energy consumption of Three-Tier and Fat-Tree data center by applying our three VNF placement algorithm.

On average, in simulation experiments, comparing our proposed algorithms with the best-fit algorithm in a simulation environment, the one based on simulated annealing saved 55.44% of energy consumption in a three-tier data center and the one based on a memetic algorithm saved 49.18% of energy consumption in a fat-tree data center. Furthermore, we showed that for service function chaining, the Three-Tier data center is more energy-efficient than the Fat-Tree data center. The second contribution of this dissertation is to develop an open-source framework.

Keywords: Network Functions Virtualization, Service Function Chaining, Cloud Computing, Energy Efficiency.

Contents

List of Abbreviations	xi
List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 Objective	3
1.2 Organization	4
2 Concepts	5
2.1 Cloud Computing	5
2.2 Data Center Network (DCN)	5
2.3 Virtualization	7
2.3.1 Virtual Machines	9
2.3.2 Containers	9
2.4 Network Function Virtualization	9
2.4.1 Virtual Network Functions	11
2.4.2 Energy Efficiency of NFV	11
2.5 Software-Defined Networking (SDN)	11
2.6 Service Function Chaining	12
2.7 Metaheuristics	13
2.7.1 Hill-Climbing Algorithm	13
2.7.2 Simulated Annealing Algorithm	14
2.7.3 Memetic Algorithm	16

3	Related Work	19
3.1	Simulation and Testbeds	19
3.1.1	Summary	23
3.2	Measurements	24
3.2.1	Summary	25
3.3	Placement Algorithms	25
3.3.1	Summary	27
4	Energy Efficient Algorithms for Virtual Network Function Placement	29
4.1	Problem Formulation	29
4.1.1	Power Model	30
4.2	Metaheuristic Algorithms	31
4.2.1	SHC-VNFP Algorithm	31
4.2.2	SA-VNF Algorithm	32
4.2.3	Mem-VNFP Algorithm	33
5	Performance Evaluation	37
5.1	Data Center Topology	40
5.1.1	First scenario	40
5.1.2	Second scenario	41
5.2	Application Scenario and Workload	42
5.3	Analysis of Execution Time	43
5.4	Analysis of Energy Consumption by Hosts	45
5.5	Analysis of Energy Consumption by Switches	48
5.6	Analysis of total energy consumption	50
5.7	Summary	52
6	Conclusions	53
7	Appendix	55

Bibliography	57
---------------------	-----------

List of Abbreviations

CAPEX	Capital Expenditures
CE	Control Element
CPE	Customer Premises Equipment
DCNA	Data Center Network Architecture
DHCP	Dynamic Host Configuration Protocol
DPI	Deep Packet Inspection
DRS	Dynamic Resource Scheduling
DVFS	Dynamic Voltage Frequency Scale
ETSI	European Telecommunications Standards Institute
FE	Forwarding Element
FIB	Forwarding Information Database
FW	Firewall
GHG	Green Houses Gases
ICT	Information and Communications Technology
IDS	Intrusion Detection Systems
IE	Interconnection element
IPS	Intrusion Prevention System
IPSec	Internet Protocol Security
LB	Load Balancer
MANO	Management and Orchestration
NAS	Network-Attached Storage
NAT	Network Address Translation
NF	Network Function

NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NIC	Network Interface Card
ONF	Open Networking Foundation
OPEX	Operating Expenses
PDP	Policy Decision Point
PM	Physical Machines
RG	Residential Gateway
SDN	Software-Defined Networking
SF	Service Function
SFC	Service Function Chaining
STB	Set-Top Box
VM	Virtual Machine
VNF	Virtual Network Function
VNFI	Virtualized Network Function Infrastructure
VoIP	Voice over IP
VPN	Virtual Private Network
WAN	Wide Area Network
WoC	WAN optimization Controll

List of Figures

1.1	Main components of electricity consumption for the ICT sector. From [CLT ⁺ 17]	1
1.2	Global mobile data traffic (ExaBytes per month). From [PJ17])	2
1.3	Service Chain of Network Functions (adapted from [BB15])	3
1.4	VNFs hosted in cloud environment	3
2.1	three-tier tree architecture	6
2.2	two-tier tree architecture	6
2.3	Fat-Tree architecture	6
2.4	DCell architecture	7
2.5	BCube architecture	7
2.6	Native hypervisor sits between hardware and multiple operating systems	8
2.7	Hosted hypervisor sits between original operating system and new operating system	8
2.8	The Network Function Virtualization architectural framework. From [LC15])	10
2.9	The relationship between NFV and SDN. From ([Fow])	10
2.10	Software-Defined Network Architecture [ONF12]	12
2.11	SFC architecture ([Bro15])	13
2.12	Roulette wheel selection	17
2.13	Single-point crossover	18
2.14	Two-point crossover	18
2.15	Uniform crossover	18
4.1	Representation and capacity of the chromosomes in the population.	34

5.1	CloudSimSDN Architecture	37
5.2	Class Diagram for simulating Service Function Chaining.	38
5.3	Simple Data Center configuration for the validation test.	39
5.4	Three-Tier data center topology with 100 hosts	40
5.5	Three-Tier data center topology with 50 hosts	40
5.6	Three-Tier data center topology with 40 hosts	41
5.7	Fat-Tree data center topology with 128 hosts	41
5.8	Fat-Tree data center topology with 63 hosts	41
5.9	Fat-Tree data center topology with 20 hosts	42
5.10	Average execution time by Three-Tier data center architecture.	44
5.11	Average execution time by Fat-Tree data center architecture.	45
5.12	Hosts' energy consumption by Three-Tier data center architecture.	46
5.13	Hosts' energy consumption by Fat-Tree data center architecture.	47
5.14	Average switches' energy consumption by Three-Tier data center	48
5.15	Average switches' energy consumption by Fat-Tree data center.	49
5.16	The total energy usage of Three-Tier data center.	50
5.17	The total energy usage of Fat-Tree data center.	51

List of Tables

3.1	Summary of the simulation related works (A “_” in the table is used when the information was not provided by the authors.)	23
3.2	Comparison of our work in relation to the related works	26
4.1	Notation	29
5.1	Web service model configuration	38
5.2	SFCs configuration	39
5.3	Second SFCs configuration	39
5.4	data centers configuration	42
5.5	VNF instances	42
5.6	Performance requirements for the SFCs	43
5.7	Average execution time(milliseconds)	45
5.8	Average hosts’ energy consumption by Three-Tier data center	46
5.9	Average hosts’ energy consumption by Fat-Tree data center	47
5.10	Average switches’ energy consumption by Three-Tier data center	49
5.11	Average switches’ energy consumption by Fat-Tree data center	50
5.12	Average total energy consumption by Three-Tier and Fat-Tree data center	51

Chapter 1

Introduction

It is estimated that facilities from Information and Communications Technology (ICT) sector, such as personal computers, servers, cooling equipment, fixed and mobile telephone instruments are responsible for consuming around 4.7% of electricity worldwide [C⁺13] [VHLL⁺14]. A problem related to this is the fact that electrical energy consumption has a direct influence on Green Houses Gases (GHG) emissions [BCR⁺12]. The ICT sector was estimated to consume over 12% of global electricity demand in 2017 and it is estimated to increase at least 7% annually through 2030, which is the double of the average rate of electricity growth globally. Figure 1.1 shows the percent of energy consumption by ICT sector equipment [AC13]. Despite recent efforts, the ICT sector's GHG emissions are likely to double by 2020.

Figure 1.1: *Main components of electricity consumption for the ICT sector. From [CLT⁺17]*

The UN's climate conference in Mexico in 2010 reported that limiting global warming to no more than 2°C above is a vital goal for countries. However, in May 2015, the UN published a new report stating that the 2°C limit was not adequate for avoiding severe impacts of climate change. Therefore, 195 countries endorsed the agreement to limit global warming to 1.5°C below instead of 2°C. Besides, the European Union has set up the objective to decline 20% of GHG emissions volume before 2020, and 40% before 2030 [GKZ16]. Nevertheless, it is necessary to take into consideration that Europe accounts for only 9.6% of the worldwide CO_2 emissions comparing to 14.0% of the US and 29.2% of China in 2016 [MGS⁺18]. Therefore, without new technologies in the ICT infrastructure, the total European telecom network energy consumption is expected to increase from 14.3 TWh per annum in 2005 to 35.8 TWh per annum in 2020 [CGW⁺12].

The number of mobile subscriptions is growing rapidly. According to Ericsson in 2017, the number of mobile subscribers will increase by 9.1 billion by 2023, a growth of more than 15% comparing to 2017. With this growth, more than 90% of mobile data traffic will come from smartphones by 2022. Thus, a sustained increase in traffic is predicted until 2023, to about 110 Exabytes per month and more than eight times the traffic of 2017 (Figure 1.2). On the other hand, the diversity of services provided by the mobile telecommunications industry cause a significant increase in the capacity of data traffic and directly affects the energy consumption and CO_2 emission [ASH19].

Figure 1.2: Global mobile data traffic (ExaBytes per month). From [PJ17])

The energy consumption by Internet can be divided in four major areas: data centers, communication networks, end-user devices, and the manufacture of the equipment for the other three areas. Indeed, Internet relies on millions of physical servers in data centers which are connected with miles of undersea cables, switches, and routers, all requiring a lot of energy to run. Therefore, the rapidly increasing of the total number of devices leads the ICT sector to consume more energy and generate more GHG footprint in manufacturing in China and other parts of Asia, which are dependent on coal fuel [CLT⁺17].

With the increasing of the rigid environmental standards and the raising of the energy consumption of data centers, energy efficiency has become a high-priority objective in communications and networks. Efforts to reduce unnecessary energy consumption were introduced both in academy and industry as “green networking technologies” [BCR⁺12]. One of these possible technologies is virtualization.

In the face of a large number of demands in various fields such as science, engineering, and business, the electrical energy consumption and carbon footprints are expected to increase also in data centers that host cloud applications. So, significant investments and research are needed in cloud computing to save energy and reduce GHG [BBA10]. The expansion of “softwarization” and virtualization for several fields also increases the use of the Internet. In point of fact, the Internet makes a connection between the final user and the solution in the cloud, moreover, it plays a significant role in enabling virtualization. [CSM15]. So, several researchers affirm that the future success of cloud computing depends on the network virtualization for many reasons, such as the ability for sharing a resource, to provide isolation among users, to build a large virtual resource that handles like a large resource, and to facilitate virtual devices management [JP13].

Network Functions Virtualization (NFV) is a network paradigm to virtualize functions such as load balancers, firewalls, Intrusion Detection Systems (IDSs), and switches [EAL17]. The modern cloud providers have deployed different technologies such as NFV to facilitate the rapidly increasing service demands from cloud tenants. The network functions become virtualized applications in these modern clouds and are called Virtual Network Functions (VNFs). The group of VNFs, that is called a service chain, should be attached for service delivery, in order to provide the required service to traffic flows [YXKZ15]. Figure 1.3 shows an example of a specific network service chain whose packets shall pass through: 1) a firewall, 2) a deep packet inspector, 3) an encryption server, 4) a network monitor, and 5) a decryption server.

NFV replaces dedicated hardware middleboxes by VNFs running on commodity servers [XLWX16]. Traditionally, network functions are implemented on physical middleboxes, and each network function is executed on special hardware equipment that is expensive and difficult to upgrade. Compared

Figure 1.3: *Service Chain of Network Functions (adapted from [BB15])*

to traditional network architectures, NFV gives immense benefits in reducing equipment costs, increasing flexibility of the network function deployment, and increasing the energy efficiency [HB16]. The main objective of NFV is to transform network architectures, by implementing many Network Functions (NFs) in software instead of on proprietary hardware or dedicated appliances. Additionally, it aims to transform traditional network operations, as software can easily be moved to various locations without using new equipment [SDC⁺14].

Moving from a hardware to a virtualized software implementation can bring some benefits, however, it leads to scheduling issues in the resource sharing. Virtual Machines located in data centers, as shown in Figure 1.4, need to be carefully placed to implement the VNFs [STV15].

Figure 1.4: *VNFs hosted in cloud environment*

The locations to allocate the VNFs must be selected to guarantee that traffic can be routed through them in a suitable sequence with minimum changes in the forwarding tables. Software-Defined Networking (SDN) is a technology that can deal with the routing requirements [KRV⁺15]. Therefore, SDN control plane can be used to programmatically configure forwarding rules in the switches/routers to enable VNF orchestration [BCAB15].

1.1 Objective

There is a growing consensus on the need to develop more energy-efficient telecommunication systems. However, reducing energy consumption by the ICT sector is a challenging issue, and “green networking technologies” are envisioned to achieve not only efficient energy consumption but also an effective way to decline GHG emissions. As we know, virtualization is an important key in the “green networking technologies”.

Nowadays, virtualization makes data centers more flexible. Also, NFV seems like a solution, but a lightweight and scalable framework for experiments is one of the main purposes to evaluate new energy-saving mechanisms. Hence, The specific objectives of this dissertation are:

- To study the energy savings impact of VNFs.
- Studying simulators and testbeds to find an open source, a lightweight, scalable framework in order to implement VNFs placement algorithms and SFC. Additionally, this framework has to support data center and network switches simulation, scheduling algorithms implementation, and provide energy consumption simulation.
- Survey virtual machines and containers to find appropriate options to implement VNFs.
- Analyzing the scheduling policies such as VNFs placement algorithms, which are substantial to decrease energy consumption by the host and network elements.
- Proposing a set of algorithms to take into account simultaneously all VNFs, SFC, host power consumption, switch power consumption, and link utilization.
- To propose a metaheuristic based on a Standard Hill- Climbing algorithm.
- To propose a metaheuristic based on a Simulated Annealing algorithm.
- To propose a metaheuristic based on a Memetic algorithm.
- Extend the CloudSimSDN-NFV and build, to make it publicly available, a Metaheuristic cloud simulation framework for VNFs placement in data centers.
- Modeling two sorts of data center topologies such as Three-Tier and Fat-Tree topology with different numbers of hosts and switches, to prove that our proposed allocation mechanisms can work with different topologies.

1.2 Organization

The remainder of this dissertation is organized as follows:

- Chapter 2 explains basic concepts.
- Chapter 3 presents the related work in terms of data center and network simulators and energy-efficient VNFs placement approaches.
- Chapter ?? summarizes the research proposal.
- Chapter 4 presents the proposed energy-efficient algorithms to place VNFs in the data centers.
- Chapter 5 presents the performance evaluation of the proposed algorithms.
- Chapter 6 concludes the dissertation with a discussion on future directions.

Chapter 2

Concepts

This chapter details the basic concepts needed to better understand the research presented in this dissertation. On Section 2.1 the cloud computing concept will be introduced, Data Center architecture is introduced on Section 2.2 and the virtualization concept on Section 2.3. NFV is explained on Section 2.4 and the explanation of the Software-Defined Networking (SDN) is on Section 2.5. Service Function and Service Function Chaining are explained on Section 2.6. On Section 2.7 the metaheuristics used in this work are detailed.

2.1 Cloud Computing

Cloud computing is a technology for hosting and delivering applications as service over the Internet by using hardware in data centers. The data center's hardware and software is what we will call a cloud [ZCB10]. According to the NIST definition of cloud computing, the offered services by clouds can be grouped into three categories: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) [MG⁺11].

- Infrastructure as a Service (IaaS) provides processing, storage, networks, and other fundamental computing resources to the consumer. Examples of IaaS providers include Amazon EC2 (Elastic Cloud Computing) Service, and S3 (Simple Storage Service) where compute and storage infrastructures are open to public access with a utility pricing model; Eucalyptus is an open-source cloud implementation that provides a compatible interface to Amazon's EC2, and allows people to set up a Cloud infrastructure at premise and experiment before buying commercial services.
- Platform as a Service (PaaS) provides an integrated environment on the cloud infrastructure to build, test, and deploy custom applications created using programming languages, libraries, services, and tools supported by the provider. An example of PaaS providers includes Google's App Engine, which enables users to build Web applications on scalable systems.
- Software as a Service (SaaS) provides a special-purpose application running on a cloud infrastructure to the consumer. The software is remotely accessible by consumers through the Internet. A typical example is Salesforce, which provides online CRM (Customer Relationship Management) Services. Other examples are Google Drive and Dropbox which allow files and folders to be shared and synchronized across multiple devices.

2.2 Data Center Network (DCN)

Data centers contain hundreds of thousands of servers (physical machines), storage, and network devices. Servers in the data center, are interconnected via switches, routers, and high-speed links. A data center network is the communication infrastructure described by the network topology, routing/switching equipment, and the used protocol.

