Academic year : 2023/2024

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF SOUSSE

HIGHER INSTITUTE OF COMPUTER SCIENCE AND COMMUNICATION TECHNIQUES

THESIS

Presented for the degree of

DOCTOR OF COMPUTER SCIENCES

Virtual Network Function Placement in Cloud Infrastructure

By
Wided KHEMILI

Supported on January 10, 2024 in front of the committee composed of :

President :	Prof. Ouajdi KORBAA	University of Sousse
Reporter :	Prof. Ali DOUIK	University of Sousse
Reporter :	Prof. Semia BOUZEFRANE	CNAM Paris
Examiner:	Prof. Lotfi BEN ROMDHANE	University of Sousse
Examiner:	M.C Saadi BOUDJIT	University of Sorbonne Paris Nord
Thesis director:	Prof. Mohamed Nazih OMRI	University of Sousse
Thesis director :	Prof. Ken CHEN	University of Sorbonne Paris Nord
Co-supervisor:	M.C Mohand Yazid SAIDI	University of Sorbonne Paris Nord

Guest as Co-supervisor: M.A Jalel Eddine Hajlaoui Université of Sousse

Abstract

Network Function Virtualization (NFV) is a technological innovation which aims at making the network more adaptable, more controllable and more profitable. This technology reduces CAPEX and OPEX costs by sharing the physical resources. NFV allows simultaneous execution of a variety of network functions over a shared network. To take the set advantage of NFV technology, researchers are equired to resolve and meet the challenges imposed by this technology such as placement, orchestration, management and consolidation. Despite the immense usefulness provided by NFV, several technical problems, which we will be able to identify and deal with in this work, should be resolved. Consolidating VNF can be considered as one of the main challenges for effective exploitation and management of VNF. The consolidation involves allocating some virtual resources in the virtual or physical to balancing resource exploitation and minimizing energy consumption. In this thesis, we present a comparative and comprehensive study dealing with the problem of placement and consolidation of VNFs. We presented a new classification of the VNF placement and consolidation approaches proposed in the literature to donate academics and researchers more experiences on how to consolidate and place elements in NFV environment so that energy and resources utilization are optimized. In the light of this study, we proposed new adapted approaches to overcome the insufficiencies of the approaches in literature.

Virtual Network Function (VNF) is the essential softwarization engine that extracts network functions from dedicated hardware devices to run them as services running on virtual machines or containers. Network services are created on Virtual Machines (VMs) may be connected or organized in a single enclosure to take advantage of all available resources within it. This flexibility lets physical and virtual resources to be used in a way it guarantees efficient management and control reduce energy consumption, balance the load and improve resource utilization, and finally minimize the cost and latency. In order to consolidate a maximum number of VNFs into the smallest number of virtual machines (VMs) by estimating the association relationship to a confidence measure in the context of possibility theory, we proposed a novel approach called Fuzzy-FCA for VNF placement based on Formal Concept Analysis (FCA) and fuzzy logic in a mixed environment that

combines cloud and Multi Access Edge Computing (MEC) architecture. Our approach combines Formal Concept Analysis and fuzzy logic to ensure the distribution of compute resources to the end user in order to reduce end-to-end latency. Simulation experiments conduced during this thesis showed have shown the efficiency of our proposal especially in reducing latency and cost, minimizing energy consumption and balancing resource usage over the network. Besides, Fuzzy-FCA outperforms the newer approaches like the MultiSwarm algorithm extensively studied in the literature. To go deeper, we addressed the issue of VNF placement while considering the chaining.

In fact, the VNF placement problem has become surrounded by other constraints that enshrine the order of VNFs according to their chain type. This restriction requires a method of extracting the best and shortest traffic from a chain of VNFs considering the replication of the same type of VNF. In this context, we proposed an algorithm that aims at finding the most appropriate location for each VNF composing the appropriate service and the best routing to confront the traffic congestion (taking into account the bandwidth) and minimize the transmission end to end delay of the request service (chain of VNFs) by respecting the resources constraints (CPU, memory and storage) in a multi-instance VNF environment. To address the issue involved by the VNF placement and chaining, we propose an ILP modeling and a novel model of classification named Parallel Bi-state module deep reinforcement learning (PBDRL) algorithm. Our proposal consists in two modules, the first is MDP(Markov Decision Process) that is used to capture service chain state transition, and the second is LSTM (Long Short-Term Memory) that enable to detect the long-term historical of service chain and its transitions. More specifically, our model aims at extracting the characteristics of VNF environment through MDP and LSTM simultaneously which takes into account the dynamicity and history of NFV environment. Thus, experimental results have shown the efficiency of our proposal which outperforms both the NFV deep approach and LSTM algorithm. As an extension, we proposed a hybrid algorithm which combines the two previous algorithms ; Fuzzy-FCA and PBDRL. This algorithm addresses the issue of VNF location taking into account the robustness and the cost of migration. The goal is to limit the cost and the delay through respecting the quality of service and the service agreements. The simulation results demonstrate the efficiency of our algorithm to minimize the migration cost and the operation cost, as well as the importance of combined ILP and PBDRL algorithms to minimize the delay and the reallocation cost.

Résumé

La virtualisation des fonctions réseau (NFV) est une des dernières innovations technologiques introduite dans le but de rendre le réseau plus flexible, plus contrôlable et plus rentable. Cette technologie réduit le CAPEX et OPEX en partageant les ressources physiques ou virtuelles de réseaux. Ainsi NFV permet l'exécution simultanée de diverses fonctions réseau sur un réseau partagé. Pour améliorer l'utilisation des ressources avec NFV, les cherchercheurs s'y sont interessés de prés, notamment pour relever les défis liés à cette technologie comme le placement, l'orchestration, la gestion et la consolidation de VNF (Virtual Network Function). Cette dernière peut être considérée comme l'un des principaux enjeux pour l'exploitation et la bonne gestion de VNF. Cette consolidation consiste à allouer certaines ressources virtuelles dans la même machine physique ou entre deux ressources virtuelles tout en exploitant diverses métriques qui peuvent correspondre à l'équilibrage de l'utilisation (exploitation) des ressources ou la minimisation de la consommation d'énérgie. Dans ce travail, nous ferons et présenterons une étude comparative et compréhensive des methodes de consolidation afin de fournir, procurer plus d'expériences sur la manière de consolider les ressources et les critères à prendre en compte. Les fonctions de réseau virtuel (VNF) est la base derrière la virtualisation des réseaux et NFV. Initialement rendue par du materiel dediés, elles sont rendue avec l'avénement de NFV sous forme logicielle tournant sur du matériel générique moins cher qui sépare les fonctions réseau et leurs periphériques matériels dédiés, tel que les roteurs, les pare-feu et les équilibreurs de charge, pour héberger leurs services sur des machines virtuelles. La VNF est responsable des services réseau qui s'exécutent sur des machines virtuelles et peuvent connecter chacune d'entre elles seules ou s'organiser dans un boîtier unique pour utiliser toutes les ressources disponibles dans ce boîtier. Cette flexibilité permet d'utiliser les ressources physiques et virtuelles de manière qui assure le contrôle de la consommation d'énergie, l'équilibre dans l'utilisation des ressources et la minimisation des coûts et de la latence. Afin de consolider les groupes VNF en un nombre minimum de machines virtuelles (VM) avec estimation de la relation d'association à une mesure de confiance dans le contexte de la théorie des possibilités, nous proposons une nouvelle approche Fuzzy-FCA pour le placement VNF basée sur l'analyse de concept formelle.