Data centers can be classified into switch-centric or server-centric categories. In the switch-centric design, switches are the major components for interconnection and routing. In the server-centric design, servers with multiple Network Interface Cards (NIC) exist to facilitate the routing and packet forwarding decisions. The conventional data center has a switch-centric design. Typical data center network architecture is made up of a three-tier tree topology (Figure 2.1) that can be designed as a graph with each node representing core, aggregation, top-of-rack switches (ToR), and servers, and with each edge representing physical links between servers and switch nodes. Each server can be used to host one or multiple VMs. Moreover, usually each server has the same configuration in terms of CPUs, RAM, and storage. Datacenter networks with a massive number of hosts and networking equipment face with performance and scalability issues that justify the deployment of topologies different from the three-tier. Some examples are the two-tier tree topology (Figure 2.2) and the Fat-Tree topology (Figure 2.3). In Fat-Tree, there are multiple links between switches that lead to a redundant and high-available network.

Figure 2.1: *three-tier tree architecture*
[HAK⁺15]

Figure 2.2: *two-tier tree architecture*
[HAK⁺15]

Figure 2.3: *Fat-Tree architecture*
[HAK⁺15]

Other two topologies are Dcell (Figure 2.4) and BCube (Figure 2.5). BCube topology was proposed for container-based data centers [HAK⁺15].

Figure 2.4: *DCell architecture*
[HAK⁺15]

Figure 2.5: *BCube architecture*
[HAK⁺15]

2.3 Virtualization

Virtualization is a term that refers to the abstraction of computer resources. The virtualization technology was introduced by IBM engineers which had worked with MIT programmers to develop a time-sharing system. The time-sharing system permitted project teams to use part of the mainframe computers [Cre81]. Therefore, the mainframes were designed to allow users to access resources simultaneously while each user has separate resources. Virtualization became an important technology to enhance system security, reduce costs, provide greater flexibility, and improve efficiency in the data center.

There are many levels of virtualization such as hardware, hosted, operating system and network.

- **Hardware level virtualization**, also known as server virtualization or simply virtualization, is the abstraction of computing resources between the operating system and the hardware. The hardware virtualization permits running multiple operating systems on the same physical hardware. Intel Virtualization Technology (VT-x) and AMD's AMD-V are examples of virtualization technologies and a new CPU execution mode that allows the VMM(hypervisor) to run in a new root mode. The hypervisor is a piece of software that facilitates creation and management of VMs. Native or Bare-Metal, as shown in Figure 2.6, is a type of hypervisor that runs directly on the host's hardware. Guest OS can be installed on top of this hypervisor. Citrix XenServer, VMware ESX, and Hyper-V are examples of the Native or Bare-Metal hypervisor.

Figure 2.6: *Native hypervisor sits between hardware and multiple operating systems*

- **Hosted level virtualization**, as shown in Figure 2.7, places a hosted hypervisor on top of the existing operating system of the computer. As a matter of fact, the hosted hypervisor sits between an existing operating system and new operating systems, therefore users use hosted on their Computer or Desktop. In Hosted virtualization, the users use their own machine's Hardware and operating system to run a virtualization software such as VMWare Server, VMWare Workstation, Oracle Virtual Box or Microsoft Virtual PC. Inside this hosted platform users run their virtual machines.

Figure 2.7: *Hosted hypervisor sits between original operating system and new operating system*

- **Operating system level virtualization** refers to an abstraction between the host operating system and applications running on a machine. In this level of virtualization, the kernel allows the existence of multiple isolated user-space instances. It enables multiple isolated execution environments within a single operating system kernel. FreeBSD Jail, Solaris Zones/Containers, Linux-VServer, OpenVZ and Virtuozzo are examples of operating system level virtualization [Kol06].
- **Network level virtualization** Network virtualization slices a physical network into multiple virtual sub-networks, therefore, multiple sub-networks can be created on the same physical network. The network virtualization can restrict data transfer across networks and improve security and data usage monitoring. Moreover, one network does not affect other networks [Kol06].

2.3.1 Virtual Machines

A Virtual Machine (VM) is an emulation of a physical computer that is sharing the underlying computing system. Virtual Machines create a new abstraction layer between the real physical computer hardware and the logical hardware in order to disconnect the physical aspects from the functional aspects of the computing unit. This characteristic of the VMs allows to create a virtualized computing environment on a physical environment [CN01]. Thus, a VM is an abstraction of a virtualized BIOS, network adapters, storage, memory, and CPU which requires a full guest OS image, additional binaries and libraries for hosting applications and services [VN15]. The main advantage of the Virtual Machines is the isolation. Running an application on a VM is completely isolated from other VMs. Moreover, the isolation property of VMs provide users an independent system and it can conceals the resources usage between the VMs. Furthermore, the VMs can be easily maintained, recovered, and provisioned [PWBN03].

2.3.2 Containers

Containers are lightweight virtualization technologies for Virtualization. These lightweight technologies could typically avoid the hardware requirements and overheads associated with VMs [CP17]. In contrast to Virtual Machines, containers create the abstraction at the OS level in order to support programs and system resources to run an application. Containers create multiple isolated space instances of a much smaller size compared to VMs. The main benefit of container is fast migration and easy instantiation because of the container lightweight nature, but this does not make VMs obsolete. VMs still provide higher security guarantees to the applications running on them compared to containers [Tow15].

Containers implement isolation of processes at the operating system level. They run on top of the same shared operating system kernel of the underlying host machine, and one or more processes can be run within each container. This approach (shared kernel) has a few disadvantages. Firstly, containers of a operating system can run only on top of the same operating system. Secondly, containers have a weak resources isolation because the host kernel is exposed to the containers and this weakness has a direct influence on the security of the virtualization [MKK15].

2.4 Network Function Virtualization

After introducing NFV to a group of network service providers at the SDN and OpenFlow World Congress in October 2012, ETSI (European Telecommunications Standards Institute) [CDT⁺] became the home of the Industry Specification Group (ISG) for NFV and defined the NFV architectural framework (shown in Figure 2.8). Network Function Virtualization (NFV) is an approach to implement network functions in an open and standardized IT virtualization environment. In fact, NFV was introduced to decouple network functions such as routers, firewalls, load balancers, NAT, DNS, and other dedicated network servers from their dedicated hardware and implement them as software components on virtualized network infrastructures. Therefore, NFV can greatly reduce CAPEX and OPEX costs for service providers and network operators by software-based development, instead of using proprietary hardware for switches and servers.

Figure 2.8: *The Network Function Virtualization architectural framework. From [LC15])*

NFV is a network architecture created to enable VNFs to be deployed and executed on a Network Functions Virtualization Infrastructure (NFVI). NFV emerges to manage traditional middleboxes as single modules of software. It can facilitate managing each VNF independently because it permits the isolation of each function [HMR⁺15]. As shown in Figure 2.8, the main components of the Network Functions Virtualization architectural framework are the NFVI (Network Functions Virtualization Infrastructure), the VNF (Virtualized Network Functions), and the NFV M&O (Network Functions Virtualization Management and Orchestration). The NFVI abstracts the underlying computing, storage, and network resources through virtualization and provides them as virtual resources to the VNF. NFV M&O focus on all virtualization-specific management tasks necessary in the NFV framework [LC15].

NFV is highly complementary to SDN. It can be implemented without a SDN, although the combination of both SDN and NFV provides greater value. Figure 2.9 demonstrates a relationship between NFV and SDN.

Figure 2.9: *The relationship between NFV and SDN. From ([Fow])*

NFV was proposed to take advantage of computing resources, such as VMs, to provide specific network functions that need variable amount of computing power. Instead of purchasing expensive dedicated hardware for CPU-intensive network functions, NFV can provide a cheaper alternative that utilizes a general hardware with virtualization technology. Additionally, NFV alters the paradigm of network function deployment via advanced virtualization technologies [CCC⁺16]. In the concept of NFV, network functions are provisioned in virtualized resources instead of being mixed with the dedicated hardware, thus, it enables provision and migration of network functions through the infrastructure elastically and dynamically [Bri14].

In the scenario of home networks, NFV can provide home services through dedicated customer premises equipment (CPE) supported by network-located back-end systems. Each CPE devices include residential gateways (RGs) to access the internet and set-top boxes (STBs) for multimedia services [AGJ⁺13].

Several use cases for NFV based on single virtual functions have been implemented, such as routing, broadband remote access server, deep packet inspection, and CPE. All these originate from the research community. However, the biggest implementations have occurred from equipment vendors. HP OpenNFV, Huawei NFV Open Lab, Intel Open Network Platform (Intel ONP), CloudNFV, Alcatel-Lucent CloudBand are some key NFV implementation and products from the industry.

NFV also plays a key role in enhancing 5G radio access networks' functions, including increased automation, operational agility, and reduced capital expenditure. By increasing mobile users and traffic data, NFV becomes interesting to improve cost efficiency, flexibility, and performance guarantees of mobile networks in general. Companies and vendors began to implement network functions in software components and are obtaining several gains, including reduction of the carbon footprint and energy consumption [AHGZ16].

2.4.1 Virtual Network Functions

VNFs are the software-based network elements that are executed on the NFV Infrastructure [LFLK17]. VNFs are generally run on commodity hardware, which can reduce the capital and operational expenditures of buying and maintaining specialized hardware. An NF is a functional block within a network infrastructure such as DHCP servers or firewalls in conventional networks. Thus, a VNF is an implementation of an NF on virtual resources such as a VM. VNF can consist of multiple components, and be deployed over multiple VMs [ETS13]. Firewall, load balancer, DPI, VPN are just some of the examples of network functions that can be virtualized [AAL17].

2.4.2 Energy Efficiency of NFV

NFV brings benefits in terms of energy consumption [BBDC10]. For instance, NFV can reduce the total energy consumption in the data center by concentrating the VNFs on fewer servers when the cores are lightly loaded during off-peak hours [MSG⁺16]. Recently, studies to reduce energy consumption in the data center which run NFV application have been proposed because NFV can consume a considerable amount of energy in large-scale networks [MSG⁺16]. There exist some approaches that aim at improving resources management in order to gain better energy efficiency [KAEC⁺18]. Server consolidation and server throttling are two approaches that enhance energy efficiency.

Server consolidation is a widely used technique that allows several VMs to pack on a single physical machine in order to reduce energy consumption by the physical machine. Therefore, server consolidation can decline the utilized physical machines in a data center. The migration of VMs and resizing the set of servers in the data center are two ways that could be used to prepare server consolidation [VG15].

Server throttling reduces the power consumption of an individual server when the server is at a low utilization level. There exist some techniques for server throttling, which indirectly affect performance and power consumption. Speed scaling, CPU pinning, and Dynamic Power Management are techniques that are used to throttle power consumption [Krz19].

2.5 Software-Defined Networking (SDN)

The software-defined networking paradigm is a fully programmable software platform that facilitates the control and management of a computer network environment. The main goal of the SDN is to centralize all the intelligence of the network in a single controller, instead of having it divided between all the switches and/or routers. SDN decouples the control plane (software) from the network data-forwarding plane (hardware). The control plane is a logically centralized controller

(SDN controller) known as the brain of the SDN. The control plane receives instructions or requirements from the data plane and provides a global view of the network to the operator. Management tasks are implemented as applications running on top of the SDN controller. These applications make packet processing decisions based on the global view and distribute the decisions to the data plane via the SDN controller. Thus, SDN makes the network programmable and configurable dynamically through the centralized controller. Furthermore, this type of fully-programmable network control logic can be provisioned and exposed through APIs as a set of on-demand, customizable, and innovative network services to other internal and external applications like business software, orchestration software, and policy engines. Figure 2.10 demonstrates a logical view of the basic SDN architecture, which is proposed by the Open Networking Foundation (ONF) [ONF12]

Figure 2.10: *Software-Defined Network Architecture* [ONF12]

SDN architectures support a set of APIs in order to implement network services. As shown in Figure 2.10, the network control layer is decoupled from the infrastructure layer (network devices) through the process of abstraction and represented in a logically centralized and fully-programmable software platform called SDN control software. The logically centralized software communicates with network devices through the control-data plane interface called SDN southbound that enables dynamic modifications of switch flow entries to manage the network efficiently. The application layer communicates with the control layer APIs called northbound APIs of the control layer. The primary goal of these northbound APIs is to provide a way to model the networking layer as a single "big Switch" for the application layer [KD13].

OpenFlow protocol, as the control-data plane interface, is the most famous SDN protocol for southbound APIs [MAB⁺08]. OpenFlow provides an open protocol that is used by the controller to install flow entries internally in a switch. OpenFlow has been supported by many physical switch vendors such as HP, Cisco, and Juniper.

2.6 Service Function Chaining

Service Function (SF) means a function to perform specific processing on network packets. SFs can be implemented in virtual or physical network equipment. More than one SF can be implemented in the same network equipment, and more than one instance can exist for a single SF [BB15]. The process of creating, altering and eliminating network SFs is traditionally very costly, and need the placement of specialized hardware devices and their interconnection. With a combination of SDN and NFV, the possibility of SFs management with a positive impact in terms of agility and cost has increased. Nowadays, deploying SFs is an essential action for a network provider.

SFs can be classified as active or passive. Active SFs are part of the main course of a packet. In this category, there are two subcategories. Firstly, a function may drop packets or forward them,

such as a firewall. Secondly, functions can change the content of packets, such as an IPsec VPN server. Passive SFs are out of the main course of the packets. These functions mainly inspect packets (e.g., a monitoring system or a deep packet inspection mechanism) [SGP⁺15].

A sequence of SFs is called a Service Function Chaining (SFC). SFC allows various service functions to connect to others in order to form a more complete service [Bro15]. Figure 2.11 presents the architecture of the SFC. According to this architecture, at the start point, a classifier recognizes and classifies traffic before forwarding it into the processing path. The control plane manages the path. Also, the SFC control plane consists of a topology server and a policy decision point (PDP). The topology server communicates to the input and output nodes (the classifiers) and locates service function nodes in the network [Bro15].

Figure 2.11: *SFC architecture ([Bro15])*

2.7 Metaheuristics

The term metaheuristic was first introduced by Glover in 1986 [Glo86]. A metaheuristic is an efficient approach for many hard optimizations problems that cannot be solved optimally. Metaheuristics algorithms consist of an iterative process which seeks for feasible solutions. They are a proper way to problems for which there is no satisfactory problem-specific algorithm or when it is not practical to implement such a method. They are not problem-specific, but they may make use of domain-specific knowledge in the form of heuristics that are controlled by a higher-level strategy [BLS13].

Some well-known algorithms are considered metaheuristics: Hill-Climbing [Tov85], Simulated Annealing [KGV83], and Memetic Algorithm [MC03]. In the following, these metaheuristics, which are used in this research, are explained.

2.7.1 Hill-Climbing Algorithm

Hill-Climbing (Algorithm 1) is one of the simplest metaheuristic algorithms for local search. [Nil96]. Initially, the algorithm begins with a randomly generated initial solution S_0 , and the new solution is generated using the current solution. In fact, the Hill-Climbing algorithm explores a neighborhood of the solution and picks the best-found solution. The new solution replaces the current solution if it has a better value, then the algorithm uses the new solution as the next point to explore. Hill-Climbing algorithm performs a loop, and repeats it until the best solution is found.

Simplicity is the advantage of the Hill-Climbing method, but one of its major problems is that the algorithm cannot escape the local optimum, which leads to premature convergence. It easily gets stuck in a local optimum that can be far away from the global optimum. There are several techniques to overcome the premature convergence issues, as to randomly restart the search process several times.

Algorithm 1 Hill-Climbing Algorithm

```

1: create an initial solution bestSolution (usually random)
2: while termination condition not met do
3:   currentSolution  $\leftarrow$  getFeasibleNeighbour(bestSolution)
4:   if getValue(currentSolution)  $\leq$  getValue(bestSolution) then
5:     bestSolution  $\leftarrow$  currentSolution
6:   end if
7: end while

```

2.7.2 Simulated Annealing Algorithm

Simulated Annealing (SA) is a learning model that is known as a threshold algorithm and is based on acceptance strategies. Simulated Annealing actually has its origins in metallurgy by simulating the physical quenching process in thermodynamics. Its starting point is based on the similarity between the annealing process of solid materials. The idea of Simulated Annealing comes from a paper published by Metropolis et al. in 1953 [MRR⁺53]. In metallurgy, annealing refers to the process of heating metal to a high temperature and then slowly cooling to alter its physical properties and change its internal structure according to a predefined strategy [VLA87]. In 1982, S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi [KGV83] took the idea of the Metropolis process and applied it to optimization problems. The idea was to use Simulated Annealing to search for feasible solutions and converge to an optimal solution.

Cooling Schedule

The way how to control temperature is called a cooling schedule or strategy. The temperature is reduced during the optimization and it has a key role in the performance of the algorithm. Moreover, the temperature will be set to the initial temperature that is greater than zero, and it will reach zero at the end of optimization. Defining a cooling schedule is a hard part of the simulated annealing algorithm, and it is necessary to define the initial temperature, final temperature, and estimate of total iterations. There are a wide range of temperature scheduling methods because there is no universal cooling schedule that performs well on all problems. Some of the cooling schedules are reducing the temperature linearly, exponentially, or in a logarithmical manner, for instance. Examples of some cooling schedules are shown in the following [NA98].

Equation 2.1 shows the exponential schedule of temperature:

$$T(t) = T_0 \alpha^t \quad (2.1)$$

Where:

$T(t)$ = Actual temperature value.

T_0 = Initial temperature.

α = Constant factor to decrease the temperature repeatedly ($0 < \alpha < 1$).

t = 'Time', is the step count

Equation 2.2 shows the linear reduction of the temperature:

$$T(t) = T_0 - \eta t \quad (2.2)$$

Where:

$T(t)$ = Actual temperature value.

T_0 = Initial temperature.

η = Constant factor to decrease the temperature repeatedly ($0 < \eta < 1$).

t = 'Time', is the step count

The linear reduction of the temperature have been used widely since it was introduced by Kirkpatrick et al. [KGV83].

Equation 2.3 shows the logarithmic cooling schedule, which was introduced by Geman et al. [GG84]

$$T(t) \geq \frac{c}{\log(1+t)} \quad (2.3)$$

Where:

- $T(t)$ = Actual temperature value.
- c = largest energy barrier in the problem
- t = ‘Time’ is the step count

The cooling schedule is a critical attribute of the simulated annealing. It can lead to a loss of global solution by restricting the search to the region of the model space around the starting point. In some cases, the cooling schedule leads the algorithm to perform like a random algorithm over the search space, which will increase the computational time. Therefore, the cooling schedule has to be defined in such a way that it must be accepted during the first iteration of the cooling loop. In practice, there is not an easy way to determine the appropriate cooling schedule. Some times, prior information about the problem can help to determine the proper cooling schedule.

Acceptance Probability

As we have seen, the Hill-Climbing suffers from problems in getting stuck at the local minima (or maxima). Simulated annealing solves this problem by giving a chance for worse moves (lesser quality) to be taken some times. That is, it allows some uphill steps so that it can escape from the local minima. Therefore. Simulated Annealing algorithm does not have a tendency to end up in the local minima or the local maxima because the new solution acceptance or rejection is characterized by a probability function. Unlike Hill-Climbing, simulated annealing chooses a random movement from the neighborhood. To find a good solution, we move from a feasible solution to one of its neighbors by a probabilistic criterion. The move will be accepted if the value is better than its current position. Otherwise, if the move is worse, then it will be accepted based on some probability that is discussed below.

According to the law of thermodynamics, when the temperature is T , the probability of the temperature drops with the energy difference of dE is $p(dE)$, which is expressed as [JHL⁺18]:

$$P(dE) = \exp\left(\frac{dE}{kT}\right), \quad (2.4)$$

Where:

- $P(dE)$ = Probability of acceptance of new solution.
- dE = Difference of the cost function between the current and the new state.
- k = A constant known as Boltzmann’s constant.
- T = Current temperature

Once the new state solution is created, the quality of the solution (cost function) will be compared with the quality of the current state solution. If the quality of the solution is better than the quality of the current state solution, $dE \geq 0$, then the neighboring state solution is accepted. Otherwise, If the neighbor state solution has a lesser quality, $dE < 0$, then the acceptance probability depends on dE and T . Temperature is a control parameter in the SA algorithm, and acceptance probability is directly proportional to temperature and reduces gradually with the temperature. At high temperatures, the acceptance probability would be high. Thus, the algorithm has enough chance to accept a solution with lower quality and escape from the local minima or maxima and reach the global optimum.

2.7.3 Memetic Algorithm

Hybrid metaheuristic algorithms improve search performance by combining different individual algorithms. Moreover, combining different algorithms can improve the explorative and exploitative power of the resulting algorithm. Memetic Algorithm (MA) was introduced by Moscato et al. [M⁺89] in 1989. It was inspired by Darwinian principles of natural evolution and Dawkin's notion of a meme, defined as a unit of cultural evolution that is capable of local refinements. In Dawkins' words:

Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperm or eggs, so memes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation.

MA is a type of hybrid metaheuristic algorithm which mixes the population-based algorithm with local search algorithms. Memetic algorithms are also known as genetic local search algorithms or hybrid evolutionary algorithms. Moscato named his algorithm as the memetic algorithms because the individuals were subject to local search operators, in addition to the traditional selection, combination, and mutation operators. In fact, a simple Memetic algorithm is an extension of the Genetic algorithm operators by one or more Local search algorithms after crossover and mutation during the evolutionary process. According to the Moscato, an individual cannot modify his genes, but he can adapt his memes before transmitting them. That is exactly what local search operators do, therefore, local search operators allow individuals to modify their inheritance before transmitting them to the next generations.

Algorithm 2 shows the MA pseudocode. A memetic algorithm begins with an initialization process, in contrary to other evolutionary algorithms, which generate a random solution, MA used more sophisticated mechanisms [SR96]. After that, an iterative process is carried out until a termination criterion is satisfied. Then, the evaluation operator evaluates the quality of individuals in the population, using the fitness value of each individual, and determines the sample of individuals that will reproduce. In the reproduction stage, new individuals are created using crossover operators. At the end of the cycle, the Hill-Climbing operator is applied for the local search, and the population will be renewed by new individuals.

Algorithm 2 Memetic Algorithm

```

1:  $P(n = 0) \leftarrow \text{Initialize population}$ 
2:  $P \leftarrow \text{Evaluate}(P)$ 
3: while termination condition not met do
4:    $P2 \leftarrow \text{Selection}(P)$ 
5:    $P2 \leftarrow \text{Crossover}(P)$ 
6:    $P \leftarrow \text{LocalSearch}(P)$ 
7:    $P \leftarrow \text{Update}(P, p2)$ 
8: end while

```

Population and Chromosomes Size

Population-based algorithms operate on a number of potential solutions which are called a population. Typically, a population is composed of between 30 and 100 individuals, although, the micro genetic algorithm uses very small populations that have around 10 individuals, with a restrictive reproduction and replacement strategy in an attempt to reach real-time execution [CFPF]. In genetic design, the chromosome length is fixed and it cannot change in generations. Evolution in this sense is understood as the process of approaching an optimal instantiation of alleles given a constant phenotype-genotype mapping and a fixed chromosome length. Chromosome length can be short (6 bits) or long (10 bits). If the problem is modeled with short chromosomes, one may not obtain good solutions due to the lack of design freedom. On the other hand, the chromosome with a long length for a particular problem will cause a high computational burden [KDW05].

Selection

In the selection method, a proportion of the existing population is selected to breed a new generation. Individuals are selected based on their fitness value process, the fitter individuals are typically more likely to be selected. Some popular and well-studied selection methods include:

- **Roulette wheel selection:** In roulette wheel selection, a probability of selection is computed for each individual. This probability is given by:

$$P_i = \frac{f_i}{\sum_{i=1}^n f_i} \quad (2.5)$$

Where:

f_i is the fitness of individual i and n is the number of individuals in the population. Candidate solutions with higher fitness value will be more likely to be selected. However, with roulette wheel selection, the candidate solution with lower fitness value has a chance to survive the selection process. Figure 2.12 shows a roulette wheel in which each candidate solution represents a region on the wheel. The size of the regions is proportional to the probability of selection of the candidate solution.

Figure 2.12: *Roulette wheel selection*

- **Tournament selection:** A tournament selection is a parent selection method which randomly selects two or more chromosomes from the population with equal probability and conducts “tournaments” between the selected chromosomes. A chromosome with a higher fitness value has a higher chance to win each tournament, and the winner of all tournaments is selected for reproduction.

Crossover

Crossover is a genetic operator which blends two chromosomes (parents) to produce and reproduce a new chromosome (offspring). The new chromosome generated by the crossover has to be better than both parents because it takes the best characteristics from the parents. Many crossover techniques exist. For instance:

- **Single-point crossover:** In this technique, the chromosomes of the parents are cut at some randomly chosen point and the tails of its two parents are swapped to get new off-springs. Figure 2.13 illustrates the Single-point crossover [PL97].

Figure 2.13: *Single-point crossover*

- **Two-point crossover:** In this crossing method, two crossing points are selected randomly from the parent chromosomes. The bits in between the two points are swapped between the parent organisms. This strategy can be generalized to k-point crossover for any positive integer k (See Figure 2.14).

Figure 2.14: *Two-point crossover*

- **Uniform crossover:** The Figure 2.15 illustrates the uniform crossover. Unlike the single-point crossover and two-point crossover, which generate two offsprings, the uniform crossover produces only one offspring. Each gene of the offspring is picked from the genes of the parents.

Figure 2.15: *Uniform crossover*

Renew Population

The update method replaces the individual with lower fitness value by the better individual in the new population. Therefore, the update component maintains the population at a constant size by substituting some pre-existing individuals in the original population by some of the obtained from the new population.

Chapter 3

Related Work

As explained in previous chapters, NFV can bring energy savings, and one way to deploy NFVs is by using VMs allocation on cloud data centers. Since, it is essential to realize experiments to evaluate the quality of an NFV placement proposal, like the ones proposed in this dissertation, this chapter provides a review of the literature in terms of methods to run experiments of NFV in cloud data centers. Works related to simulator are presented in Section 3.1, and Section 3.2 presented works relevant to measurements. Lastly, in Section 3.3, we also summarize VNFs placement algorithms proposed in the literature and compares them with the ones proposed in this dissertation.

3.1 Simulation and Testbeds

CloudsimSDN-NFV

CloudsimSDN-NFV [SB] is a new simulation framework that is an extension of CloudsimSDN. Conventional network simulators have some difficulties to evaluate a new method on a large-scale, but CloudsimSDN-NFV can overcome such a hardship. In fact, CloudsimSDN-NFV is an easy-to-use and scalable toolkit to research. This new framework provides new features to support NFV functionalities, such as VNF allocation, migration, and auto-scaling. Besides, the concept of edge computing is integrated and supported by the CloudsimSDN-NFV, including multiple data centers, inter-cloud networks, and differentiated data center capacities.

In the simulator, the edge data centers are microscale data centers that are placed near the end-users. Therefore, end-users can access application services from edge data centers or cloud data centers through inter-cloud networks. Also, VNFs can be located in the edge or cloud data centers, and cloud data centers can provide VMs for application services and VNFs for network services.

The simulator also can simulate SFC in the framework and the SFC is defined as a chain of multiple service functions (i.e., VNFs) as an ordered and directed list. Enforcing SFC is determined in ServiceFunctionForwarder which forwards the matched packet to the chain of SFs. In fact, the forwarder checks every ServiceFunctionChainPolicy defined in the current simulation configuration, which is composed of the source and destination VM's. On top of the SFC policies, CloudsimSDN-NFV also implement the auto-scaling policy for SFCs. The auto-scaling policy can increase the capacity of the SF and/or the number of SF machines for the specific function, also can increase the allocated bandwidth for the specific SFC if the bottleneck is at the network. ServiceFunctionAutoScaler retrieves the average end-to-end latency for packets in the SFC and the utilization of SFs in the SFC.

CloudSimNFV

Yang et al. [YXLT15] developed the CloudSimNFV framework to evaluate resource management policies for NFV scenarios over a cloud data center environment. It is an extension of the CloudSim [BBA10] platform, a cloud simulator, with specific NFV scenarios. CloudSimNFV already provides some energy consumption models and extends scheduling algorithms under NFV

scenarios. In CloudSimNFV, users can specify both physical and virtual topology configurations. The structure of simulations is composed of CloudSim core logic, VM scheduler modules, and Energy-aware modules.

Therefore, the CloudSim core logic simulates the basic compute elements of the cloud infrastructure. Also, physical hosts can be defined with specific configurations, and VMs are placed on the host that meets resource requirements such as CPU, memory, network bandwidth, and storage size. According to VM scheduling policies, VMs may migrate from one physical host to another during the simulation. Moreover, CloudSimNFV uses the concept of NFVlet, which is a portion of the cloud used to simulate the NFV tasks under NFV scenarios.

This simulator has a specific class to evaluate the energy consumption of simulated NFV environments. The class considers CPU, memory, disk, and network utilization. The framework already provides three VM allocation policies: DRS, EcoCloud, and NFV. The DRS (Dynamic Resource Scheduling) algorithm allocates VMs to physical machines (PM) with the lowest load and predefines a threshold that controls system imbalance level. During the migration stage, in the case of increasing the predefined threshold, VMs can be reallocated to other PMs in order to decrease the threshold. The EcoCloud algorithm and NFV algorithm adopt a probabilistic function to select the most suitable PM to allocate VM.

In order to validate the accuracy of CloudSimNFV, the authors compared the energy consumption in CloudSimNFV and VMware virtualization platforms with the same workload. Also, to model the scenario in VMware, they set up two physical nodes with 12 cores and 48GB memory, one physical node with 8 cores and 4GB memory. The core capacity is equivalent to the Intel Xeon CPU E5-2620 capacity. Each node can support several virtual machines, and virtual machines are configured as one type of 4 cores with 4GB or 2cores with 2GB. The validation results showed differences between VMware, and CloudSimNFV always less than 5%. This slight loss of accuracy was due to network bandwidth usage. By comparing the three different VM allocation policies already shipped with CloudSimNFV, the authors noticed that the NFV algorithm is better in terms of energy consumption.

CloudSimSDN

Son et al. [SDC⁺15] presented a framework named CloudSimSDN, which can evaluate the effects of different resource management policies in simulated SDN-enabled cloud data centers. CloudSimSDN is an extension of the CloudSim, and it simulates cloud data center, physical machines, switches, network links, and virtual topologies to measure energy consumption and to ensure QoS. In CloudSimSDN, network bandwidth has an important role related to the SDN features for networking provisioning. Further, physical and virtual topologies can be defined in the simulations. Besides, switches can be managed through an SDN controller, which is responsible for install dynamic forwarding rules dependent on the network traffic. Moreover, A Network Operating System monitors all the network's channels and packets to determine possible reconfigurations of each network element.

To validate CloudSimSDN, Son et al. performed a series of experiments and compared the framework and Mininet emulator with the same workload. They created a tree topology of depth 2 with four hosts. In each scenario, each host is configured to send data with different sizes to the other hosts simultaneously. In the first scenario with fixed data size, the difference between CloudSimSDN and Mininet was at most 2.5%. In the second scenario with variable data size and the same data paths, the differences were below 4.6%. In the third scenario, where the data path consisted of more network elements, the difference slightly increased compared to the first scenario. For the case of the fourth scenario, with the same data path and variable data size, there was only a narrow increase in the difference.

VM consolidation is a way to achieve energy savings in cloud data centers, and it can be simulated in CloudSimSDN. By employing this technique on CloudSimSDN, unused hosts and switches are turned off by the controller. Various VM placement policies such as Best Fit and Worst Fit were evaluated in CloudSimSDN. In the Best Fit policy, VMs are consolidated in a small

number of hosts, and network traffic between hosts can be reduced since more VMs are placed within a host. In the Worst Fit policy, the more available host is the first option to instantiate a VM. The results of comparing the two algorithms demonstrate that 23% of energy consumption was saved when the Best Fit was applied. In addition, consolidation of network traffic and deactivation of idle switches also saved 18% of energy consumption from switches.

DROPV2

In [BLBM14], authors presented a framework named DROPv2, which contains a power management mechanism to achieve energy efficiency in real-world emulated NFV environments. Also it has a number of mechanisms to manage the internal topology, to aggregate or disaggregate network configuration data, like the forwarding information database (FIB) or the power management configurations of elements. DROP is an open source software framework toward NFV/SDN architectures with power management capabilities, and DROPV2 is the extension of the DROP project which was extended to provide an integration environment to native Linux control-plane and administration command-line tools. It can also be integrated with OpenFlow switches.

The architecture of DROPv2 consists of three building blocks: Interconnection Elements (IEs), Forwarding elements (FEs), and Control Elements (CEs). IEs are realized by OpenFlow switches. The external ports of a DROPv2 router reside on the IEs, and it can prepare simple and customizable traffic switching operations to/from internal elements or external ports. IEs perform switching operations and prepare the router external ports. FEs are realized on top of “general-purpose” IT platforms with the Linux Operating System. This block allows a high level of programmability. Forwarding operations reside on FEs and they can be customized among the different general-purpose processors elements. CEs are also realized on top of Linux. These elements are allocated to orchestrate the operations of IEs and FEs by maintaining and updating the router FIB, and to prepare control-plane services to the router.

The CE controller is responsible for configuring all IEs and FEs, managing the delivery of signaling traffic among control processes and router external port. Also, it is responsible for a direct connection to the IEs by using the Open Flow protocol. Authors of DROPv2 evaluated the framework with an experiment in a scenario consisting of one CE controller, one IE, and three FEs (two Xeon servers and one XLP board). The result of the experiment showed that the power management mechanisms of DROPv2 were able to achieve an average energy saving of almost 37% and without any service degradation.