(FCA) et logique floue dans un environnement mixte basé sur des centres de données cloud et Architecture Edge Computing (MEC) à accès multiples. Ainsi, l'inclusion de cette architecture dans l'environnement cloud assure la distribution des ressources de calcul à l'utilisateur final afin de réduire la latence de bout en bout. Pour évaluer et montrer l'efficacité de notre solution, nous l'avons comparée à l'un des meilleurs algorithmes étudiés dans la littérature, à savoir l'algorithme MultiSwarm. Les résultats de la série d'expériences réalisées montrent la faisabilité et l'efficacité de notre algorithme. En effet, nos résultats d'expérience confirment la capacité de notre algorithme à maximiser et équilibrer l'utilisation des ressources, à minimiser la latence et le coût de la consommation énergétique. Pour aller plus loin, nous traitons le problème de placement VNF en considérant les caractéristiques de chaînage VNF. Ainsi, le problème initial de placement des VNF est desormais entouré de nouvelles contraintes qui consacrent l'ordre des VNFs ou de leur chainage. La prise en compte de ces nouvelles contraintes requiert la détermination d'un routage efficace minimisant les chemins tout en prenant en compte la replication de VNF. Dans ce cadre, nous proposons un algorithme visant à trouver la localisation la plus appropriée pour chaque VNF composant le service réseau, permettant de pallier la congestion et de minimiser le délai de transmission de la requête de bout en bout (chaîne de VNF) tout en verifiant les contraintes liés aux ressources demandées (CPU, mémoire et stockage) dans un environnement VNF multi-instance. Pour résoudre ces problèmes, nous proposons une modélisation ILP et un nouveau modèle de classification nommé algorithme d'apprentissage par renforcement profond du module bi-état parallèle (PBDRL). Cet algorithme est basé sur deux modules, le premier est MDP (Markov Decision Process) qui est utilisé pour capturer la transition d'état de la chaîne de service, alors que le second est LSTM (Long Short-Term Memory) qui permet de détecter l'historique à long terme de la chaîne de service et ses transitions. Notre modèle vise à extraire les caractéristiques de l'environnement VNF via MDP et LSTM en tenant compte simultanément de la dynamique et de l'historique de l'environnement NFV. Comme extension, nous avons proposé une algorithme hybride qui combine les deux precedents algorithme de Fuzzy-FCA et PBDRL. Ce nouvel algorithme aborde le problème de placement en tenant en consideration la migration et le cout de migration. L'objectif est de minimiser le cout et le delai en respectant les contraintes liées à la qualité de service et Service Agreements. Nos experimentations et simulations ont montré l'efficacité de notre algorithme, notamment pour optimiser le cout de migration et le cout d'operation, ainsi que l'importance des algorithmes combinés ILP et PBDRL pour réduire les délais et les coûts de réallocation.

Contents

List of Figures

List of Tables

List of Algorithms

List of Important Abbreviations

Acronym	Meaning
AI	Artificial Intelligence
VM	Virtual Machine
NFV	Network Function Virtualization
CPU	Central Processing Unit
SDN	Software Defined Network
SFC	Service Function Chain
DC	Data Center
MILP	Mixed Integer Linear Programming
ETSI	European Telecommunications Standards Institute
FF	First Fit
IaaS	Infrastructure-as-a-service
IBM	International Business Machines
IDE	Integrated Development Environment
I/O	Input/Output
IT	Information Technology
KVM	Kernel-based Virtual Machine
MDP	Markov Decision Process
LSTM	Long Short-Term Memoy
FCA	Formal Concept Analysis
NS	Network Service
SLA	Service Level Agreement
OPEX	Operational Expenditure
OS	Operating System
IoT	Internet of Things

PM	Physical Machines
RAM	Random Access Memory
RL	Reinforcement Learning
RQ	Research Question
vCPU	Virtual CPU
VM	Virtual Machine
VMM	Virtual Machine Monitor

Introduction

Context and issue

Cloud computing represented a big leap in technological evolution. Where virtualization is considered the main driver of cloud computing, which serves to separate physical resources to meet the demands of a gigantic data center. In fact, the growing demand for the use on telecommunication networks represent the first reason for the advent of network virtualization. Network Function Virtualization (NFV) creates virtual copies of network services (firewall, router, DPI (Deep Packet Inspection)) and runs them on virtual machines. It allows network services to be provided to operators on demand without the need for additional hardware. In addition, NFV works to facilitate the execution and distribution of virtualized network functions on different servers or move and migrate them dynamically from server to other one according to the request, ie anywhere on the network [5]. In another sense, NFV allows to create hybrid constructor where the network functions and resources can coexist and consolidated from one host machine to another as needed [6]. This characteristics provides a suitable environment of the consolidation. Generally, the consolidation consists of migrating functions and resources from their original location to another location and putting emptied locations in deactivated mode. This aims to reduce the number of active physical devices and therefore the number of virtual devices, leading to the improvement of the resource exploitation and to energy consumption. The majority of studies dealing with the issue of energy minimization in virtual environments is treated in connection with the problem of consolidation [7] [8]. Moreover, [9] tackles the problem of consolidation under interference constraint. We notice that the dimensions or the members of consolidation are different from one approach to another. In [10], the consolidation is performed between the VM and the server, in [7] [8] between VM and tasks, in [9] between requests and

offers, in [3] between VNFs and servers, in [11] between VNFs and VMs, in [12] between the software licenses. There are also several proposed approaches which are interested this problem (VCMM) [3], heuristics [13] [14], genetic algorithms [15] [16], "Gossiping" [17], etc. Some approaches do not indicate the use and exploiting of resources in multi-dimensionality by tackling only one or some resources. Obviously, there are some inconsistencies with consolidation, and some approaches fail to address the issues of interference, machine overload, minimizing bandwidth, and so on. A set of approaches are concerned with these contradictions, such as reduced bandwidth usage, energy savings, migration costs [3], and interference [7]. Due to consolidation which reduces the physical and virtual machines number, the flow of communication between virtual machines will inevitably be reduced. Some works do not consider the importance of this factor which also requires a large amount of energy. Some other works consider the consolidation as migration that they model formulate by counting their cost. As an extension of the VNF placement problem, it cannot solve the VNF placement problem without taking into account the chaining characteristics of the VNFs. A flow of VNFs is routed in chains according to a predefined order, which constitutes a network service chain called SFC [6] [18]. Each service chain consists in a sequence of VNF which is organized in order on demand and transmitted by packets called SFC request in multi-instance environment. The main challenge is to find the appropriate placement for each VNF when constituting SFC considering the multiplicity of VNF instance and the multiplicity of path which can satisfy the predefined order for the SFC query. Furthermore, the flow on the router and switch is a challenge at the level of the control unit (SDN) [3] [19]. Thus, the growing of traffic requires a high-performance routing strategy to minimize the routing computation time. The service request (or the service chain) is also characterized by an end-to-end delay which refers to the total time required for the traffic coming from a given source to reach the destination.

Lately, a multiple research works has addressed the problem of VNF selection and chaining. In [8], Coa et al developed a log-competitive online COATS algorithm to direct traffic in an SDN network. Their algorithm aims to control the traffic by considering the time of arrival and departure times of traffic. However, the proposed algorithm does not address the bandwidth availability issue which considers the VNF instance multiplicity. In [20] a new deep learning based strategy has been proposed to solve the VNF selection and chaining problem to minimize the end-to-end delay and get the best SFC routing

path. Note that, these two studies [20] [8] focus on the role of SDN to control traffic in multi-instance VNF network. [21] focused on improving shared resource utilization of edge servers and physical links within latency bounds. To distribute VNF chains efficiently, the authors of [21] proposed two algorithms: constrained depth-first search (CDFSA) algorithm for path selection and a path-based greedy algorithm (PGA). In their approach, the VNF assignment is done by reusing as many VNFs as possible by formulating and saving the problem with MILP.

Motivation and Goals

Based on recent research works dealing with the management of virtual and physical resources in the cloud, the consolidation and placement of resources taking into account the chained structure of resources represents a recent challenge. This challenge requires efficient and adapted strategies for the problem of consolidation and placement according to the type and characteristics of the resources. In this context, we deal with the placement and consolidation problem at two levels; the first at the level of VNF itself and the second at the level of the VNF chain. Clustering and classification is a completely clever method that the FCA [22] and Fuzzy-FCA are based on to limit the number of active virtual machines by consolidating the maximal number of VNFs in VM. Initially, VNFs are classified and organized according to a formal context with network service (NS) to extract the concept that identifies the best placement of VNFs by applying the different FCA rules. In the second process of Fuzzy-FCA, candidate VNF harvested from the FCA consolidates into the most suitable VM based on the degree of confidence [23]specified by a threshold value between 0 and 1 to avoid data uncertainty. We recall that Fuzzy-FCA [24] is based on the confidence threshold to determine the similarity between two concepts and removes one to avoid repetition and minimize active virtual machines. In the last step, VNF (in Fuzzy-FCA processes) is placed in adequate VM based on swarm intelligence method to ensure balance and resource exploitation maximization. This method can detect the state of the hosting machine by an internal stimilus to calculate the equilibrium rate. The inclusion of the MEC standard in cloud provides cloud computing capabilities and compute resources to the end user at the edge. Also, MEC allows computing resources to be distributed over distributed cloud data centers, which ensures end-to-end latency minimization. All these factors

collectively take part in accomplishing the maximization in exploitation and balance of resources distribution, minimizing latency, lowering the cost of resources and optimizing energy consumption. However, considering the chaining criterion builds constraints and devotes consolidation and placement of precise VNFs according to the existing routing network. To respond to these conditions and constraints, we propose a new approach to place and chain VNFs by considering the multiplicity of VNF instances and the networks congestion. Our approach is based on ILP for the mathematical formulation of problem and deep reinforcement learning to estimate the adequate classification of the VNFs in chain according to the request.

Research Questions

The foremost research questions that outline our work are:

- To what extent the putting VNFs in the correct VM enables to decrease the cost, latency and energy in cloud data center?
- How does Fuzzy FCA approach can contribute to place VNFs in the minimum number of VMs so that the energy consumption of active VMs is minimized?
- How much the new approach of deep reinforcement learning can improve the first experimental results concerning the problem of placement and consolidation considering the chaining?
- How does our approach communicates with small scale and big scale data centers and what are the effects on cost, latency, resource exploitation balance and power consumption?

Research Hypotheses and Thesis Contributions

Research Hypotheses

This work aims to find the best management and orchestration of resources so that it will keep away from the waste of unused resources and energy. These objectives based on the best placement strategies and consolidations of VNFs to decrease the

quantity of active VMs, maximization and balance in exploitation of resources, minimize the cost, reduce latency and energy consumption. Thus, we can use the candidate VNFs to estimate the best placement by considering the predefined VNF chaining and VNF instance multiplicity to minimize throughput and latency.

Thesis Contributions

The foremost contributions of our thesis concerning the problem posed above are as follows:

- Presentation of a state of the art concerning placement and consolidation in the cloud environment to enhance the consumption of resources and energy.
- Defining the general context for the hassle of consolidation and placement of VNFs within side the cloud environment.
- Proposing a new approach of consolidation and placement of VNFs which combines between FCA and Fuzzy FCA algorithm. FCA consolidates the VNF into suitable network service (NS) to extract candidate VNFs. And the fuzzy FCA algorithm is used to consolidate candidate VNFs into the most suitable VMs.
- Proposing a complementary approach to the previous one taking into account the multiplicity of VNF instances and chaining based on deep reinforcement learning named PBDRL.
- Hybridization of the two previous approaches of Fuzzy-FCA and PBDRL to optimize the results of placement and chaining.
- Implement our proposed approach to show their effectiveness and measure its performance by comparing it with other approaches in the literature. Presenting the effect of our solution in minimizing the cost of resources for operators, minimizing latency and energy consumption, and improving throughput in the cloud and MEC environment.

Implemented Methodology

In this part, we present our research methodology illustrated in Fig 2 to answer the research questions (RQs) evoked previously. At first, the model is defined according to multi-objectives; Minimize cost, balance resource usage, reduce latency and power consumption. Considering chaining and network equipment, we add the following objectives: throughput optimization and end-to-end delay minimization. Secondly, the consolidation and placement is carried out as follows:

1. Placement of VNF: it is the migration of VNF from one VM to another more adequate in order to minimize the number of active VMs.
2. Consolidation and placement of VNF: it is the consolidation of VNF in VM according to their capacity.
3. Consolidation and placement of VNFs in chain (SFC): it is the placement of VNFs in VM taking into account a pre-defined order of a chain VNFs (SFC).
4. Migration and placement of VNF : it is the process of moving a specific VNF from one framework to another one considering the technical requirements.

In what follows, we evaluate the efficiency of our approach by comparing its performance results with other approaches in the literature for both large-scale data and unbalanced resource distribution.

Thesis Outline

This thesis is divided into five chapters including an introduction and general conclusion as shown in figure 1. In the introduction, we present the general background and the context of our thesis as well as the motivation and our contributions under research questions. The first chapter is the state of the art which presents NFV approach to identify the main characteristics and features, show its relations with other technologies, presents in details and explain the problem of consolidation in cloud, as well as a list of approaches based on (FCA). The rest is the state of the art for some approaches with discussion. In Chapter 2, we first formulate the VNF consolidation and placement problem and then present our proposed solution which is based on a combination of two algorithms; FCA algorithm and

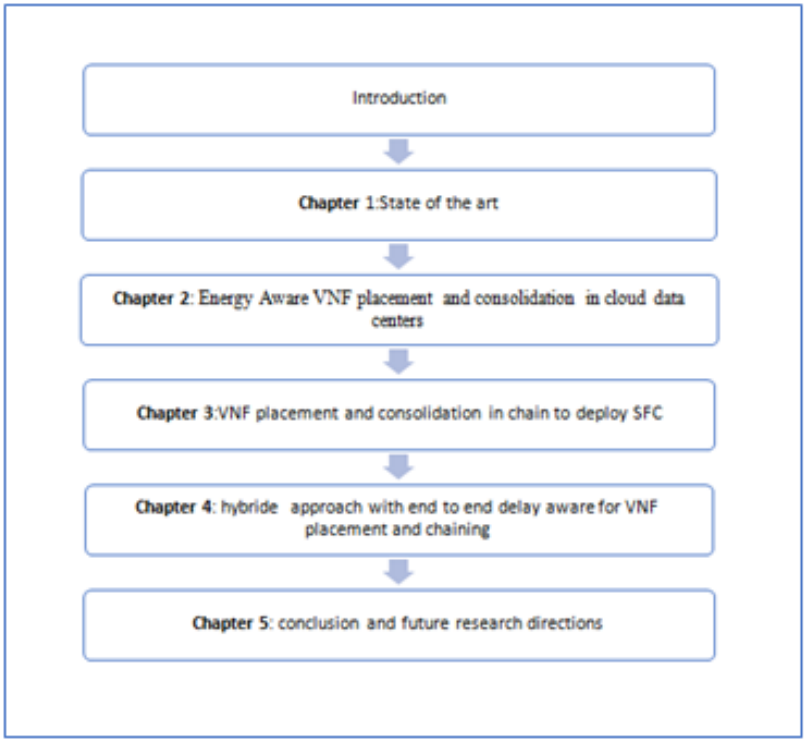

Figure 1: Plan of the thesis.

Fuzzy FCA. The chapter ends by giving some experimental results measuring the performance of our solution by comparing it with Multiswarm algorithm. Considering VNF instance multiplicity and VNF chaining, Chapter 3 presents in details our new deep reinforcement learning based approach named PBDRL developed to consolidate VNFs during each SFC request accommodation and gives experimental results. In chapter 4, we present a hybrid approach which combines the two approaches mentioned in the previous chapters. This hybridization aims at improving the performance and meet the requirements of user and supplier. Finally, in chapter 5, the general conclusion which summarizes our main contributions and results, as well as a general evaluation under a critical and objective scale is given.

Chapter 1

State of the Art

1.1 Introduction

This chapter briefly introduces the main characteristics and aspects of NFV as a virtualization technology. Next, we will identify the relationships of NFV with other technologies and give a detailed explanation for the VNF consolidation problem in cloud data centers. Then, we will present a specific classification of FCA based approaches. In what follows, we show the adaptation of the consolidation technique in different technological environments. We will also present a critical analysis for some consolidation approaches studied with a summary of the major findings and challenges. Finally, we conclude with a summary of VNF placement and consolidation approaches.

1.2 Main characteristics and features of NFV

In this section, we highlight the NFV concept by presenting its relevant criteria as architectural basis. Next, we present and study the issue of consolidation in a cloud data centers.

1.2.1 Virtualization of network functions

NFV makes it possible to separate the software applications of the network functions from resources, storage, and computing [25] [21]. It also allows to separate network functions like firewalls and packet detection inspectors (DPI) and put

them in virtual machines. On the other hand, the European Telecommunications Standards Institute (ETSI) shows that NFV is opposite of network function, closely related to the infrastructure in which it operates. This technology is achieved by consolidating different types of VNF into standard material(servers, devices of storage, etc.) that can be inserted in data centers, network nodes and near or end-user [26]. NFV therefore makes it possible to reduce dependence on equipment, scaling and to have very complex and personalized networks at low cost. NFV has an architecture divided into two orchestrated layers and a MANO management layer [25] [27] as follow:

- The Infrastructure layer: consists of physical infrastructure and the necessary software to run the virtual machines. These infrastructures can be of all kinds: points of presence (servers, storage), routers or switches.
- The Virtual Layer: contains the virtualized functions that run inside the virtual machines.
- The Management layer (MANO): this layer acts a bit like a control plan. It is responsible for establishing connections between virtual functions and managing resources at the infrastructure layer. It is responsible for establishing connections between virtual functions and managing resources at the infrastructure layer.