EmPOWER

EmPOWER [RRG13] is an open-source testbed to run experiments of SDN and NFV in WiFi networks. It is a Software-Defined Radio Access Network (SD-RAN) Platform based on open-source software and running on off-the-shelf hardware components. The EmPOWER is currently used by both undergraduate and graduate students at the University of Trento and by the research staff at CREATENET. It allows full control of every experiment, and each one is isolated from others. Further, the researcher can monitor the actual energy consumption of network elements in real-time.

The EmPOWER is an extension of OpenVSwitch and uses the Click Modular Router for the datapath, Floodlight as SDN controller, and an Arduino add-on to monitor energy consumption. EmPOWER provides programmable access points equipped with two interfaces. One of them is connected to the control and management network, which allows researchers to collect network statistics, and perform administrative tasks without affecting the actual traffic of the experiment flowing via the second interface. Besides, researchers can evaluate energy-aware mobility management mechanisms over the realistic infrastructure provided by the testbed. For instance, it is possible to design mechanisms in the controller to concentrate clients on a reduced number of access points so the others can be turned off.

DartCSim+

Li et al. [LJYH13] presented a cloud simulator named DartCSim+. It supports energy-aware network simulations and network-aware live migrations. The main objective to present DartCSim+ is the disability of CloudSim for simulating power and network models simultaneously. DartCSim+ is extended with the Network Interface Card (NIC), which can measure the energy consumption of NICs. Experiments made by the DartCSim+, simulated 50 hosts, each one powered by two Intel Xeon CPU, 64 GB of the DRAM, four hard disks, and 1000Mbps Ethernet card. In terms of network, 10 switches were simulated. Also, the datacenter broker requests 100 virtual machines and submits 1000 cloudlets to run on the datacenter. Then, the experiment analyzed the effect of data transfer on the energy consumption of the network.

The first experiment explores the energy consumed by the hosts and interconnection elements to demonstrate the effect of data transferred on the energy consumption of the network. The authors define α to determine the network intensity of a task as the ratio between the size of network packets for transmission and the number of instructions for execution. The experiment shows the energy consumption of each component in the datacenter as a function of α . Consequently, the result shows that hosts consume 70%, switches consume 25%, NICs consume 3% and links 2% of energy. Therefore, networks (links, NICs, and switches) consume comparable energy with hosts.

GreenCloud

Kliazovich et al. [KBK12] presented GreenCloud as a simulation environment, which is an extension of NS-2 [IH09], and it has a focus on the packet-level simulations of communications in data centers. GreenCloud gathers data about the energy consumption of communication elements of the data center to capture communication patterns of current and future data center architectures. In GreenCloud, a cloud computing data center is a set of computing and communication resources.

The extension delivers accurate modeling of the energy consumed by servers, switches, and links, which are the components of the data center. In terms of network topologies, the simulator can simulate two-tier, three-tier, or three-tier high-speed data center architectures. To allow realistic simulations, GreenCloud considers the state of the CPU to simulate its power consumption by implementing Dynamic Voltage/Frequency Scaling (DVFS). Also, realistic physical placement of elements, like Top-of-Rack (ToR) architecture, which puts computing and communication elements on the same or in adjacent racks, is considered in the simulator.

The results illustrate that by utilizing DVFS, there was a reduction in power consumption to 84% of the nominal level. Besides, most of the reduction comes from downshifting CPU voltage, since CPU components account for 43% of the total energy consumed by the simulated servers.

NetworkCloudSim

NetworkCloudSim [GB11] is an extension of CloudSim [BBA10], which supports applications with communicating elements. The authors designed a network flow model for cloud data centers utilizing bandwidth sharing and latencies, which make this simulator scalable and faster. So, NetworkCloudSim is extended because CloudSim does not support modeling of the networked resources and complex data centers with different applications model (in particular, CloudSim can not model bandwidth and communication between different data centers). Besides, NetworkCloudSim designed NetworkPacket and HostPacket to illustrate the data flow between the VMs in a data center. HostPacket is the packet that moves through the virtual network, whereas NetworkPacket is the packet that moves between the servers. Also, VMs are placed in the host and connected by a virtual network.

Experiments made by the authors demonstrated that NetworkCloudSim is capable of simulating cloud data center networks, applications with communicating tasks, advanced scheduling, and resource allocation mechanisms for clouds. The impacts of the shared network on the performance of data centers can help researchers to optimize the data center usage.

Random-NonOverlap and Random-Overlap are scheduling policies implemented in the simulator. So, In the Random-NonOverlap, the first incoming networkcloudlet would be executed in the VM, and other networkcloudlet have to wait for finishing the currently performing one. On the other hand, Random-Overlap chooses the networkcloudlet in front of the waiting queue to perform if the currently executing task is waiting for the data (packet) to reach from other tasks. The results showed the average response time for Random-Overlap is quite low in comparison to Random-NonOverlap, because of the communication delays in receiving data by the VM. Additionally, with the rise of applications, the average response time in the Random-NonOverlap policy is increased, while it is declined in the Random-Overlap scheduling policy.

Table 3.1 shows a detailed comparative analysis of the seven simulators summarized in this section. According to the table, it is possible to conclude that most of the tools are written in Java and designed for any operating system. Also, in terms of platform, we can conclude the most simulators are extensions of CloudSim.

Parameters	CloudSimSDN-NFV	CloudSimNFV	CloudSimSDN	DROPV2	EmPOWER	DartCSim+	GreenCloud	NetworkCloudSim
OS	Any	Any	Any	Linux	Linux	Linux	Any	Any
Platform	CloudSim	CloudSim	CloudSim	DROPV	OpenVSwitch	CloudSim	NS2	CloudSim
Language	Java	Java	Java	—	Python	Java	C++/OTcl	Java
Availability	Open source	No	Open source	—	—	No	Open source	Open source
Graphical support	×	✓	✓	✓	×	×	×	×
Environment	Simulation	Simulation	Simulation	Testbed	Testbed	Simulation	Simulation	Simulation
Power-aware support	✓	✓	✓	✓	✓	✓	✓	×
Network Model	✓	×	✓	✓	✓	✓	✓	✓
NFV Support	✓	✓	✓	×	✓	×	×	×
Power saving Mode	✓	✓	✓	—	✓	✓	✓	×
VM Support	✓	✓	✓	✓	×	✓	✓	✓
SFC Support	✓	×	×	×	×	×	×	×

Table 3.1: Summary of the simulation related works (A “—” in the table is used when the information was not provided by the authors.)

3.1.1 Summary

On 3.1, we analyzed the simulators in order to utilize them for our research, simulate NFV environments, and Service Function Chaining (SFC) to experiment with different energy-saving mechanisms.

Firstly, we analyzed CloudSimSDN-NFV that is an extension of CloudSimSDN. As a matter of fact, the combination of SDN and NFV can bring some facilities for service providers. Moreover, SDN and NFV can simplify network implementation. SDN provides the connectivity between Virtualized network functions in an automated way that can improve network management. Furthermore, SDN can assist the NFV to implement a Service function Chaining (SFC part in the cloud environment). On the other hand, SDN Controller and Management Applications can run as VNFs in a scalable environment.

The second simulator that we considered was CloudSimNFV (we can not use CloudSim directly because it does not support power and network simulation at the same time). CloudSimNFV can simulate NFV scenarios, already having several scheduling algorithms for NFV applications, and having enough extensibility. Unfortunately, CloudSimNFV is not open source, and it does not consider an important factor, namely the users’ behavior in terms of service consumption and mobility.

Concerning CloudSimSDN, this simulator even though it can easily be extended and it focused on SDN but it does not support NFV and SFC. Although SDN and NFV have different goals, they have a lot in common. Therefore, some data centers apply techniques to mix SDN and NFV to use essential features such as availability, reliability, and elasticity.

The forth simulator considered was DartCSim+. It enhances the CloudSim to integrate the power and network model at the same time, but unfortunately, it does not support NFV components.

GreenCloud was another simulator that we analyzed. Although GreenCloud is an easily understandable simulation to evaluate and simulate all types of networks in a cloud environment,

evaluating NFV-based applications on GreenCloud is not easy because of the low-level specification required by NS-2.

By evaluating DROPV2 and EmPOWER testbeds, which are testbeds with special hardware, we concluded that these two testbeds do not accessible and easy to use for us.

Finally, the last simulator considered was NetworkCloudSim. In this simulator, network elements such as switches and links are implemented and used to estimate network transmission time. However, this simulator focuses on modeling and simulating message-passing applications in a data center that do not include NFV and SFC features.

3.2 Measurements

The first aim of the research in [AHA⁺16] is a better understanding of the virtualization level in the Operating System (OS) level. This work focuses on to recognize the factors that influence the packet delay and throughput of container-based applications and virtual machines, in the context of NFV service chains. Several networking technologies were identified available for routing network traffic between VMs or containers on Linux. These technologies are classified as a software switch or device virtualization. Macvlan is an instance of Network Device Virtualization, which allows the abstraction of a single network interface into multiple network interfaces with different hardware addresses assigned to them. Also, Single Root I/O Virtualization (SR-IOV) is a specification that allows a physical PCIe device to present itself to the OS as multiple separate PCIe devices.

In this work, VNFs can be implemented on the same host or multiple hosts on a network. To evaluate the OS-level virtualization platform, the authors utilized Docker and Xen. Also, The Single Root I/O Virtualization (SR-IOV) and Open vSwitch (OVS) are applied to a physical interface. In the experiments, each physical machine was a Dell C8220 server, with dual Intel Xeon E5-2660v2 10-core 2.20Ghz CPUs, 256GB ECC RAM, and Intel 82599SE 2-port 10Gbe network interface controller (NIC). Machines were co-located on the same rack and connected by a 1Gbps control network. A 10Gbps data network was linked to a Dell Force10 S6000 switch and all machines ran Ubuntu 14.04 LTS. The performance evaluation was made measuring the networking mechanisms in the Docker containers, Xen virtual machines, and directly on the physical host.

The results demonstrated that OS-level virtualization can offer network performance benefits compared to hardware virtualization. Each comparison illustrates the Docker containers had lower latency cost and lower variability than equivalent Xen VMs. Further, during the transmission of packets over a physical network interface, both macvlan and SR-IOV presented lower latency and more predictable variation than the Linux bridge and OVS. Moreover, between containers on the same host, OVS produces similar delay costs, but in the case of SR-IOV, it causes much larger costs by offloading the switching to a PCIe device. Therefore, this result can be intensified by service chaining VNFs, while macvlan and OVS perform similarly even in long chains. On the other hand, in this comparison, the performance of the Linux bridge and Open vSwitch was acceptable for many applications.

The work in [BdSG16] evaluates the performance of ClickOS, CoreOS, and *OS^v*, which are open source solutions for Network Function Virtualization. ClickOS is a Xen-based software platform for fast network packet processing, which can support typical network requirements such as high throughput and low latency. CoreOS deploys functions using Linux containers (LXC) virtualization. LXC focused on functions instead of entire virtualized hosts. *OS^v* is an open-source operating system based on the library OS design, and its standard version is based on the QEMU Kernel Virtual Machine (KVM).

In the experiments, the authors used servers with an AMD 3.6 GHz and 4 GB of memory. VNFs servers are responsible for hosting VMs or containers that will run VNFs. CoreOS showed the smallest Memory utilization, with less than 700 MB used with 128 containers instantiated. Memory utilization of ClickOS was very close to CoreOS, which showed less than 1000 MB of Memory consumption. Finally, *OS^v* depicted the worst memory consumption, which used more than 3500 MB to instantiate 128 VMs. Response time is the main objective of this work which analyzes how

much time each virtualization solution spends to receive, process, and response packets. The result of experiments showed that ClickOS had the smallest response time (around 0.16 milliseconds), and the response time of CoreOS is under 0.5 milliseconds. OS^v had the worst response time (around 3 milliseconds).

In [GHC17], the authors propose the virtualization of a software-defined radio access network (RAN) architecture by using VMs and Docker containers. They used two testbeds *i*) the Kernel-based Virtual Machine (KVM) for virtual machine deployment and *ii*) Alpine Linux version 3.2, which is a 5 MB image, for Docker containers. The goal of this work is to measure the processing rate under the two NFV deployments.

Firstly, they implemented RAN architecture by the OpenDayLight (ODL) SDN controller for RAN features. Also, they used Radio Net Flow (RNF), which is the SB protocol plugin introduced to integrate eNBs to this framework. RNF protocol helps each eNB to establish a connection to the SDN controller. Then, the eNB uses RNFA to send the configuration of information to the controller. The configuration will be taken by the RAN Statistics Manager in order to store RAN statistical information in its database. Besides, the RAN configuration has the responsibility to validate the initial connectivity to the eNB. The experiment results showed that the Docker container-based architecture can provide a better performance in inter-arrival, average waiting time, and Average service rate compared to the VM based architecture.

In [GP18], the author evaluates the efficiency of a Docker container environment and a VM environment. The research uses the ONOS SDN controller application as a network function in a container and VM format to compare performance metrics such as CPU, memory consumption, throughput, and VNF provisioning time. Furthermore, this investigation aims to help network operators to identify a platform with better performance to host SDN-VNF services. The result of the experiments demonstrates that the containerized ONOS VNF showed CPU utilization close to 99.50% in each core (it has two cores), and the memory utilization of 436 megabytes. The CPU utilization for the VM platform was 99.40% of utilization in each core, and the memory utilization by the VM-based platform showed the consumption of 857 megabytes. Therefore, the CPU utilization depicts no significant differences between the VM and the containerized ONOS VNF.

3.2.1 Summary

In 3.2, we studied some related works to recognize the negative and positive side of using containers and virtual machines for virtualization. Initially, we should take into consideration that CPU utilization plays a vital role in our research because energy consumption has a direct relation with CPU utilization. Our study demonstrates the CPU utilization depicts no significant differences between the VMs and the containers.

Also, security is the most important factor for public cloud providers. Tenants share the same resources in the cloud environment, therefore it is critical to ensure that the virtualized network is secure and isolated from other tenants. Consequently, the isolation problem in the container can influence the security of the virtualization since containers do not isolate the resources, and more than one process can be run within each container.

On the contrary, the main advantage of Virtual Machines is isolation. According to our study VMs provide a more isolation environment for users, and it can conceal the resource usage between the VMs. Moreover, the isolation property of VMs provides an independent system for users. Furthermore, the VMs can be easily maintained, recovered, and provisioned.

3.3 Placement Algorithms

Rankothge et al. [RLRL17] propose genetic algorithms to allocate resources for new VNFs provisioning, and scaling of existing VNFs when traffic changes occur. The author used a platform called Network Function center (NFC) which supports the data center with different architecture such as k-fat tree architecture, a BCube architecture, and a VL2 architecture. The goal of resource

allocation in this algorithm is to decrease the required resources, adjust the resources to satisfy the traffic changes and reduce the number of configuration changes. According to reported experiments, a genetic programming approach takes few milliseconds to allocate VNFs in a large network, while a traditional integer linear programming approach can take hours. The experiments also show that the proposed algorithms can decrease the link utilization and, additionally, suggest new allocations with minor changes to the current configuration.

The work in [MGZ16] proposes a model for Virtual Network Function (VNF) placement and chaining across a distributed cloud. Authors provided the adjacency matrices of the requests and the hosting graph in order to gain efficient solutions for VNF placement and chaining. Moreover, the authors consider a greedy algorithm to compare with their solution. The authors compute the convergence time and acceptance rate in order to evaluate the effectiveness of the proposed solution. The first experiment for data centers with 100 servers and the second experiment with 1000 servers show that the greedy algorithm requires more than 1.5 minutes, and the proposed solutions by authors required about 0.00016 minutes to solve the problem for VNF placement. Therefore, the result shows the proposed approach is faster and stable than a greedy heuristic algorithm.

Marotta et al. [MA15] consider an algorithm based on simulated annealing to minimize energy consumption by possible VM migrations. The algorithm ensures low power consumption and the minimum number of active hosts. For instance, experiments showed that the proposal can decrease, on average, the number of active hosts by 27%, and the power consumption by 37%. Moreover, by applying the best fit and first fit policies, the result illustrates the reduction between 31% and 44% in the power consumption.

Liu et al. [LLZ⁺15] study the placement problem to find optimal locations for middleboxes, and to optimize the performance of service chaining. Two heuristic algorithms, a greedy algorithm and an algorithm based on simulated annealing, are proposed considering the end-to-end delay and the bandwidth consumption. Experiments showed that the proposals efficiently reduce 22% of the end-to-end delay and also can save 38% of bandwidth consumption. The performances of algorithms are close to the optimal solutions and suitable for many scenarios to optimize middlebox deployment in the network and to optimize service location in NFV-based applications.