In [28], NFV is made up of the following elements: infrastructure of NFV (NFVI), set of VNF management and orchestration blocks. NFVI is distributed in groups form of physical nodes that make up the network. Each node is a location that can host one or more network functions. Chaining a group of VNFs constitutes what is called SFC (Service Function Chain) which provides a certain service. NFV management and orchestration layer allows to manage and orchestrate different VNF simultaneously by providing a set of common functions such as run virtualization, migration, update and integrity.
In [1], the NFV architecture comprises three components: the services layer, the NFVI layer, and the NFV Management and Orchestration layer (NFV MANO), as shown in Figure 1.
-Services Layer: is made up of a cluster of VNFs which can be installed into one or more virtual machines. These virtual machines are placed in the operating system

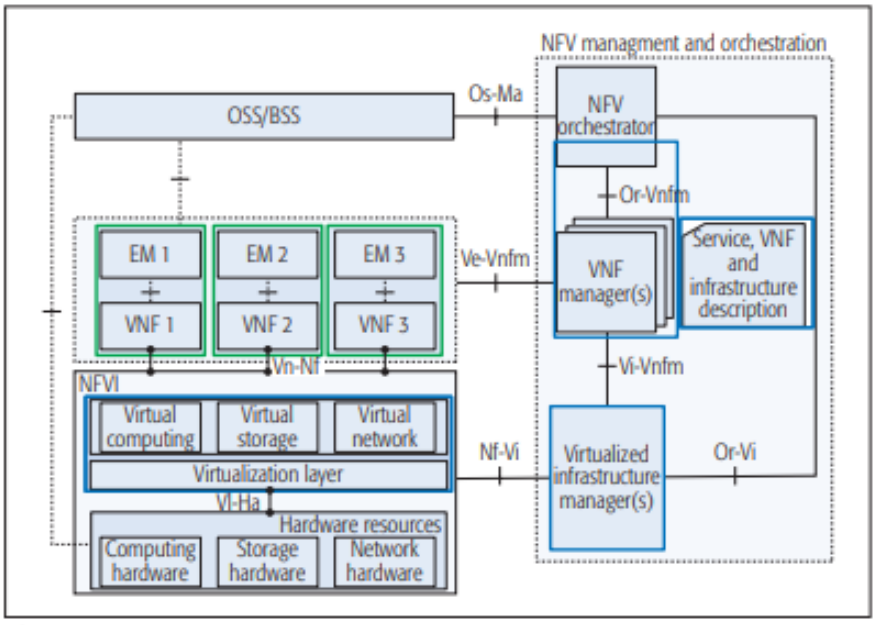

Figure 1.1: NFV-ETSI architecture [1].

or directly on the hardware. It is controlled by hypervisors [18] or virtual machine monitors. An Element Management System (EMS) is typically the VNF administration manager responsible for creation, configuration, security, performance and monitoring. It also gives basic information needed for Operational Support System (OSS) in a TSP environment. Thus, the OSS operates with the Business Support System (BSS), devote the general management system which helps suppliers to control and manage multiple end-to-end telecommunications services.

-NFVI Layer: the NFV infrastructure consists of the hardware and software that builds an environment of NFV. NFVI operates to network connectivity between sites, for example between data centers, hybrid, public or private clouds.

-NFV-MANO layer: called Management and Orchestration of NFV, it is divided into an coordinator of NFVI resources and two managers, one for VNF management (VNF manager) and the other for infrastructure management (virtualized infrastructure managers). It also has databases used to archive modeled information and data that determines both the distribution and the life cycle characteristics of functions, services and resources. Between the different elements of the NFV-MANO, the interfaces that can be employed for communications is defined by framework. Also, to operate of VNF and functions running on traditional equipment, it is required an orchestration with traditional network management systems [6] [29].

1.2.2 Service Function Chain (SFC)

A service function chain is a set of network service functions placed in a predefined order. A network service function is a closed box responsible for a specific task on the network. These functions are logical and can, with the help of NFV techniques, be assigned to virtual machines or directly to physical equipment.

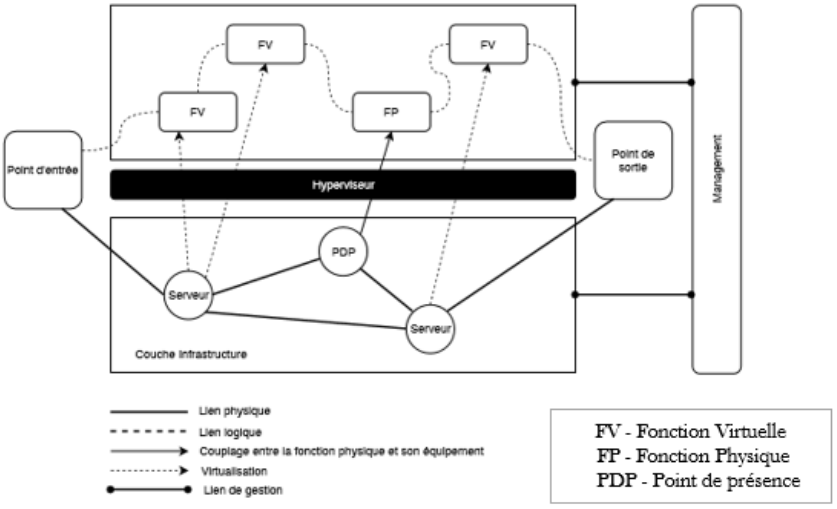

Figure 1.2: Chaine de fonction dans l'architecture du NFV [2]

Figure 1.2 shows the integration of a function chain into the NFV architecture. The function chain is formed by logical links. We see that this chain has an entry point and an exit point. Virtual functions are assigned to servers and/or network points of presence. In this figure, we clearly see the link between virtualized network functions and SFCs. To make this work in reality, it is necessary to define an architecture. This definition must take into account a certain number of problems:

1.2.3 Isolation, virtualization and cloud

Virtualization and isolation are two technologies that contribute to create clouds. Indeed, Virtualization is a technology used to generate virtual representations of physical resources (servers, storage, networks, and other physical machines).

Also, it isolates physical machine resources and distributes them appropriately using a hypervisor [20] that is installed on the physical hardware. It also allows to share the same resources providing the appropriate environment to multiple application so that they can run on more than one platforms. Hypervisors divide in two type; hypervisors of the first type run directly on the receiving machine with an additional software layer removed between the virtual machines and the primary host device. Hypervisor of the second type essentially works mainly in the form of applications, to provide some comfort because the host machine does not need to be specially configured to create virtual machines. Whereas clouds are environments that separate, aggregate, and share scalable resources on a network. Concretely, virtualization is a technology that dematerializes physical resources while the cloud is a large and rich environment that enshrines portability. In the traditional architecture, for each physical machine, a single operating server (OS) is installed natively for more stability and uniformity [30]. However, the virtualized architecture is slower than the native architecture. For this architecture, the CPU usage can increase from 40 to 60% [19]. Indeed, the resources dedicated to the virtualized environment reflects the real needs and requirements of the real tasks compared to the traditional architecture where a physical or virtual machine can share the same resources and perform the same tasks.

1.2.4 Objectives and use case

Seen that the large and increasing use of specific and proprietary device for the network operators and the costs involved, NFV is a new direction to minimize the amount of hardware required. Virtualization is used to avoid repeating operations, installing new proprietary hardware, promote elasticity, and incurring new purchase, installation, operating, and energy costs to maintain new network service.

NFV's goal is to change the method of network operators and network designers to manage and operate the network infrastructure through the development of virtualization. This change is made by consolidating different types of VNF into standard commodity machines (servers, storage devices, etc) [26]. Knowing that virtualization does not necessarily spread resources while Telecommunications Service Providers (TSP) can buy or develop software (Network Function NF) and run

it on physical machines. These network functions must be able to operate on basic servers. However, flexibility, dynamic scaling of resources, and the energy saving are very good marketing factor for NFV. To address different technical obstacles, ETSI has recognized several pertinent scenarios [8]:

- Network Function Virtualization as a service,
- Virtualization of Mobile Core Network and IMS,
- Virtualization of the mobile base station,
- Virtualization of the domestic environment,
- Virtualization of CDN,
- and Virtualization of fixed access network functions

1.3 Relationships to other emerging technologies

1.3.1 Relationship between NFV and SFC

The characteristics of virtualization (flexibility, extensibility, dynamic scaling of resources, energy efficiency) are the basis of a service function chain (SFC) supply which enables VNF to be placed anywhere wherever and whenever [3].

Thus, SFC can be created by join between the different network functions. For example, SFC dedicated to security is chained in order as follow: first in a firewall, then in NAT, finally at the proxy level. On the other hand, the location of VNFs with the specification of switch and routing of SFCs constitutes an important factor in efficiency of network[14] [29]. In this context, many approaches [21][27] [31] [32] have been discussed regarding the placement of VNF for SFC requests. [31] proposes CALVIN, is an approach allowing to manage the chains of functions of distributed service (SFC) for the tactile Internet applications with low latency.

Indeed, CALVIN implements the VNF virtual network functionality in the kernel space (if VNF only requires simple treatment) or in the user space (if VNF requires advanced treatment) avoiding transmission between these two spaces to process a specific VNF file. [27] and [32] work on the virtual function chaining, [21] studies the problem of placement and chaining of VNFs which mainly consists of real application time and the availability of VNFs by proposing as solution an algorithms and mechanisms for placement and chaining in the NFV environment. In [32], the

problem posed consists of recent network architectures of mobile which complains from ineffectiveness of flexibility and agility to be able to integrate quickly recent services and the ability to adapt to big industries. Under this measure, proactive caching and other virtual network function (VNF) files that may be invoked to perform the services in an integrated manner are considered to be based on the detail of a new mathematical programming framework. This method is suitable for implementation strings cache VNF and a heuristic without scale Probability-Prior Proactive Caching-Chaining (PPCC) as a solution to increase the network efficiency. There are also many idle servers on the network as SFC requests arrive and leave dynamically with the evolution of resource requirements, in particular at low load times [29] [13]. Knowing that servers in standby state consume an average of 70% of the peak power consumed [33].

1.3.2 Relationship between VNF and NS

A set of chained VNFs constitutes a Network Service (NS), for each NS a certain number of VNFs placed in an order and in chaining. A chain of VNF is placed in the (NFVI) Network Function Virtualization Infrastructure. The VNFs [34] can implement in virtual machines wich is installed in OS or directly on the hardware and controlled by native hypervisors or VMM (Virtual Machine Monitors). Thanks to NFV, NS doesn't need to install of hardware and physical space, which incurs acquisition, operation and energy costs. Depending on the NS requested, their properties and budget constraints etc, the network service provider is identified. The quality of these services is also discussed with the service provider. Therefore, the NS is subject to SLA constraints to describe the QoS parameters [35] [36]. The definition of NS is mainly determined by the concept of NFP and VNF-FG as illustrated in Figure 1.3; Network Forwarding Path (NFP) presents the path of actual traffic flows on Virtual Links (VL).Forwarding Graph of VNF (VNF-FG) is a key concept to know how to control traffic and user traffic where we speak of two types of VNF-FG; VNF-FG for control traffic and VNF-FG for user traffic.

1.3.3 NFV in relation to Software Defined Networks (SDN)

NFV is specified by close relationships to an other promising and recent technologies, such as SDN [33] [37]. Thus, SDN is a networking concept that enshrines

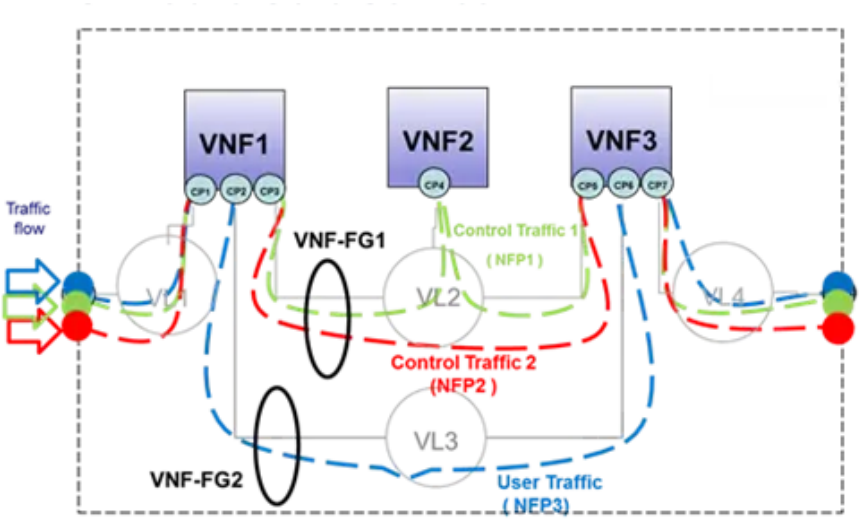

Figure 1.3: Network Service (NS) architecture

centralized management and enables intelligent control of individual hardware components using software. It is a specific interface that dissociates between the control management and the underlying data management.

NFV is in mutually collaboration with SDN at level of services and benefits, which are worked in complementary and have the same characteristics; innovation, creativity, openness and competitiveness [13] [33].

SDN integrate to NFV and can be used to address dynamic resource management and service coordination. By using SDN, NFV also allows dynamic and real-time delivery of functions as well as flexible traffic redirection. While, NFV can help SDN create dynamically a virtual service for certain service chains that involve specialized hardware and complex tasks to deliver a new service demand [33].

1.4 VNF consolidation problem in Cloud data centers

Concisely, the consolidation of VNFs mainly consists of migrating and placing VNFs in the most suitable location to balance the consumption of resources while minimizing the active elements of the locations and the energy consumed. In the literature, the most frequent locations for the allocation of VNFs are: VMs, servers, containers, ... etc. For example, consolidation decreases the number of servers used by moving VNF1 from server S1 to host server S2 in order to empty and deactivate S1 as shown in Figure 3 [13]. Knowing that two criteria must be taken into account: multiple resources of the server at the same time, such as

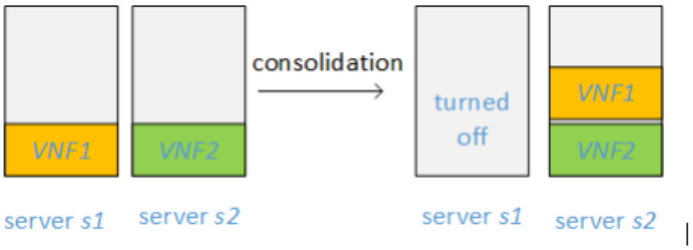

Figure 1.4: Simple example of consolidating VNF into NFV [3]

CPU, RAM and I / O. This forms a multidimensional packaging problem much more difficult than that of a single dimension and reduces the data center power consumption by putting the servers in a low power consumption mode during their low usage period. A virtual machine consolidation solution should create periods of server inactivity by grouping virtual machines into as few servers as possible. Various research studies have been carried out[10] [38] [9] [39] to solve the problem of consolidation of VNFs, and more different techniques for solving the problem have been proposed. It associates between two extensions derived from the same FCA source with the same objective but in a different domain (special type of pattern structure).

1.4.1 Consolidation with factors, reasons and requirements

Among the main factors of consolidation are: exploitation, communication and maintenance. For exploitation factor, the consolidation is used for three main reasons as shown in the table 1 below: 1) energy management, 2) load balance, 3) fault tolerance. While the communication factor involves consolidation which is used to improve the management and cost of communication. In addition, consolidation can support the system through maintenance. Thus, the consolidation meets the main needs of cloud which aims to optimize the service quality based on the following conditions:

- Performance: this is about ensuring good performance and avoiding violations of service level agreements (SLAs).
- Scalability: system capability to withstand demand and service overload by using additional resources.

- Efficiency: is based mainly on the balanced use of resources and on the achievement of elasticity.
- Reliability: aims to maximize service reliability and machine reliability.
- Availability: reflect on the performance and robustness of the system.

The consolidation takes into account several requirements and restrictions, the most important the capacity of the allocation, the equivalence between the elements of consolidation in CPU, memory, RAM, etc.