Work in [CZA⁺16] assumed Greedy-NSGA-II, Random-NSGA-II, Greedy-MOGA, and Random-MOGA to study VNF placement problem in a hybrid NFV environment. Authors consider these four Genetic Algorithms to reduce bandwidth consumption and decrease the maximum link utilization when placing VNFs in a hybrid NFV environment. The result of experiments showed that Greedy-NSGA-II has the best performance to decline bandwidth and link utilization. Furthermore, the considered algorithms had better results than a non-genetic algorithm and are efficient solutions for the VNF placement problem.

Table 3.2 compares all the five works presented on this section with our work. It is possible to observe that our work is the only one supporting VNFs, host power consumption, switch power consumption and link utilization at the same time.

Parameters	[RLRL17]	[MGZ16]	[MA15]	[LLZ ⁺ 15]	[CZA ⁺ 16]	Our Work
VNFs support	✓	✓	×	✓	✓	✓
Host power consumption	×	×	✓	×	×	✓
Switch power consumption	×	×	✓	×	×	✓
Link utilization	✓	×	—	×	✓	✓
Data center architecture	BCube k-fat tree VL2	—	Mesh	FatTree-4	—	Three-tier
SFCs support	×	✓	×	✓	×	✓
Approach	Genetic Algorithms	Greedy	Simulated Annealing	Greedy Simulated Annealing	Genetic Algorithms	Standard Hill Climbing Simulated Annealing Memetic Algorithm

Table 3.2: Comparison of our work in relation to the related works

3.3.1 Summary

Following the study of the related works presented in 3.3, we analyzed that scheduling policies such as VNFs placement algorithms are important to reduce energy consumption by the host and network elements. Since we did not find any proposal that considers, at the same time, VNFs, host power consumption, switch power consumption, and link utilization, we propose a set of algorithms to fulfill all these requirements. Therefore, to guarantee good solutions in reduced execution time, the algorithms will be based on metaheuristic approaches.

Chapter 4

Energy Efficient Algorithms for Virtual Network Function Placement

In this chapter we introduce our metaheuristic algorithms designed to decrease the energy consumption during the VNFs placement in data center. Section 4.1 introduces the notation and description of the problem formulation and Section 4.2 presents three algorithms. One based on the Standard Hill Climbing method, one based on Simulated Annealing and one based on a Memetic Algorithm.

4.1 Problem Formulation

The notation used in the problem formulation is presented in Table 4.1. The objective of the problem is to minimize the power consumption of data centers while placing VNFs in virtual machines on these data centers.

Symbol	Description
$H = \{h_1, h_2, \dots, h_n\}$	List of hosts in the data center
$S = \{s_1, s_2, \dots, s_t\}$	List of switches in the data center
$V = \{v_1, v_2, \dots, v_m\}$	List of VNFs to be placed
$E(h_i)$	Energy consumption of host i
E_{idle}	Idle power consumption of host
E_{peak}	Peak power consumption of host
u_i	CPU utilization of host i
$E(s_i)$	Power consumption of switch i
E_{static}	Power consumption of switch without traffic
E_{port}	Power consumption of each port on switch
q_i	Number of active ports on switch i
$rr(v_j)$	Resources requested by VNF j
$BW(v_{j,i})$	Bandwidth requested by VNF j
$M(v_j)$	Processing requested by VNF j
$PE(v_j)$	Processing Elements requested by VNF j
$HDD(v_j)$	Storage requested by VNF j
$R(v_j)$	Memory requested by VNF j
σ_i	number of VNF placed on host i
$BC(h_i)$	Bandwidth capacity of host i
$MC(h_i)$	Processing capacity of host i
$C(h_i)$	Capacity of host i
$HDDC(h_i)$	Storage capacity of host i
$RC(h_i)$	Memory capacity of host i

Table 4.1: Notation

Given a set of hosts $H = \{h_1, h_2, \dots, h_n\}$, where n is the number of hosts in the data center, each $h_i \in H$ has a processing capacity expressed in MIPS (Million Instructions Per Second), bandwidth capacity expressed in bps, RAM size expressed in MB, and storage size expressed in GB. We summarize this information to each host as follows:

$$h_i = \{MC(h_i), BC(h_i), RC(h_i), HDDC(h_i)\}$$

Each SFC includes a set of network functions $V = \{v_1, v_2, \dots, v_m\}$, where m is the number of VNFs to be placed in the hosts. Each $v_j \in V$ requires a minimum processing capacity in *MIPS*, a proper bandwidth in bps to communicate with other VNFs, a minimum available *RAM* expressed in MB, a minimum available storage size expressed in GB, and a throughput of the VNF expressed in *MIPO* (Million Instructions Per Operation). We summarize this information to each $v_j \in V$ as follows:

$$v_j = \{M(v_j), BW(v_j), R(v_j), HDD(v_j), PE(v_j)\};$$

The objective of the algorithms proposed in this dissertation is to minimize the energy consumption by hosts and minimize the energy consumption by switches assuring the minimum requirements of resources by the *VNFs*. Therefore, the list of VNFs will be mapped in a list of hosts in a way that the energy consumed by both hosts and switches is minimized (Equation 4.1).

$$\text{minimize} \sum_{i=1}^{|H|} E(h_i) + \sum_{j=1}^{|S|} E(s_j) \quad (4.1)$$

Subject to the following constraints:

$$\sum_{i=1}^{|H|} \sigma_i = |V| \quad (4.2)$$

$$\forall h_i, \sum_{j=1}^{\sigma_i} rr(v_j) \leq C(h_i) \quad (4.3)$$

Where Equation 4.2 ensures that each VNF is placed only once in a host and, according to Equation 4.3, the resources given to VNF in a host cannot exceed the capacity of the host.

4.1.1 Power Model

CPU utilization percentage has a direct influence on the power consumption of hosts [PMWV09] as stated on Equation 4.4

$$E(h_i) = \begin{cases} E_{idle} + (E_{peak} - E_{idle}) \times u_i & \text{if } \sigma_i > 0 \\ 0 & \text{if } \sigma_i = 0 \end{cases} \quad (4.4)$$

Consuming energy by hosts in the idle time is a constant factor and the workload do not affect it. It can be reduced only if the host is turned off. Moreover, processing workload which leads to higher CPU utilization can increase the energy consumption of a host. The power consumption of a host will be the same to another host if the CPU utilization is the same.

Power consumption of switches is calculated based on their active ports [WYW⁺12] as stated on Equation 4.5:

$$E(s_i) = \begin{cases} E_{static} + E_{port} \times q_i & \text{if } s_i \text{ is on} \\ 0 & \text{if } s_i \text{ is off} \end{cases} \quad (4.5)$$

Regardless of switch network traffic, a switch has a static part in its power consumption. Switch consumes more energy when more ports are active with traffic passing through the switch.

4.2 Metaheuristic Algorithms

In this work, we propose three algorithms based on metaheuristics. The first two are local search algorithms, one based on the Standard Hill Climbing method and the other based on Simulated Annealing. The last algorithm is based on a Memetic Algorithm.

4.2.1 SHC-VNFP Algorithm

Algorithm 3 shows the pseudo-code of the SHC-VNFP, the algorithm based on Standard Hill Climbing method.

The algorithm receives as input a list of hosts and a list of VNFs. It computes the required resources of each VNF via the for loop (line 1). The second for loop computes the available capacity of each host. To find the best host, we used a normalized unit to combine CPU requirements (total MIPS) and bandwidth constraints. CPU requirements and bandwidth constraints are two-dimensional requirements that should be considered at the same time. Therefore, for each host, the algorithm calculates how much *MIPS* are available in percentage (line 5) and how much bandwidth is available, also in percentage (line 6). At the end of the second for loop, the algorithm computes the capacity of each host ($C(h_i)$) (line 7).

The third for loop runs m times in order to place v_j in h_i . At first, we defined the initial cost and the index of the host (line 10 and 11). After that, the algorithm compares the potential of the h_i with initial cost. If the $C(h_i)$ is higher than the initial cost (line 13), the algorithm will use the index of the h_i to place the v_j . Otherwise, the algorithm would seek among the neighbor of h_i to find a suitable host (line 16 to 19).

After finding the index of the host with sufficient capacity, the v_j is placed in this host (line 18), then, the resources used by the v_j will be reduced from the whole capacity of the selected host in order to prevent exceeding the capacity of the host (line 19).

Algorithm 3 SHC-VNFP

Input: V, H
Output: assignment of V in H

```

1: for  $j \leftarrow 1$  to  $m$  do
2:    $rr(v_j) \leftarrow PE(v_j), M(v_j), BW(v_j)$ 
3: end for
4: for  $i \leftarrow 1$  to  $n$  do
5:    $freeMIPSpercentage \leftarrow freeMIPS(h_i)/MC(h_i) \times 100\%$ 
6:    $freeBCpercentage \leftarrow freeBC(h_i)/BC(h_i) \times 100\%$ 
7:    $C(h_i) \leftarrow freeMIPSpercentage \times freeBCpercentage$ 
8: end for
9: for  $j \leftarrow 1$  to  $m$  do
10:   $initialCost \leftarrow -\infty$ 
11:   $Idx \leftarrow -1$ 
12:  for  $i \leftarrow 1$  to  $n$  do
13:    if  $C(h_i) > initialCost$  then
14:       $initialCost \leftarrow C(h_i)$ 
15:       $Idx \leftarrow i$ 
16:    end if
17:  end for
18:  place  $v_j$  in  $h_{Idx}$  ▷ Allocation takes place here
19:   $C(h_{Idx}) \leftarrow C(h_{Idx}) - rr(v_j)$  ▷ The free resources on each host is updated removing the amount used by VNF  $v_j$ 
20: end for

```

4.2.2 SA-VNF Algorithm

Algorithm 4 shows the pseudo-code of the SA-VNFP, the algorithm based on Simulated Annealing.

The operation of the simulated annealing contains a set of parameters to control the process of SA-VNFP. The parameters include the initial temperature T_{min} , the way of temperature reduction α , a certain temperature T , and an incremental counter. The set of parameters are the key which ensures the algorithm produces the acceptable optimal solution in a defined time. The acceptable optimal solution is checked based on the satisfaction of the Metropolis criterion, and this procedure iterates until convergence. The Metropolis criterion is a set of conditions which improve the solution.

First of all, we modeled the annealing parameters, including the cooling rate α (line 2), the minimum temperature T_{min} (line 3), and the final temperature T , which is equal to the number of hosts n (line 4) based on the related works [WC13] and [Ban97].

The algorithm computes the required resources of each VNF via the for loop (line 5). The second for loop computes the available capacity of each host. For each host, the algorithm calculates how much *MIPS* are available in percentage (line 8) and how much bandwidth is available, also in percentage (line 9). At the end of the second for loop, the algorithm computes the capacity of each host ($C(h_i)$) (line 10).

The third for loop runs m times to place v_j in h_i . For this reason, the first host is chosen in order to generate a *Current* solution (line 13). After selecting *Current* solution, the algorithm generates a feasible neighbor host as a *Candidate* solution and then, computes the capacity of the *Candidate* solution to compare with the *Current* solution (line 16). If the capacity of the *Current* host is a less than *Candidate* host (feasible neighbor) (line 17), this feasible neighbor can be accepted or rejected according to a acceptance probability.

Lines 18 and 19 are responsible to ascertain the acceptance probabilities by the Metropolis method. First, in line 18 the algorithm calculates ρ as an acceptance probability, where T is the current temperature. After, we create a random number in line 19. If the ρ is greater than the random number, the solution *Candidate* can be accepted as the current solution. Therefore, the index of *Candidate* host will be the current optimization of h_i to create the instance of the v_j . Then, the T would be decreased by cooling rate (line 25).

After placing v_j in the host (line 27), the resources used by the v_j will be reduced from the whole capacity in order to prevent exceeding the capacity of the host (line 28).

Algorithm 4 SA-VNFP

Input: V, H
Output: assignment of V in H

```

1:  $\alpha \leftarrow 0.99993$ 
2:  $T_{min} \leftarrow 0.00001$ 
3:  $T \leftarrow n$ 
4: for  $j \leftarrow 1$  to  $m$  do
5:    $rr(v_j) \leftarrow PE(v_j), M(v_j), BW(v_j)$ 
6: end for
7: for  $i \leftarrow 1$  to  $n$  do
8:    $freeMIPSpercentage \leftarrow freeMIPS(h_i)/MC(h_i) \times 100\%$ 
9:    $freeBCpercentage \leftarrow freeBC(h_i)/BC(h_i) \times 100\%$ 
10:   $C(h_i) \leftarrow freeMIPSpercentage \times freeBCpercentage$ 
11: end for
12: for  $j \leftarrow 1$  to  $m$  do
13:    $Current \leftarrow C(h_1)$ 
14:   while  $T > T_{Min}$  do
15:     for  $i \leftarrow 1$  to  $n$  do
16:        $Candidate \leftarrow C(h_i)$ 
17:       if  $Current < Candidate$  then
18:          $\rho \leftarrow \text{Math.exp}(-\text{Math.abs}(Candidate - Current)/T)$ 
19:         if  $(\text{Math.random}() < \rho)$  then
20:            $Current \leftarrow Candidate$ 
21:            $Idx \leftarrow i$ 
22:         end if
23:       end if
24:     end for
25:      $T \leftarrow T \times \alpha$ 
26:   end while
27:   place  $v_j$  in  $h_{Idx}$  ▷ Allocation takes place here
28:    $C(h_{Idx}) \leftarrow C(h_{Idx}) - rr(v_j)$  ▷ The free resources on each host is updated removing the amount used by VNF  $v_j$ 
29: end for

```

4.2.3 Mem-VNFP Algorithm

Algorithm 5 shows the pseudo-code of the Mem-VNFP, the algorithm based on Memetic Algorithm.

Initially, the algorithm receives a set of H and a set of V as input. In lines 1 to 3, we initialize variables related to the evolutionary aspects of the algorithm (iterator, population size and chromosome size). The population size is equal to the number of hosts, and each chromosome consists of 10 hosts.

The first for loop computes the required resources of each VNF (line 4). In line 7, the second for loop computes the available capacity of each host. For each host, the algorithm calculates how much *MIPS* are available in percentage (line 8) and how much bandwidth is available, also in percentage (line 9). At the end of the second for loop, the algorithm computes the capacity of each host ($C(h_i)$) (line 10).

Line 14 is responsible for creating initial population. The population size depends on the number of hosts in the data center and contains a set of chromosomes. Each host in the data center represents each chromosome in the population. Figure 4.1 illustrates a chromosomes representation. by using a list with “k” dimension: $Ind = \{Ind_1, Ind_2, \dots, Ind_k\}$.

Figure 4.1: Representation and capacity of the chromosomes in the population.

Chromosomes are the individuals in the population which are used to explore the space of placement. In the chromosome structure, the space of the placement is a host, and the last space in the chromosome represents the capacity of hosts. The capacity of hosts updates after initializing the population in line 15.

After initialized and evaluated the population, the algorithm select parents to produce the child (line 18). To do this, we applied the roulette wheel selection method to choose the chromosomes. By using roulette wheel selection, we will generate a new population from the original population.

The roulette is generated by adding the capacity of each host into a proportion of the wheel. In fact, roulette maintains a list of capacities of hosts which represent the probabilities of each host. If $C(h_i)$ is the capacity of h_i in the population, its probability of being selected is according to Equation 4.6.

$$P_i = \frac{C(h_i)}{\sum_{i=1}^N C(h_i)} \quad (4.6)$$

where N is the number of hosts in the population.

Once a parent's gene is selected from original population by the roulette wheel selection method, then, the chromosome's recombination method starts to make a child gene from the parent's gene to create new population (line 19).

In line 20, the algorithm uses the local search procedure strategy. The local search algorithm is an iterative algorithm that starts with an arbitrary solution to a problem (line 21), then attempts to find a better solution by incrementally changing the solution. If the change produces a better solution (line 24), this better solution will be considered as a new solution (line 25). Then, the index of host which has a maximum capacity will be the current optimization of h_i to create the instance of the v_j .

Line 29 is responsible to replace chromosomes with lower capacity in the original population by the better chromosomes in the new population. In fact, this function would increase the probability of a better VNF placement.

The v_j will be placed in the host (line 32), and the resources used by the v_j will be reduced from the whole capacity in order to prevent exceeding the capacity of the host (line 33).