Table 1.1: The circumstances of consolidations: reasons, factors,requirements and restrictions

Circumstances	**Factors**	**Reasons**
Consolidation, Factors and Reasons	Exploitation	-Energy management -Load balance -Fault tolerance
	Communication	- Improve management -Improve cost
	Maintenance	Support system
Consolidation requirements	-Performance -Scalability -Efficiency -Reliability -Availability	
Consolidation restrictions	-Memory capacity -CPU -RAM	

1.4.2 Multi-measures based approaches

In this classification party, we find several approaches to the problem of consolidation taking into account multi measures. The authors in [3] approached the problem of consolidation of VNF from several sides such as; energy, bandwidth, migration cost by following a new method of consolidation known as VCMM (Consolidation Method Based on Multiple Status Characteristics of VNF). This method exploits the intelligent characteristics of an artificial neural network [39] known as a political network as a learning factor. It checks the state of several features at the end of each period and predicts the likelihood that servers will be down in the meantime following. In addition, the PSO (Particle Swarm Optimization) algorithm [40] is applied to improve the benchmarks that are within the policy of

networks to produce a suitable server shutdown policy. In the goal to migrate VNFs (whose hosting servers must be deactivated), VCMM exploits a greedy method to relocate them to a fresh optimal position.

On the other hand, in the works proposed in [9] and [8] the problem of consolidation was approached from a single measure. The authors in [9] followed an Adaptive Interference-Aware (AIA) heuristic approach based on automatic placement of VNFs in custom 5G network slices while avoiding interference. In addition, [8] proposed a solution of virtual machine placement algorithm by applying the formal concept analysis(FCA) grouping technique which first groups virtual machines in communication with each other to execute a task, then consolidates a virtual machine or set of dependent virtual machines into the fewest number of servers and racks then consolidates dependent virtual machines into the fewest number of servers and racks to maximize and balance resource utilization. As with other approaches, [41] has addressed the problem of consolidation for the purpose of reducing energy costs in data centers mentioning the effect of migration on service response time [42] . Contrary to what prevails in most of research, in [41] an attempt to avoid unnecessary costly VM migration by proposing a method of migration control based on LP formulation and heuristic. Also, for the same objective, [43] developed and exploited the pMapper structure and some server consolidation algorithms for virtualized resources.

1.4.3 Multi-resource approaches

The multi-resource approaches are mainly aimed at achieving a balance in the exploitation of resources [44] [45]. For example [8] provided an environment to ensure the maximum and balanced exploitation of resources by ensuring an optimal allocation of virtual machines by FCA. In [38], the proposed approach consists of a dynamic resource provisioning planner for the joint adaptive adjustment of: admitted traffic; throughput provided and reconfiguration of resources and consolidation of virtual networked data center platforms. In addition, [46] proposed a new virtual machine consolidation framework for Apache CloudStack which is an open source cloud platform software package. This proposed framework is an emerging architecture that brings together different systems of resource monitoring, mechanisms of energy control and VM packaging algorithms.

On the other hand, certain approaches use only a few resources specified for example in [9], two resources have been mentioned which are memory and CPU. Other approaches have not mentioned any resource such cases [10] [3]. Whereas, [47] carried out a full investigation especially on the allocation of resources in the context of NFV. This allocation consists of three stages. Thus, the first stage is VNF-Chain Composition VNFs-CC which serves to efficiently associate the VNF in order to generate a desired NS in the most adequate manner, respecting the objectives of the service provider. The second step is the VNF-Forwarding Graph Embedding VNFFGE which serves to position the location in NFVI, where the VNFs will be placed appropriately, taking into account the individual needs and requests and then the global requests of all NSs. The third and last step is the VNFsSCH stage, which is used to select the preferable time to implement each function respectively in the NFVI to decrease the total implementation time without affecting the efficiency of the service and considering all the priorities and rules between the VNFs. To reduce energy consumption and wasted resources, [13] presented a solution based on VM consolidation that focuses on the use of resources in balanced operation and on 3-dimensions (CPU, memory, and E / S). To adapt the VM consolidation problem in big data centers, the mentioned approach is based on the homogeneity of the ACO (Ant Colony Optimization) meta-heuristic with an equitable utilization of resources. But in [48], resources are studied just in their effect on VNF placement and processing in single dimension (CPU).

1.4.4 Heuristic and meta-heuristic based approach

Consolidation issues in cloud infrastructure as a difficult NP issue are primarily addressed through heuristic approaches. Therefore, the majority of approaches follow the greedy heuristic model to consolidate by proposing mutations of greedy simplified algorithms such as First Fit Decreasing (FFD) [49], Best Fit [50], Best Fit Decreasing [51] , and so on [52] [53]. The First Fit strategy was used to select the destination of the containers'[54] and the appropriate host. Whereas, Best Fit strategy was used to put the attributes in objects that have the least amount of resources available and have more capacity to pack. The comparison works in [55] showed that ACO is better than greedy algorithm FFD through better server utilization and less energy consumption.

AIA (Adaptive Interference Aware) is also classified among heuristic algorithms [56], which are used to avoid interference and to adapt the demand model by consolidation of VNFs based on automatic placement of VNFs in custom 5G network slices. Furthermore, [56] considered that excessive use of VNF resulting from dynamic movement (input and output) of network services is a problem of VNF consolidation. Therefore, they proposed a formulation of Linear Integer Programming (ILP). Also, they proposed a greed-based heuristic for solving large-scale issues. Heuristic algorithms differ and vary but gather in precise points like COMBINED which is based on FFD [57].

The meta-heuristic approaches mainly consist in genetic algorithms (GA), particle swarm optimization (PSO) and swarm intelligence algorithms. The first family is evolutionary approaches [58] [59] based in genetic policy, and the second family is approaches that mimic animal movement (biology or ethology). For the latter category, ACO [14] [60] and PSO are the commonly used methods in resource allocation. Therefore, you can classify the consolidation problem among the resource allocation problem. PSO is a stochastic technique which is based on candidate solutions to develop an optimal solution to the problem e.g [61]. ACO is probabilistic methods inspired by ant behavior and constitute a family of optimization meta-heuristics for naturally solving complex problems. These methods are among the meta-heuristics intended to solve the more difficult optimization problems, more precisely the continuous variable problems.In [14], ACO is an algorithm dedicated to dealing with the problem of placing VMs in multi-objective dimension (minimize the waste of resources and energy consumption). Some approaches of consolidation followed method that different from what is prevalent such as in [10] which based on a policy of the Monte Carlo Tree Search approach. In [62] [41], another method adopted for consolidation problem is based in constraint programming and in [62] linear programming was used. The heuristic and meta-heuristic algorithms are generally used for the less complex consolidation problem while in the consolidation problem which considers the constraints as the chaining for the consolidation of VNF uses other type of the more adapted algorithms [10] [27] [31].

1.4.4.1 Consolidation based algorithms

The consolidation algorithms that are studied can classify and organize according to the taxonomy of figure 1.5. The heuristic, meta-heuristic, clustering, LP, CP and AI algorithm are the most common known algorithms for the cloud data center consolidation problem. In the consolidation problem, there are two heuristic algorithm; Greedy algorithm [49] and COMBINED [57]. This type of algorithm aims to quickly provide a workable solution for a difficult optimization problem. Whereas, the meta-heuristic algorithm includes genetic algorithm [63] and Swarm intelligence algorithm. Generally, meta-heuristics aims to find the best approximate solution by learning the characteristics of a problem. This last is divided into two algorithm classes which are PSO [61] and ACO [14] [60] . As we have seen previously, each type of these algorithms has a derivation dedicated and adapted according to the problem of each proposed approach. This derivation can change the complexity of algorithms and also their type and purpose. For example, deriving Fuzzy-FCA from FCA [8] [15] changes the problem from type of uncertainty to probabilistic [64] based on the fuzzy context. In [65], the author shows the scaling of a fuzzy context into a minimum pattern structure which is a special type of pattern structures to study the relationship between pattern structures and FCA. Thus, this work is an extension of three dimensions; FCA, Pattern structure and Fuzzy-FCA. It associates between two extensions derived from the same FCA source but with different objectives and in a different domain (treat the complexity of descriptions by Pattern structure and uncertainty by Fuzzy-FCA). There is also another type of hybrid algorithm which associates between two types of algorithms of different family for example in [41] associating between LP and heuristic algorithm and in [66] [67] associating between genetic algorithm and heuristic.

1.4.4.2 Discussion

In all the works, each approach has tried to choose the most appropriate methodology according to the complexity of the problem proposed. But, there are approaches that its algorithm derive from other existing algorithm and surpass them on a few criteria such as in [60] ACO-BF derived from ACO. This construction of different algorithms can classify as shown in figure 1.5. These different algorithms have adapted with the proposed consolidation problem in a consistent way but

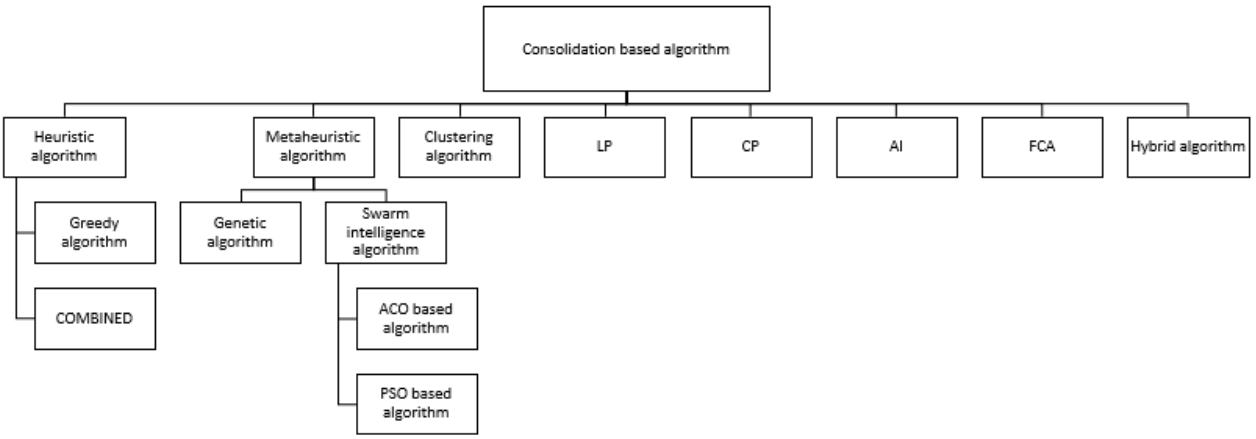

Figure 1.5: Consolidation based algorithms

the majorities of these approaches have provided an adequate environment alongside the algorithm used to associate between conflicting objectives [63] [68]. Thus, [63] proposed a genetic algorithm in a cloud-based MEC environment to minimize response time over the length of network and at the same time to provide a highly available service. In [41], the consideration of the contradiction between accommodation capacity and minimizing the number of active accommodation by control algorithm presents a weakness point that can logically weaken the value of the approach. Other approaches have used two types of algorithms such as LP and heuristic algorithm in [66] [67] [66], between LP and Artificial Neural Network (ANN) algorithm [68] and between genetic algorithm and heuristic in [41] which affects the complexity of the proposed system. Furthermore, in terms of time, heuristic algorithms are lighter in computation compared to meta-heuristics [41] and LP [68], which explains their effect on complexity. There also other type of algorithms discovered based on graph directed that inspired from an algorithm (for example Almohamad's Linear programming) to an other as in [69].

1.5 FCA-based approaches

1.5.1 Problem formalization

To formalize a problem in mathematical form, FCA (Formal Concept Analysis) answers by bases suitable to transfer such a problem in mathematical form via

binarization. FCA is a methodology based on computer intelligence to classify data and generate concept from the formal context, it is used in large scale and in different fields. Also, it determines and characterizes the relationships between objects and attributes.

The authors in [8] proposed an approach of placement based on the classification technique FCA which first groups the communicating virtual machines to implement a certain task, then moves the respective virtual machines to as few servers and racks as possible to maximize and balance resources utilization taking into account communication between virtual machines. This approach is made up of two techniques. The first is FCA, which allows a group of VMs and group of tasks to interact and generate a concept. The second technique is the recruitment process in Ant Colonies to place VM clusters on the fewest servers and racks. In [70], FCA was proposed to minimize power consumption and address the scheduling VMs problem by unloading tasks in MEC. More precisely, FCA was summarized in rules generated by VMs and tasks which are considered as objects and attributes of formal concept. Thus, FCA characterizes the relationships between objects and attributes. This solution is made up of three key modules:

- Characterization of the profile of VMs based on FCA: This module is for characterizing VM profiles using the FCA algorithm. The idea includes; (1) a formal context on virtual machines and their performance measures is built, (2) the corresponding conceptual network of virtual machines is then built from the context of the constructed virtual machines. (3) Thus applying the set of VM rules extracted from the VM profile.

- Characterization of the task profile based on FCA: Similar to the characterization of VM profiles, another module consists in characterizing the task profiles using FCA. The idea is to build the formal context of the tasks called the task context,then possibility of generating the lattice concept according to the tasks. Finally, extract all the task rules derived from the profile.

- Fusion / Matching: this involves linking the rules of VM and the rules of task to obtain the best match (i.e. the appropriate allocation). Practically, it is

the evaluation of the correspondence between the formal concepts of tasks and VM. The task rules are efficiently merged with the VM rules arriving at the maximum degree of similarity indicated. In other words,virtual machine-to-task mappings are generated to minimize the movement of data between data centers and power consumption. [15] addressed the placement problem by proposing an FCA algorithm-based data placement approach that aims to bring together the maximum amount of data and tasks into the fewest data centers. And,it is carried out in four stages: 1) the hierarchical organization of concepts(pair of tasks and data) according to rules of FCA, 2) the extraction of candidate concepts, 3) the allocation of data in cloud centers and 4) phase of data replication.

Also, [71] proposed an approach to data placement based on the FCA algorithm which takes into account the original data sets via different devices of communication (routers, switches,...) and the resources used in data center. This approach aims to reduce execution time, the energy consumed of two communication members and the cost of placement by gathering the maximum number of data sets and tasks into the fewest storage servers (SC).

1.5.2 Classification FCA based approaches

The variety of FCA-based approaches reveals different objectives and lines of research concerning consolidation as indicated in the following table 2. Thus, from this table, we conclude that FCA has dedicated the decentralized architecture by cutting with the centralization to guarantee fault tolerance. The table 2 also shows that the migration is not considered with the FCA algorithm, which implies that FCA cannot express the migration but can express the final placement of object or attribute. Also, some approaches, besides FCA, have used other methods to support and strengthen it by adding another algorithm or providing an appropriate architecture.

Table 1.2: Classification of the main VNF placement approaches

Reference	Objective	Architecture	Consolidation components	Collaboration	Migration
[6][4]	The balance in the exploitation of resources	distributed	VMs and servers	with ACO algorithm	No
[57]	Minimize energy consumption	distributed	VM and tasks	MEC Architecture	No
[52]	Minimize data movement, minimize workflow execution time, communication energy consumption and cost	distributed	Data, tasks and conservative storage SC	No	No
[72]	Reduce data movement between data centers and minimize energy consumption	distributed	data, task and data centers	No	No

1.5.3 Limits of the studied approaches and discussion

The FCA-based Grouping-based Virtual Machine Placement (G-VMP) approach [8] ensures the reduction of server power consumption of servers and network communications flows in the flowing steps: first grouping the virtual machines communicating to perform a task, then allocating dependent virtual machines into the fewest servers and racks guaranteeing greatest and adjusted utilize of resources. This approach analyzed the placement problem in a very efficient way based on two additional processes taking into account various measures such as; VM capacity, communication cost, balance in resource exploitation. However, grouping communicating virtual machines together to reduce the cost of communication can lead to interference (co-channel interference).

In [70], the FCA approach succeeded in solving the problem of scheduling virtual machines in MEC by identifying the mapping of tasks to virtual machines. But, this approach uses relational analysis regardless of several measures such as; cost of communication, the balance of resources utilization and the problem of interference. In addition, duplicating FCA in two processes to generate the concept of VMs and tasks can increase execution time. In [72], we cannot include the data replication step among the functions of the FCA since it works in principle on

reduction, not on replication. Therefore, replication is opposed to the principles of FCA and the objectives of approach. In addition, it can cause data center overload and pose the problem of consistency and interference.

1.6 Consolidation at different environmental levels

The environment directly participated in supporting and succeeding the consolidation technique in several works that we saw in the previous sections. This environment is defined by an axis or a level determined for consolidation. Generally, we can see that the studied approaches dealt with different level and axes of network such as IoT, Fog computing, 5G, data center, MEC,...etc as shown in table 3. This variation shows the importance of consolidation at different network levels and with their different compositions.