Algorithm 5 Mem-VNFP**Input:** V, H **Output:** assignment of V in H

```

1:  $Iterator \leftarrow 10$ 
2:  $populationsize \leftarrow n$ 
3:  $chromosomesize \leftarrow 10$ 
4: for  $j \leftarrow 1$  to  $m$  do
5:    $rr(v_j) \leftarrow PE(v_j), M(v_j), BW(v_j)$ 
6: end for
7: for  $i \leftarrow 1$  to  $n$  do
8:    $freeMIPSpercentage \leftarrow freeMIPS(h_i)/MC(h_i) \times 100\%$ 
9:    $freeBCpercentage \leftarrow freeBC(h_i)/BC(h_i) \times 100\%$ 
10:   $C(h_i) \leftarrow freeMIPSpercentage \times freeBCpercentage$ 
11: end for
12: for  $j \leftarrow 1$  to  $m$  do
13:   for  $i \leftarrow 1$  to  $n$  do
14:     Create the original population
15:     Evaluate the original population
16:      $t \leftarrow 0$ 
17:     while  $t < Iterator$  do
18:       Select parents from original population
19:        $P_{new} \leftarrow$  Reproduce new population
20:       while  $k < populationsize$  do
21:          $Current \leftarrow -\infty$ 
22:          $Idx \leftarrow -1$ 
23:         while  $i < chromosomesize - 1$  do
24:           if  $C(h_i) < Current$  then
25:              $Current \leftarrow C(h_i)$ 
26:              $Idx \leftarrow i$ 
27:           end if
28:         end while
29:       end while
30:       Renew original population and new population
31:        $t \leftarrow t + 1$ 
32:     end while
33:     place  $v_j$  in  $h_{Idx}$  ▷ Allocation takes place here
34:      $C(h_{Idx}) \leftarrow C(h_{Idx}) - rr(v_j)$  ▷ The free resources on each host is updated removing the amount
      used by VNF  $v_j$ 
35:   end for
36: end for

```

Chapter 5

Performance Evaluation

The experimental analysis of our proposed algorithms was carried out by using the version of CloudSimSDN-NFV [SB] available at April 5th, 2019. We extended it to enable the implementation of metaheuristic algorithms and named this extension as Meta-CloudSimSFC. Figure 5.1 shows the architecture of CloudSimSDN.

Figure 5.1: *CloudSimSDN Architecture*

First of all, the user code and scenarios can be supplied by the users of the CloudSimSDN framework. Also, physical and virtual topology configurations can be supplied either as program codes or as JSON files. Also, the end-users' requests, which compose the input workload, can supply in CSV files. Furthermore, the VM placement algorithm and network policies should be prepared as the scheduling policies. Also, Brokers can simulate the behavior of data centers or users.

In the Resource Provisioning layer, VM Provisioning provision VMs within the data center according to the VM placement policy and the Network provisioning performs base on the network policies. VM Services layer manages VMs and network, by calculating application execution and packet transmission time between VMs. The next layer, the Resource Allocation allocates resources specified in the bottommost layer of the architecture, Cloud Resources. The new extension of the framework called CloudSimSDN-NFV is a discrete event-driven simulation framework implemented in Java language. Components in NFV and SFC are designed as Java Classes, and it can be extended based on the requirements.

Figure 5.2 demonstrates the class of the framework to simulate the service function chaining. SFC is a chain of multiple service functions, which consist of numbers of VNFs, as an ordered and directed list. SFC execution is determined in ServiceFunctionForwarder which forwards the matched packet to the chain of SFs. In fact, every ServiceFunctionChainPolicy has to be checked by the forwarder in the current simulation configuration, which is composed of the source and destination VMs. For instance, if a network flow from VM1 to VM2 has to go through a chain of VNF1 and VNF2, the SFC has to be contained VNF1, VNF2, and also executing a policy to redirect the network flow from VM1 to VM2 to pass through the defined SFC. SFCs and executing policies are defined in the simulation configuration, loaded and deployed when the simulation started and enforced for all network traffics matching the policy. ServiceFunctionAutoScaler retrieves the average end-to-end latency for packets in the SFC and the utilization of SFs in the SFC.

Figure 5.2: Class Diagram for simulating Service Function Chaining.

To see the placement of each VNFs, We designed a tree topology with four hosts (Figure 5.3). Our first data center topology contains one core switch, which is connected to two edge switches. Also, two edge switches have four hosts. Besides, the virtual topology includes 4 service function chaining. As well as, To see the placement of each VNFs, We designed a tree topology with four hosts (Figure 5.3). Our first data center topology contains one core switch, which is connected to two edge switches. Also, two edge switches have four hosts. Besides, the virtual topology includes 4 service function chaining. As well as, workloads are generated based on a typical web service model that contains Web Server, App Server, and DB Server. Each user has an application running on Web Server, App Server, and DB Server. The table 5.1 demonstrates the Configuration of each web service model, and the instance of the web service creation requests are generated based on randomly selected web service types. Therefore, We create a web application consisting of 1 web, 1 app, and 1 database (DB) to form a simple application with the detailed specification described in Table 5.1.

Table 5.1: Web service model configuration

Web applications	PE	M
Web server	8 vCPU	4000 MIPS per vCPU
App server	4 vCPU	4000 MIPS per vCPU
Database	12 vCPU	4000 MIPS per vCPU

To create SFs chain, we generate 4 VNFs (VNF1, VNF2, VNF3, VNF4) which are enforced by 4 different types of SFC policies. Network traffics between the web services (Web, App, DB) are forwarded to different SFCs consisting of a subset of the 4 VNFs as shown in Table 5.2. The instances of the SFCs were randomly generated from these four options following a uniform distribution. The arrival process of the selected SFCs to be allocated was randomly defined following a uniform

distribution.

Table 5.2: *SFCs configuration*

Source	Destination	SFC Chain
Web server	App server	VNF1
App server	Database	VNF1-VNF2
Database	App server	VNF2-VNF3-VNF4

Initially, from the webserver to the AppServer, the packets go through the VNF1. In the second chain, the request from AppServer to DataBase is sent through VNF1 and VNF2. Finally, in the third chain from DataBase to AppServer, the packet goes through the VNF2, VNF3, and VNF4. Figure 5.3 demonstrates the location of each application and SFs on hosts.

Figure 5.3: *Simple Data Center configuration for the validation test.*

In the second step, we designed larger web applications composed of 4 web servers, 12 application servers, and 2 database servers (Table 5.3). Moreover, our new service function chain consists of 3 SFC policies that manage the network traffic to go through a chain of 7 SFs, such as VNF1, VNF2, VNF3, VNF4, VNF5, VNF6, and VNF7. These long policies that we designed for this performance are not typical in the real world.

Then, we applied this web application and SFCs chain to the three-Tier data center with 4 hosts, but the experiment was failed because of the lack of hosts. Later, we increased the number of hosts to 5, and we redo the experiment, but it failed since the algorithm can not find adequate hosts to place all of the applications and SFCs chain. Also, the experiment failed when we used the Three-Tier data center with 6 hosts. Ultimately, when we increase the number of hosts to 7, the algorithm could locate all of the applications and SFCs chain in the hosts, and the experiment has finished with success.

Table 5.3: *Second SFCs configuration*

Source	Destination	SFC Chain
Web server	App server	VNF1
App server	Database	VNF1-VNF2-VNF3-VNF4
Database	App server	VNF2-VNF3-VNF4-VNF5-VNF6-VNF7

To simulate Metaheuristic VNF allocation policies, we designed two scenarios. First of all, we extended the CloudSimSDN-NFV to support our policies, and also model two types of data center

architectures, application scenarios, and workload. Our extension is available as free software¹. The proposed algorithms are evaluated in a simulation environment to show the effectiveness of the metaheuristic algorithms. Besides our three proposed algorithms, we also used a simple random selection method to serve as a baseline and compare with our three algorithms. The random selection method chooses a host randomly to allocate each VNF. We compared all the algorithms in terms of energy consumption of hosts and switches, response time a user would experience if the allocation was used, and the execution time for the algorithm to return the allocation.

The following will be explained scenarios, DC parameters, and VNF chain parameters that were used in the simulations. The cloud data center configuration will be described on Section 5.1. In Section 5.2, we will explain the workload used. Sections 5.3, 5.4, and 5.5 shows the results obtained with the experiments.

5.1 Data Center Topology

In the following, we used the Three-Tier and Fat-Tree topology to demonstrates energy consumption by two different data center network architecture.

5.1.1 First scenario

In the first scenario, we designed a three-tier data center architecture with different sizes. In reality, The data center size depends on the number of servers and switches.

- *The first Three-Tier DCNA parameters* : Figure 5.6 illustrates the tree-tier topology data center network with 100 hosts connected through 10 edge switches and 1 core switch. Each edge switch is connected to 10 hosts.

Figure 5.4: Three-Tier data center topology with 100 hosts

- *The second Three-Tier DCNA parameters* : The second data center contains 50 hosts connected through 5 edge switches and 1 core switch. Also, each edge switch is connected to 10 hosts. Figure 5.5 shows the second data center architecture.

Figure 5.5: Three-Tier data center topology with 50 hosts

¹<https://github.com/fatemehm/Meta-CloudSimSFC>. Accessed at November 17th, 2019

- *The third Three-Tier DCNA parameters* : The second data center contains 40 hosts connected through 4 edge switches and 1 core switch. Also, each edge switch is connected to 10 hosts.

Figure 5.6: Three-Tier data center topology with 40 hosts

5.1.2 Second scenario

Subsequently, the Fat-Tree topology is the second data center architecture which we used in our experiment.

- *The third Fat-Tree DCNA parameters* : The third data center topology composed 128 hosts which connected with 32 edge switches, 32 aggregation switches, and 16 core switches. Also, each pod contains 4 edge and 4 aggregation switches, and each edge switch connects 4 hosts (Figure 5.7).

Figure 5.7: Fat-Tree data center topology with 128 hosts

- *The second Fat-Tree DCNA parameters* : Figure 5.8 shows the Fat-Tree data center which consists 7 pod, 63 hosts connected with 21 edges, 21 aggregations, and 9 core switches. Each pod composed 3 edges and 3 aggregations switches, and each edge switch connect 3 hosts.

Figure 5.8: Fat-Tree data center topology with 63 hosts

- *The third Fat-Tree DCNA parameters* : *The first Fat-Tree DCNA parameters* : This data center contains 5 pod, 20 hosts which are connected with 10 edge switches, 10 aggregation switches, and 4 core switches. Also, each pod contains 2 edge and 2 aggregation switches, and each edge switch connects 2 hosts (Figure 5.9).

Figure 5.9: Fat-Tree data center topology with 20 hosts

Table 5.4 demonstrates the configuration of each data center. Hence, Each host is configured with 16 cores, and each core has 10,000 MIPS capacity. Also, the network bandwidth between hosts and switches are configured with 2Gbps and 0.1msec of latency.

Table 5.4: data centers configuration

	Number of hosts	Number of switches	HDD	R	MIPS	PE	BW
First Three-Tier DCNA	100	11	10GB	10GB	10000	16	2Gbps
Second Three-Tier DCNA	50	6	10GB	10GB	10000	16	2Gbps
Third Three-Tier DCNA	40	5	10GB	10GB	10000	16	2Gbps
First Fat-Tree DCNA	128	80	10GB	10GB	10000	16	2Gbps
Second Fat-Tree DCNA	63	51	10GB	10GB	10000	16	2Gbps
Third Fat-Tree DCNA	20	24	10GB	10GB	10000	16	2Gbps

5.2 Application Scenario and Workload

For virtual topology, we use 3-tier web applications consisting of web server, application server (app), and database servers (DB) to form an entire application. There are 8 web servers, 24 application servers, and 2 database servers. All servers receives workloads consisting of CPU processing and network transmission. The workload is generated based on the Wikipedia traces in German language [TSCB19]. When web server receives the request from end-users, the request is sent to an application server to retrieve the information from DB. Then, the DB responds to the app server with the relevant information and the information will be returned to the end-user through the web server. According to this process, we generate different VNFs such as firewall (FW), load balancer (LB), and intrusion detection system (IDS). The specific detail of VNFs are depicted in Table 5.5.

Table 5.5: VNF instances

Instance Type	HDD	PE	M	RAM	BW
Firewall	8 GB	8 vCPU	4000 MIPS per vCPU	8GB	800 Mbps
LB	8 GB	4 vCPU	4000 MIPS per vCPU	8GB	20 Mbps
IDS	8 GB	8 vCPU	4000 MIPS per vCPU	8GB	200 Mbps
NAT	8 GB	8 vCPU	4000 MIPS per vCPU	8GB	200 Mbps

We generated 4 SFCs policies which is generated randomly based on four VNFs. In the first chain, from the web server to app server, the packets go through the Network Address Translation (NAT) to enables private IP networks ,the firewall (FW) to filter the request, and the load balancer (LB) to distribute the request across the app servers. In the second chain, the request from app server to DB server is sent through a load balancer (LB) and intrusion detection system (IDS). For the response from DB to app, the packet goes through the IDS and LB in the opposite direction. From app to web, the traffic goes through only LB because firewall is not necessary for the response packets. These four different SFCs are detailed on Table 5.6.

Table 5.6: *Performance requirements for the SFCs*

Source	Destination	SFC Chain
Web server	App server	NAT-FW-LB
App server	Database	LB-IDS
Database	App server	IDS-LB
App server	Web server	LB

5.3 Analysis of Execution Time

Figure 5.10 depicts the average execution time of the proposed algorithms for the different Three-Tier data center. Indeed, the execution time illustrates the time spent by the system for executing the task by each algorithm. Therefore, we repeated the execution for each algorithm 100 times to calculate the average and the standard deviation. The standard deviation is presented as error bars in the graphs (In all the next graphs, error bars demonstrate the standard deviation). All the experiments were executed on a computer powered with a quad-core Intel i5, 2.50 GHz and 8GB of memory, running Ubuntu 18.10. The Java version was 1.8.0-201.

Figure 5.10: Average execution time by Three-Tier data center architecture.

We noticed that the average execution time is declined a little bit for the BestFit algorithm compared with our proposed algorithms when dealing with different numbers of hosts and switches. Also, we note that in total, the RandomSelection depicts a slower execution time, while the BestFit algorithm shows a faster execution time. On average, the number of hosts and switches do not have a direct influence on execution time.

Following that, figure 5.11 illustrates the average execution time for the different Fat-Tree data center network architecture. The experiments demonstrate the execution time of the BestFit algorithm is less than other algorithms. Also, we observed that RandomSelection algorithm repeatedly demonstrates slower execution time compared to the other four algorithms. In general, in Random Selection with 63 hosts and 51 switches and SA-VNFP with 20 hosts and 24 switches, the dispersion of the execution times was greater.

Figure 5.11: Average execution time by Fat-Tree data center architecture.

Table 5.7 show details of average execution time of both Three-Tier and Fat-Tree data center network architecture. Ultimately, we can conclude that the average execution time of both Three-Tier and Fat-Tree data centers totally is close, and the execution time was not affected by the size of the data centers. Also, in general, there isn't a pattern related to the network size.

Table 5.7: Average execution time(milliseconds)

Algorithms	Three-Tier			Fat-Tree		
	40Hosts	50 Hosts	100 Hosts	20 Hosts	63 Hosts	128 Hosts
BestFit	11.0	11.36	10.81	11.27	10.72	11.72
RandomSelection	11.65	11.3	12.15	12.2	12.2	11.55
SHC-VNFP	11.85	11.65	11.45	12.1	11.6	11.9
SA-VNFP	11.45	11.7	11.3	11.9	11.35	12.0
Mem-VNFP	11.9	11.7	11.35	12.0	11.45	11.65

5.4 Analysis of Energy Consumption by Hosts

We also evaluated our proposed algorithms in terms of energy consumption. Figure 5.12 shows the energy consumption for hosts in the three-tier data centers. On average, Mem-VNFP and SA-VNFP presented the best results, with the first being the best of the two. For example, with 50 hosts, the energy consumption by the allocation returned by Mem-VNFP was 60.40% lower when comparing with BestFit. In the same configuration, the energy consumption by SA-VNFP was 58.57% lower when comparing with BestFit.

Figure 5.12: Hosts' energy consumption by Three-Tier data center architecture.

As shown in Table 5.8, when we used Three-Tier data center with 40 hosts the results demonstrate that the Mem-VNFP algorithm consumes energy around 20292.68 Wh had energy saving up to 60.50% compared with BestFit algorithm. Then, SA-VNFP algorithm with 21287.16 Wh energy consumption and it had power saving up to 54.67%. Additionally, notice how the SHC-VNFP algorithm consumes more energy than the others.

Then, by analyzing the Three-Tier data center with 50 hosts, we found the Mem-VNFP saved energy around 60.40% compared with BestFit algorithm. Also, SA-VNFP consumes energy about 21287.34 Wh and it saved power up to 58.57% compared with BestFit algorithm. On the other hand, we found SHC-VNFP used more energy than our base algorithm.

Ultimately, the result demonstrates the Mem-VNFP, with 58.59% energy saving, consumed less energy than BestFit algorithms, when we used data center with 100 hosts. Besides, SA-VNFP which used more hosts than other, consumed 21292.79 Wh and it had up to 58.56% power saving compared with BestFit algorithm. Nevertheless, the SHC-VNFP algorithm depicts the more energy consuming than other algorithms. Accordingly, the result represented that Mem-VNFP strategy is more energy-efficiency.

Table 5.8: Average hosts' energy consumption by Three-Tier data center

Algorithms	Hosts' energy consumption(Wh)			Energy saving		
	40 Hosts	50 Hosts	100 Hosts	40 Hosts	50 Hosts	100 Hosts
BestFit	51384.34	51384.34	51384.34	Ref.	Ref.	Ref.
RandomSelection	46961.46	49442.47	50978.63	8.61%	3.78%	0.79%
SHC-VNFP	51515.15	51515.17	51515.17	-0.25%	-0.25%	-0.25%
SA-VNFP	21287.16	21287.34	21292.79	58.57%	58.57%	58.56%
Mem-VNFP	20292.68	20346.27	21275.35	60.51%	60.4%	58.59%

Figure 5.13 shows the average energy consumption for hosts using Fat-Tree data center architecture with different numbers of hosts. Similar to the results with three-tier data centers, Mem-VNFP and SA-VNFP achieved the best results. However, in average, SA-VNFP was the best algorithm in two configurations. With 20 hosts, the energy consumption by the allocation returned by SA-VNFP

was 0.12% lower compared to Mem-VNFP. With 128 hosts, it was 2.33% lower. With 63 hosts, the allocation returned by Mem-VNFP provided a saving of 0.63% compared to SA-VNFP.

Figure 5.13: Hosts' energy consumption by Fat-Tree data center architecture.

As shown in table 5.9 shows the energy consumption of our proposed algorithm, when we used Fat-Tree data center with 20,63, and 128 hosts. First of all, we used the data center with 20 hosts and we noticed that the SA-VNFP saved energy around 58.79% compared with BestFit algorithm. Although, we found that the SHC-VNFP algorithm consumes more energy than the other algorithms.

Moreover, in contrary to the Fat-Tree data center with 20 hosts, the data center with 63 hosts indicate the Mem-VNFP had energy saving up to 58.82% comparing to BestFit algorithm. While, SA-VNFP algorithm consumes energy about 21287.34 Wh and it saved power up to 58.56% compared with BestFit algorithm.

Ultimately, the result demonstrates, using the Fat-Tree data center with 128 hosts, the SA-VNFP, with 58.56% energy saving, consumed less energy than BestFit algorithms. Besides, Mem-VNFP consumed 21801.10 Wh and it saved power up to 58.73% compared with BestFit algorithm. Consequently, the result depicts that Mem-VNFP and SA-VNFP strategy are more energy-efficiency than BestFit algorithm. However, SA-VNFP totally saved maximum energy compared with other algorithms.

Consequently, the result depicts that Mem-VNFP and SA-VNFP strategy are more energy-efficiency than our proposed algorithm. However, SA-VNFP saved maximum energy compared with other three algorithms.

Table 5.9: Average hosts' energy consumption by Fat-Tree data center

Algorithms	Hosts' energy consumption(Wh)			Energy saving		
	20 Hosts	63 Hosts	128 Hosts	20 Hosts	63 Hosts	128 Hosts
BestFit	51384.34	51384.34	51384.34	Ref.	Ref.	Ref.
RandomSelection	46961.46	49442.47	52011.62	8.6%	3.77%	-1.2%
SHC-VNFP	51458.17	51465.17	51515.17	-0.14%	-0.15%	-0.25%
SA-VNFP	21178.49	21292.79	21292.79	58.79%	58.56%	58.56%
Mem-VNFP	21205.44	21158.95	21801.10	58.73%	58.82%	57.57%

5.5 Analysis of Energy Consumption by Switches

Figure 5.14 shows the average energy consumed for all core and edge switches of the entire Three-Tier data center architecture with different numbers of switches. The BestFit algorithm is more energy-efficient, in terms of switches, when compared with all the other algorithms. Comparing the three proposed algorithms in the best case scenarios, they produced an increase in energy consumption of 62,05% (Mem-VNFP with 5 switches), 49,31% (SHC-VNFP with 5 switches), and 38,83% (SA-VNFP with 5 switches).

Figure 5.14: Average switches' energy consumption by Three-Tier data center

As shown in Table 5.10, when we used Three-Tier data center with 40 hosts and 5 switches, the results depicts that the Best algorithm consume energy around 1548.40 *Wh*, and it saved energy around 27.96%, 38.28% compared with SA-VNFP and Mem-VNFP. Also, it saved energy around 33.02%, 39.44% compared with SHC-VNFP and RandomSelection.

Also, the Three-Tier data center with 6 switches shows that the BestFit algorithm had energy saving up to 32.50%, 42.96% compared with SA-VNFP and Mem-VNFP. Also, BestFit saves energy around 38.28%, 45.87% comparing to SHC-VNF and RandomSelection.

Ultimately, by analyzing the results of the data center with 11 switches, the result shows BestFit with 32.53%, 59.82% energy saving, consumed less energy than SA-VNFP and Mem-VNFP. Besides, BestFit had up to 39.32% and 59.77% power saving compared with SHC-VNFP and RandomSelection algorithm. Accordingly, the result represented that BestFit strategy is more energy-efficiency.

Table 5.10: Average switches' energy consumption by Three-Tier data center

Algorithms	Switches' energy consumption(Wh)		
	5 Switches	6 Switches	11 Switches
BestFit	1548.40	1548.40	1548.40
RandomSelection	2557.06	2860.99	3849.33
SHC-VNFP	2311.96	2509.14	2551.98
SA-VNFP	2149.60	2294.22	2295.26
Mem-VNFP	2509.14	2714.89	3854.47

Figure 5.15 shows the average energy consumption of the Fat-Tree data center architecture with different number of core and edge switches. The Three-Tier data centers that we used for our second scenario contain 24,51, and 80 switches. As shown in figure 5.15, the SA-VNFP algorithm demonstrates the best result for data centers with 51 and 80 switches. While, regarding 24 switches, the Mem-VNFP consumed less energy than others.

Figure 5.15: Average switches' energy consumption by Fat-Tree data center.

As shown in Table 5.11, the energy consumption of Mem-VNFP for data center with 24 switches is 7638.65 Wh which had a power saving up to 7.83% and 9.4% compared with BestFit and RandomSelection. Then, SHC-VNFP algorithm with 8113.05 Wh energy consumption, and it saved energy around 3.78% and 2.1% compared with a RandomSelection and BestFit algorithm. On the other hand, we noticed that the SA-VNFP and RandomSelection consumes more energy than our base algorithm. However, in terms of Fat-Tree data center including 51 and 80 switches, our three proposed algorithm consumed more energy than the BestFit algorithm.

Accordingly, we can conclude BestFit algorithm is more energy-efficiency when we used data centers with 51 and 80 switches. Moreover, in the case of 24 switches, the result represented that Mem-VNFP strategy saved more energy than SA-VNFP, SHC-VNFP, RandomSelection, and BestFit algorithm.

Table 5.11: *Average switches' energy consumption by Fat-Tree data center*

Algorithms	Switches' energy consumption(Wh)		
	24 Switches	51 Switches	80 Switches
BestFit	8287.61	6415.36	4747.18
RandomSelection	8432.02	10590.43	11307.53
SHC-VNFP	8113.05	8655.90	7340.26
SA-VNFP	9479.54	8637.48	7311.86
Mem-VNFP	7638.65	9270.71	10897.11

5.6 Analysis of total energy consumption

Figure 5.16 demonstrate the total energy consumption of the Three-Tier data center architecture. As shown in the figure, When we used a data center with 100,50 and 40 hosts, the SA-VNFP consumes less energy than others. Then, Mem-VNFP is a second algorithm that used less energy than SHC-VNFP, RandomSelection, and BestFit algorithm. The SHC-VNFP and the RandomSelection algorithms altogether consumes more energy than the BestFit algorithm.

Figure 5.16: *The total energy usage of Three-Tier data center.*

Figure 5.17 depicts the total energy consumption of the Fat-Tree data center architecture. According to the figure, the SA-VNFP algorithm depicts less energy consumption when it applied to the fat-tree data center with 128,63 and 20 hosts. Also, Comparing with SHC-VNFP, the RandomSelection, and the BestFit algorithm, the Mem-VNFP is our second energy-efficient algorithm that uses less energy than these three algorithms. Besides, the BestFit algorithm consumes less energy than the SHC-VNF and the RandomSelection algorithm.

Figure 5.17: The total energy usage of Fat-Tree data center.

Table 5.12 illustrates the average of total energy consumption of both the Three-Tier and Fat-Tree data centers by detail. Comparing the Three-Tier data centers with Fat-Tree data centers show the SA-VNFP saved energy up to 20.80%, Mem-VNFP 17.70%, SHC-VNFP 9.56%, the RandomSelection 17.23%, and the energy saving of the BestFit algorithm is around 11.29%.

Also, the experiment demonstrates the Three-Tier data center with 50 hosts and 6 switches is more energy-saving than the Fat-Tree data center with 63 hosts and 51 switches. In fact, the energy saving of the SA-VNFP algorithm is 21.18%, and the Mem-VNFP algorithm saved 21.19% of the energy. Also, the SHC-VNFP algorithm shows the energy-saving around 10.15%. Moreover, the RandomSelection algorithm saved energy up to 15.69%. Furthermore, the energy saving of the BestFit algorithm is up to 8.42%. Accordingly, we can infer that the Mem-VNFP with 23588.05 *Wh* energy consumption is energy-efficient than other algorithms when we used data centers with average numbers of hosts and switches. As a conclusion, the results show the total energy consumption of the Three-Tier data center is less than a Fat-Tree data center.

Eventually, we compared the Three-Tier, which contains 100 hosts and 11 switches, with a large-scale Fat-Tree data center, which includes 128 hosts and 80 switches. So, the results show, the SA-VNF saved energy up to 5.69%. Moreover, the Mem-VNFP shows energy saving around 21.84%. Furthermore, the energy-saving of the SHC-VNFP algorithm is 8.05%, and also the RandomSelection algorithm saved 12.97% of the energy. Besides, the BestFit algorithm saved energy around 12.97%. Thus, the Mem-VNFP algorithm with 25129.83 *Wh* shows more energy-saving than other algorithms, when we used a large-scale Three-Tier data center. Therefore, it leads us to the conclusion that the Three-Tier data center with maximum numbers of hosts and switches also consumes less energy than large-scale Fat-Tree data center.

Table 5.12: Average total energy consumption by Three-Tier and Fat-Tree data center

Algorithms	Three-Tier data center total energy consumption(Wh)			Fat-Tree data center total energy consumption(Wh)		
	40Hosts-5switches	50Hosts-6switches	100Hosts-11switches	20Hosts-24switches	63Hosts-51switches	128Hosts-80switches
BestFit	52932.75	52932.75	52932.75	59671.95	57799.70	56131.52
RandomSelection	49518.52	52303.47	54625.57	59827.73	62037.17	62767.21
SHC-VNFP	53822.12	54016.92	54063.06	59516.22	60121.08	58800.44
SA-VNFP	23436.77	23587.01	23588.05	29593.43	29930.27	28604.65
Mem-VNFP	23737.76	23974.79	25129.83	28844.09	29375.79	32152.97

5.7 Summary

In this chapter, we evaluated our metaheuristic VNF placement algorithm in a simulation environment with a Three-Tier data center, large-scale Fat-Tree data center, and multiple applications. The results depict that our proposed VNF placement algorithms can reduce the energy consumption of the hosts. In terms of the switches energy consumption BestFit algorithm show the best result comparing other algorithm in both Three-Tier and Fat-Tree data centers. Despite not showing good results when evaluating only the energy consumed by switches, the results of Mem-VNFP and SA-VNFP considering the total energy consumption (switches and hosts) are the best between all algorithms and scenarios. Since the execution time of the algorithms was similar, the metaheuristics employed by Mem-VNFP and SA-VNFP (Memetic Algorithm and Simulated Annealing) proved to be good options when allocating SFCs. The Standard Hill Climbing did not provide good results and can not be considered a good metaheuristic. Moreover, the results demonstrate the energy consumption of the Three-Tier data center is less than the energy consumption of the Fat-Tree data center.

Chapter 6

Conclusions

Increasing energy consumption and the requirements to reduce global Greenhouse Gas emissions are key issues in the ICT sector, therefore, energy efficiency is one of the main aspects of modern communication networks. So, Network Function Virtualization plays a pivotal role to minimize power consumption and GHG emission. Network Function Virtualization is an approach to decouple the dependency between hardware and software. In fact, NFV decouples network functions from their dedicated hardware and implements network functions as software components.

Due to the information technology and telecommunications industry demands, the combination of NFV and SDN are predicted to be the new architecture solutions in order to implement network functions by software.

In this dissertation we presented results obtained with an extension to the CloudSimSDN-NFV simulator, in order to allow the simulation of metaheuristic-based VNF placement algorithms, and with the usage of three new algorithms based on the Standard Hill Climbing method, Simulated Annealing and Memetic Algorithm. Results showed the advantages in using the metaheuristic approaches, with highlights to the Simulated Annealing method.

In this work, we have achieved all of the objectives that we had defined in Chapter 1. First of all, in Chapter 2, we reviewed all the concepts that were important for this work. For instance, the survey of the Network Function Virtualization, Service Function Chaining, energy-efficiency of NFV, the definition of Data Center Network topologies, and also the review of metaheuristic algorithms, that are used on this work, were presented in this chapter.

Then, in Chapter 3, by comparing simulators and testbed, we concluded that the CloudSimSDN-NFV is a proper simulator for this work because it simultaneously supports the simulation of VNF and SFC. Also, we analyze the negative and positive aspects of both virtual machines and containers. Therefore, we noticed that a Virtual machine is more secure than a container. Moreover, our survey shows the CPU utilization illustrates no significant differences between the Virtual machines and the containers. Further, by analyzing some VNFs placement algorithms, we recognized that we need an algorithm to consider the reduction of host power consumption, switch power consumption, and execution time. Therefore, we proposed a set of algorithms based on metaheuristic approaches.

In Chapter 4, we presented our proposed metaheuristic algorithms to VNFs placement in a data center, these algorithms have to be an energy-efficient placement. The first algorithm is based on Hill Climbing, the second algorithm is based on Simulated Annealing, and the third algorithm is based on the Memetic algorithm. However, we altered the proposed algorithms to support the VNFs placement. The results have illustrated in Chapter 5, the algorithms have applied to the Three-Tier and Fat-Tree data centers with different numbers of hosts and switches. In the Three-Tier data centers, the results show that the proposals based on the Simulated Annealing saved the total energy consumption up to 55.44% when placing VNFs in a three-tier data center with 100 hosts and 11 switches. Also, in the Fat-Tree data center, the proposal based on a memetic algorithm reduced the total energy consumption around 51.66% when placing VNFs in a fat tree data center with 20 hosts and 24 switches. Furthermore, the energy consumption of the Three-Tier data center is less than Fat-Tree data center.

Chapter 7

Appendix

Bibliography

- [AAL17] Talal Alharbi, Ahamed Aljuhani e Hang Liu. Holistic ddos mitigation using nfv. Em *Computing and Communication Workshop and Conference (CCWC)*, 2017 *IEEE 7th Annual*, páginas 1–4. IEEE, 2017. [11](#)
- [AC13] Anders Andrae e Peter M Corcoran. Emerging trends in electricity consumption for consumer ict. 2013. [1](#)
- [AGJ⁺13] Vaneet Aggarwal, Vijay Gopalakrishnan, Rittwik Jana, KK Ramakrishnan e Vinay A Vaishampayan. Optimizing cloud resources for delivering iptv services through virtualization. *IEEE Transactions on Multimedia*, 15(4):789–801, 2013. [11](#)
- [AHA⁺16] Jason Anderson, Hongxin Hu, Udit Agarwal, Craig Lowery, Hongda Li e Amy Apon. Performance considerations of network functions virtualization using containers. Em *Computing, Networking and Communications (ICNC)*, 2016 *International Conference on*, páginas 1–7. IEEE, 2016. [24](#)
- [AHGZ16] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani e Taieb Znati. Network function virtualization in 5g. *IEEE Communications Magazine*, 54(4):84–91, 2016. [11](#)
- [ASH19] AMO Abdulmula, Kamaruzzaman Sopian e Lim Chin Haw. Power consumption modeling based on real-time data traffic for balancing power supply and energy demand to develop green telecommunication tower: A case study. *Engineering, Technology & Applied Science Research*, 9(3):4159–4164, 2019. [1](#)
- [Ban97] Rafael E Banchs. Simulated annealing. Relatório técnico, Research progress report, 1997. [32](#)
- [BB15] Michael Till Beck e Juan Felipe Botero. Coordinated allocation of service function chains. Em *Global Communications Conference (GLOBECOM)*, 2015 *IEEE*, páginas 1–6. IEEE, 2015. [xv](#), [3](#), [12](#)
- [BBA10] Rajkumar Buyya, Anton Beloglazov e Jemal Abawajy. Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*, 2010. [2](#), [19](#), [22](#)
- [BBDC10] Raffaele Bolla, Roberto Bruschi, Franco Davoli e Flavio Cucchietti. Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. *IEEE Communications Surveys & Tutorials*, 13(2):223–244, 2010. [11](#)
- [BCAB15] Md Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed e Raouf Boutaba. On Orchestrating Virtual Network Functions. Em *Network and Service Management (CNSM)*, 2015 *11th International Conference on*, páginas 50–56. IEEE, 2015. [3](#)
- [BCR⁺12] Aruna Prem Bianzino, Claude Chaudet, Dario Rossi, Jean-Louis Rougier et al. A Survey of Green Networking Research. *IEEE Communications Surveys & Tutorials*, 14(1):3–20, 2012. [1](#), [2](#)

- [BdSG16] Lucas Bondan, Carlos Raniery Paula dos Santos e Lisandro Zambenedetti Granville. Comparing virtualization solutions for nfv deployment: A network management perspective. Em *2016 IEEE Symposium on Computers and Communication (ISCC)*, páginas 669–674. IEEE, 2016. [24](#)
- [BLBM14] Raffaele Bolla, Chiara Lombardo, Roberto Bruschi e Sergio Mangialardi. DROPv2: Energy Efficiency through Network Function Virtualization. *IEEE Network*, 28(2):26–32, 2014. [21](#)
- [BLS13] Ilhem BoussaïD, Julien Lepagnot e Patrick Siarry. A survey on optimization meta-heuristics. *Information sciences*, 237:82–117, 2013. [13](#)
- [Bri14] ONF Solution Brief. Openflow-enabled sdn and network functions virtualization. *Open Netw. Found*, 2014. [10](#)
- [Bro15] Gabriel Brown. Service chaining in carrier networks. *Heavy Reading*, 2015. [xv](#), [13](#)
- [C⁺13] EINS Consortium et al. Overview of ICT Energy Consumption (D8. 1). *Report FP7-2888021. European Network of Excellence in Internet Science*, 2013. [1](#)
- [CCC⁺16] Franco Callegati, Walter Cerroni, Chiara Contoli, Rossella Cardone, Matteo Nocentini e Antonio Manzalini. Sdn for dynamic nfv deployment. *IEEE Communications Magazine*, 54(10):89–95, 2016. [10](#)
- [CDT⁺] Chunfeng Cui, Hui Deng, Deutsche Telekom, Uwe Michel e Herbert Damker. Network functions virtualisation. [9](#)
- [CFPF] Andrew Chipperfield, Peter Fleming, Hartmut Pohlheim e Carlos Fonseca. Toolbox. [16](#)
- [CGW⁺12] Chien A Chan, André F Gyga, Elaine Wong, Christopher A Leckie, Ampalavanapillai Nirmalathas e Daniel C Kilper. Methodologies for assessing the use-phase power consumption and greenhouse gas emissions of telecommunications network services. *Environmental science & technology*, 47(1):485–492, 2012. [1](#)
- [CLT⁺17] Gary Cook, Jude Lee, Tamina Tsai, Ada Kong, John Deans, Brian Johnson e Elizabeth Jardim. Clicking clean: Who is winning the race to build a green internet? *Greenpeace Inc., Washington, DC*, página 5, 2017. [xv](#), [1](#), [2](#)
- [CN01] Peter M Chen e Brian D Noble. When virtual is better than real [operating system relocation to virtual machines]. Em *Proceedings Eighth Workshop on Hot Topics in Operating Systems*, páginas 133–138. IEEE, 2001. [9](#)
- [CP17] Richard Cziva e Dimitrios P Pezaros. Container network functions: bringing nfv to the network edge. *IEEE Communications Magazine*, 55(6):24–31, 2017. [9](#)
- [Cre81] Robert J. Creasy. The origin of the vm/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490, 1981. [7](#)
- [CSM15] Massimo Condoluci, Fragkiskos Sardis e Toktam Mahmoodi. Softwarization and virtualization in 5g networks for smart cities. Em *International Internet of Things Summit*, páginas 179–186. Springer, 2015. [2](#)
- [CZA⁺16] Jiuyue Cao, Yan Zhang, Wei An, Xin Chen, Yanni Han e Jiyan Sun. Vnf placement in hybrid nfv environment: Modeling and genetic algorithms. Em *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, páginas 769–777. IEEE, 2016. [26](#)

- [EAL17] Vincenzo Eramo, Mostafa Ammar e Francesco Giacinto Lavacca. Migration energy aware reconfigurations of virtual network function instances in nfv architectures. *IEEE Access*, 5:4927–4938, 2017. 2
- [ETS13] GSNFV ETSI. 001:" network functions virtualisation (nfv). *Architectural framework*, 2013. 11
- [Fow] Martin Fowler. xv, 10
- [GB11] Saurabh Kumar Garg e Rajkumar Buyya. Networkcloudsim: Modelling parallel applications in cloud simulations. Em *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, páginas 105–113. IEEE, 2011. 22
- [GG84] Stuart Geman e Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984. 15
- [GHCR17] Aravinthan Gopalasingham, Dalia Georgiana Herculea, Chung Shue Chen e Laurent Roullet. Virtualization of radio access network by virtual machine and docker: Practice and performance analysis. Em *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, páginas 680–685. IEEE, 2017. 25
- [GKZ16] Damien Meadows Yvon Slingenberg Artur Runge-Metzger Stefaan Vergote Jake Werksman Ger Klaassen, Jürgen Lefever e Peter Zapfel. Eu climate policy explained 2020, 2016. 1
- [Glo86] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986. 13
- [GP18] Dewang Gedia e Levi Perigo. Performance evaluation of sdn-vnf in virtual machine and container. Em *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, páginas 1–7. IEEE, 2018. 25
- [HAK⁺15] Sandra Herker, Xueli An, Wolfgang Kiess, Sergio Beker e Andreas Kirstaedter. Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements. Em *Globecom Workshops (GC Wkshps), 2015 IEEE*, páginas 1–7. IEEE, 2015. 6, 7
- [HB16] Juliver Gil Herrera e Juan Felipe Botero. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016. 3
- [HMR⁺15] Brent Hirschman, Pranav Mehta, Kannan Babu Ramia, Ashok Sunder Rajan, Edwin Dylag, Ajaypal Singh e Martin McDonald. High-performance evolved packet core signaling and bearer processing on general-purpose processors. *IEEE Network*, 29(3):6–14, 2015. 10
- [IH09] Teerawat Issariyakul e Ekram Hossain. Introduction to network simulator 2 (ns2). Em *Introduction to network simulator NS2*, páginas 1–18. Springer, 2009. 22
- [JHL⁺18] Shardrom Johnson, Jinwu Han, Yuanchen Liu, Li Chen e Xinlin Wu. Hybrid approach with improved genetic algorithm and simulated annealing for thesis sampling. *Future Internet*, 10(8):71, 2018. 15
- [JP13] Raj Jain e Subharthi Paul. Network Virtualization and Software Defined Networking for Cloud Computing: A Survey. *IEEE Communications Magazine*, 51(11):24–31, 2013. 2

- [KAEC⁺18] Jakub Krzywda, Ahmed Ali-Eldin, Trevor E Carlson, Per-Olov Östberg e Erik Elmroth. Power-performance tradeoffs in data center servers: Dvfs, cpu pinning, horizontal, and vertical scaling. *Future Generation Computer Systems*, 81:114–128, 2018. [11](#)
- [KBK12] Dzmitry Kliazovich, Pascal Bouvry e Samee Ullah Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283, 2012. [22](#)
- [KD13] Keith Kirkpatrick e Bruce Davie. The magazine archive includes every article published in communications of the acm for over the past 50 years. *Communications of the ACM*, 56(9):16–19, 2013. [12](#)
- [KDW05] Il Yong Kim e OL De Weck. Variable chromosome length genetic algorithm for progressive refinement in topology optimization. *Structural and Multidisciplinary Optimization*, 29(6):445–456, 2005. [16](#)
- [KGV83] Scott Kirkpatrick, C Daniel Gelatt e Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983. [13](#), [14](#), [15](#)
- [Kol06] Kirill Kolyshkin. Virtualization in linux. *White paper, OpenVZ*, 3:39, 2006. [8](#)
- [KRV⁺15] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky e Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015. [3](#)
- [Krz19] Jakub Krzywda. *May the power be with you: managing power-performance tradeoffs in cloud data centers*. Tese de Doutorado, Umeå University, 2019. [11](#)
- [LC15] Yong Li e Min Chen. Software-defined network function virtualization: A survey. *IEEE Access*, 3:2542–2553, 2015. [xv](#), [10](#)
- [LFLK17] Aris Leivadeas, Matthias Falkner, Ioannis Lambadaris e George Kesidis. Optimal virtualized network function allocation for an sdn enabled cloud. *Computer Standards & Interfaces*, 54:266–278, 2017. [11](#)
- [LJYH13] Xiang Li, Xiaohong Jiang, Kejiang Ye e Peng Huang. Dartcsim+: Enhanced cloudsims with the power and network models integrated. Em *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, páginas 644–651. IEEE, 2013. [22](#)
- [LLZ⁺15] Jiaqiang Liu, Yong Li, Ying Zhang, Li Su e Depeng Jin. Improve service chaining performance with optimized middlebox placement. *IEEE Transactions on Services Computing*, 10(4):560–573, 2015. [26](#)
- [M⁺89] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989. [16](#)
- [MA15] Antonio Marotta e Stefano Avallone. A simulated annealing based approach for power efficient virtual machines consolidation. Em *2015 IEEE 8th international conference on cloud computing*, páginas 445–452. IEEE, 2015. [26](#)
- [MAB⁺08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker e Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008. [12](#)
- [MC03] Pablo Moscato e Carlos Cotta. A gentle introduction to memetic algorithms. Em *Handbook of metaheuristics*, páginas 105–144. Springer, 2003. [13](#)

- [MG⁺11] Peter Mell, Tim Grance et al. The nist definition of cloud computing. 2011. 5
- [MGS⁺18] M Muntean, D Guizzardi, E Schaaf, M Crippa, E Solazzo, J Olivier e E Vignati. Fossil co2 emissions of all world countries, 2018. 1
- [MGZ16] Marouen Mechtri, Chaima Ghribi e Djamal Zeghlache. Vnf placement and chaining in distributed cloud. Em *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, páginas 376–383. IEEE, 2016. 26
- [MKK15] Roberto Morabito, Jimmy Kjällman e Miika Komu. Hypervisors vs. lightweight virtualization: a performance comparison. Em *2015 IEEE International Conference on Cloud Engineering*, páginas 386–393. IEEE, 2015. 9
- [MRR⁺53] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller e Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. 14
- [MSG⁺16] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck e Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016. 11
- [NA98] Yaghout Nourani e Bjarne Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41):8373, 1998. 14
- [Nil96] Nils J Nilsson. Stuart russell and peter norvig, artificial intelligence: A modern approach. *Artificial intelligence*, 82(1-2):369–380, 1996. 13
- [ONF12] Software-defined networking: The new norm for networks, April, 2012. xv, 12
- [PJ17] Jasmeet Singh Sethi Mats Arvedson Ritva Svenningsson Per Lindberg Kati Öhman Patrik Hedlund Peter Jonsson, Stephen Carson. Ericsson mobility report, 2017. xv, 2
- [PL97] Riccardo Poli e William B Langdon. A new schema theorem for genetic programming with one-point crossover and point mutation. *COGNITIVE SCIENCE RESEARCH PAPERS-UNIVERSITY OF BIRMINGHAM CSRP*, 1997. 17
- [PMWV09] Steven Pelley, David Meisner, Thomas F Wenisch e James W VanGilder. Understanding and abstracting total data center power. Em *Workshop on Energy-Efficient Design*, volume 11, 2009. 30
- [PWB03] Ian Pratt, Andrew Warfield, P Barham e R Neugebauer. Xen and the art of virtualization. Em *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, páginas 113–1118, 2003. 9
- [RLRL17] Windhya Rankothge, Franck Le, Alessandra Russo e Jorge Lobo. Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms. *IEEE Transactions on Network and Service Management*, 14(2):343–356, 2017. 25, 26
- [RRG13] Roberto Riggio, Tinku Rasheed e Fabrizio Granelli. Empower: A testbed for network function virtualization research and experimentation. Em *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, páginas 1–5. IEEE, 2013. 21
- [SB] Jungmin Son e Rajkumar Buyya. Cloudsimsdn-nfv: Modeling and simulation of network function virtualization and service function chaining in edge computing environments. 19, 37
- [SDC⁺14] Joao Soares, Miguel Dias, Jorge Carapinha, Bruno Parreira e Susana Sargento. Cloud4nfv: A platform for virtual network functions. Em *Cloud Networking (Cloud-Net), 2014 IEEE 3rd International Conference on*, páginas 288–293. IEEE, 2014. 3

- [SDC⁺15] Jungmin Son, Amir Vahid Dastjerdi, Rodrigo N Calheiros, Xiaohui Ji, Young Yoon e Rajkumar Buyya. Cloudsimsdn: Modeling and simulation of software-defined cloud data centers. Em *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, páginas 475–484. IEEE, 2015. [20](#)
- [SGP⁺15] João Soares, Carlos Gonçalves, Bruno Parreira, Paulo Tavares, Jorge Carapinha, João Paulo Barraca, Rui L Aguiar e Susana Sargento. Toward a telco cloud environment for service functions. *IEEE Communications Magazine*, 53(2):98–106, 2015. [13](#)
- [SR96] Patrick D Surry e Nicholas J Radcliffe. Inoculation to initialise evolutionary search. Em *AISB Workshop on Evolutionary Computing*, páginas 269–285. Springer, 1996. [16](#)
- [STV15] Marco Savi, Massimo Tornatore e Giacomo Verticale. Impact of Processing Costs on Service Chain Placement in Network Functions Virtualization. Em *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, páginas 191–197. IEEE, 2015. [3](#)
- [Tov85] Craig A Tovey. Hill climbing with multiple local optima. *SIAM Journal on Algebraic Discrete Methods*, 6(3):384–393, 1985. [13](#)
- [Tow15] K Townsend. Containers: The pros and the cons of these vm alternatives. *TechRepublic, February*, 2015. [9](#)
- [TSCB19] Adel Nadjaran Toosi, Jungmin Son, Qinghua Chi e Rajkumar Buyya. Elasticfc: Auto-scaling techniques for elastic service function chaining in network functions virtualization-based clouds. *Journal of Systems and Software*, 2019. [42](#)
- [VG15] Amir Varasteh e Maziar Goudarzi. Server consolidation techniques in virtualized data centers: A survey. *IEEE Systems Journal*, 11(2):772–783, 2015. [11](#)
- [VHLL⁺14] Ward Van Heddeghem, Sofie Lambert, Bart Lannoo, Didier Colle, Mario Pickavet e Piet Demeester. Trends in Worldwide ICT Electricity Consumption from 2007 to 2012. *Computer Communications*, 50:64–76, 2014. [1](#)
- [VLA87] Peter JM Van Laarhoven e Emile HL Aarts. Simulated annealing. Em *Simulated annealing: Theory and applications*, páginas 7–15. Springer, 1987. [14](#)
- [VN15] SJ Vaughan-Nichols. Containers vs. virtual machines: How to tell which is the right choice for your enterprise, 2015. [9](#)
- [WC13] Feng-Sheng Wang e Li-Hsunan Chen. *Simulated annealing*. Springer, 2013. [32](#)
- [WYW⁺12] Xiaodong Wang, Yanjun Yao, Xiaorui Wang, Kefa Lu e Qing Cao. Carpo: Correlation-aware power optimization in data center networks. Em *2012 Proceedings IEEE INFOCOM*, páginas 1125–1133. IEEE, 2012. [30](#)
- [XLWX16] Zhifeng Xu, Fangming Liu, Tao Wang e Hong Xu. Demystifying the Energy Efficiency of Network Function Virtualization. *2016 IEEE/ACM 24th International Symposium on Quality of Service, IWQoS 2016*, páginas 1–10, 2016. [2](#)
- [YXKZ15] Ruozhou Yu, Guoliang Xue, Vishnu Teja Kilari e Xiang Zhang. Network Function Virtualization in the Multi-tenant Cloud. *IEEE Network*, 29(3):42–47, 2015. [2](#)
- [YXLT15] Wutong Yang, Minxian Xu, Guozhong Li e Wenhong Tian. Cloudsimnfv: modeling and simulation of energy-efficient nfv in cloud data centers. *arXiv preprint arXiv:1509.05875*, 2015. [19](#)
- [ZCB10] Qi Zhang, Lu Cheng e Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010. [5](#)