1.6.1 Consolidation at IoT level

The Internet of Things or IoT defines a network of physical terminals that are directly associated with the transport of objects, whether vehicles or pedestrians, providing intelligent transportation. However, the mobility of objects creates a problem in determining the transport plan of these objects. In this context, [73] proved that the main challenge is in object mobility and is the rapid variation of network topology of VNF which necessitates frequent reconstruction by proposing a new method of placing VNFs called Border VNF Chain Placement (BVCP). This method is based on the division of the graph into multiple sub-graphs and the total use of the control programs. So, IoT often deals with the placement of VNF in chaining. Integrating IoT with VNF considers a great challenge that requires high-level peripherals to optimize end-to-end latency. Thus, the couple of IoT and peripheral edge constitute a main basis for IoT service virtualization approaches [73] [68] [74]. In [75], IoT used as an agent in cloud fog environment to solve the workflow problem.

1.6.2 Consolidation at Fog cloud level

Fog computing is used to minimize the latency induced by remote clouds by distributing computing resources in edges. In fog computing, virtual network functions can be the components of applications that can be implemented in a chain. Thus, the placement of VNF for fog computing is mainly the placement of application components on infrastructure nodes [76]. Applications are defined by building blocks called micro-services. The placement algorithms of VNF consider not entirely appropriate for the decentralized fog system [77] because the criterion of mobility of the fog nodes and the decentralized infrastructure of end-user consecrates the decentralization. Thus, [77] proposes an investment strategy based on Markov processes without centralized orchestration. Also, [78] proposes a strategy based on the distribution of resources to avoid the centralization and follow new strategy.

1.6.3 Consolidation at MEC level

The MEC architecture can extend the placement and distribution interval of chained VNFs in service to accommodate critical and time-sensitive traffic. MEC contributes in distributing compute resources to the end user in order to minimize the latency. In this context, it is possible to associate MEC and IoT, since the MEC infrastructure meets the requirements of IoT and provides the appropriate environment for their proper functioning [74]. Thus, all the research that works on IoT of objects is based on the MEC and Fog architecture because these two environments provide very low latency. Equally, these two environments establish decentralization. Also, these two environments devote decentralization [79] by considering the characteristics of user mobility.

1.6.4 Consolidation at 5G level

Different architectures reacted and adapted with the problem of consolidation and placement of VNF for 5G. Mobile network operators rely on 5G technology to respond to the market demand and with mobile data movement. Despite the importance of 5G technology services, the importance of widening the range of dynamic network functions has encouraged researchers to open up to VNF and

distribute 5G network functions as VNF [80]. The 5G network is sliced to apply VNF to each slice for the requested service. In this context, some works has proposed to exploit a 5G structure which has a combination of an advanced cloud server and a main cloud server to dynamically adapt with service requirements. All the expectation mentions that the 5G network can introduce multi-service by supporting a group of vertical use cases, like Internet of Things, remote machines, autonomous driving and virtual reality (VR). This generation [81] makes it possible to properly specify the placement of VNF requested for a service in each slice with meeting the service needs and improving the quality of service.

1.7 Critical analysis of the results of some studies on consolidation

After studying some referential works and evaluating the methods, we devote this part to discuss the results of some works concerning consolidation at different levels of environment as illustrated in table 3. Indeed, the efficiency of consolidations can vary from one machine to another because of the heterogeneity of the platforms and the resources [82]. Also, there are other factors related mainly to CPU and workload performance as well as to different heterogeneous service levels. Figure 1.6 presented by Cho et al [4] in the context of processing the VNF placement problem for low migration latency for efficient resource management based on control of active VMs number. This figure shows the difference between before and after VNF migration latency based on workload. As illustrated in Figure 1.6, the algorithm proposed by Cho et al dramatically minimizes network latency through VNF migrations which proves the effectiveness of the VNF-Real Time Migration (VNF-RM) algorithm. Thus, Figure 1.7 proves the dependency relation between the number of VM and the number of migration, as well as to latency rate and workload (Figure 1.8). Generally, the VNF-RM algorithm has succeeded in minimizing migration latency in order to improve resource management considering the following factors:

- Ensure that the total number of CPU requests for each network service (VNF chain) is not greater than the CPU capacity of the connected VMs.

Table 1.3: Table of consolidation at environment level

Environment level	**Type of architecture**	**Task**	**Connectivity models**
IoT	Decentralized	-provide intelligent transportation - integrate sensors, software and applications to ensure connection and exchange of information between mobile objects via the Internet	machine to machine, machine to cloud, machine to gateway, Back-End-Data Sharing
MEC	Decentralized	-allows to distribute IT services closer to an end user and move from a centralized cloud to an edge network −Aims to minimize response time and provision the suitable infrastructure for real-time applications with high bandwidth	From cloud center to peripherals
5G	Decentralized	-enables new functionalities, in particular virtualization and new architectures, conducive to the development of connected objects, applications hosted in the cloud, passing through various network layers-provide access to speeds that far exceed those of 4G, with very short latency and high reliability, while increasing the number of simultaneous connections per covered area.	Full-duplex, multi-user MIMO
Fog	Decentralized	store and process data through the use of equipment and resource located at the edge of cloud	works at the intermediary interface level of connected objects and cloud environment

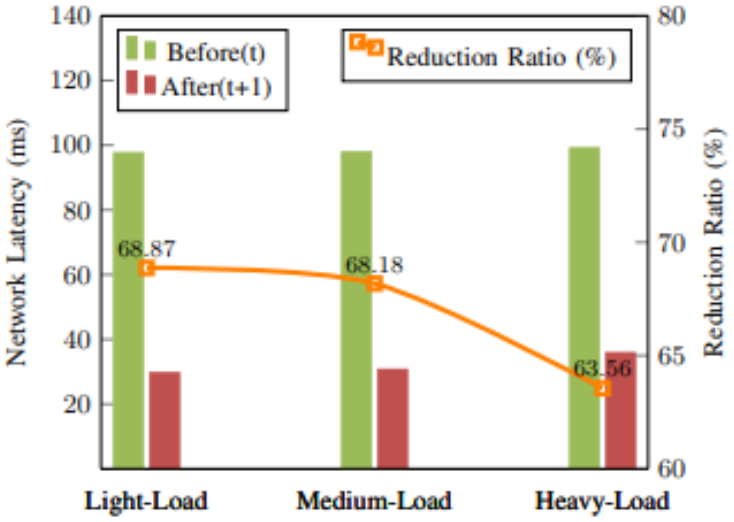

Figure 1.6: Network Latency variation before and after VNF migrations [4].

– Ensure that the total width of the network bandwidth required by a network service is not less than the available network capacity of the connected VMs.

However, this algorithm does not take into account the cost of migration and communication interference which requires to revise the efficiency of this algorithm on the large-scale.

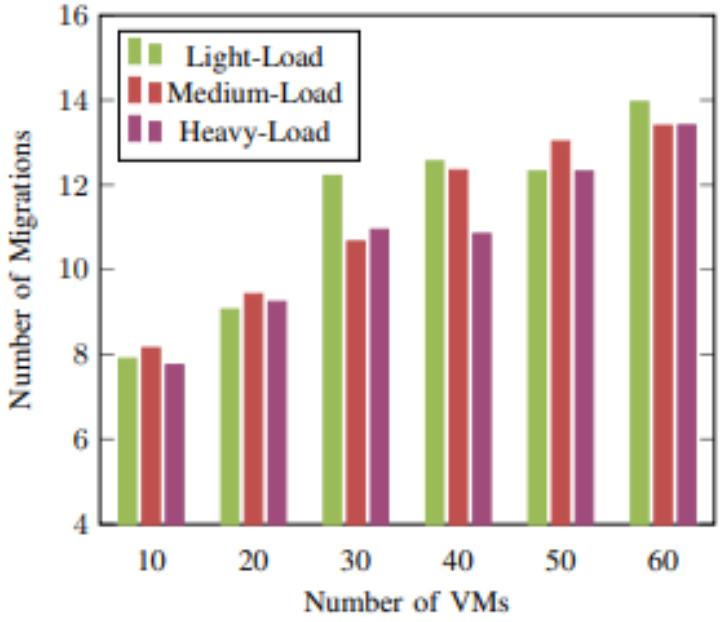

Figure 1.7: The variation in the quantity of VMs according to the quantity of migration [4].

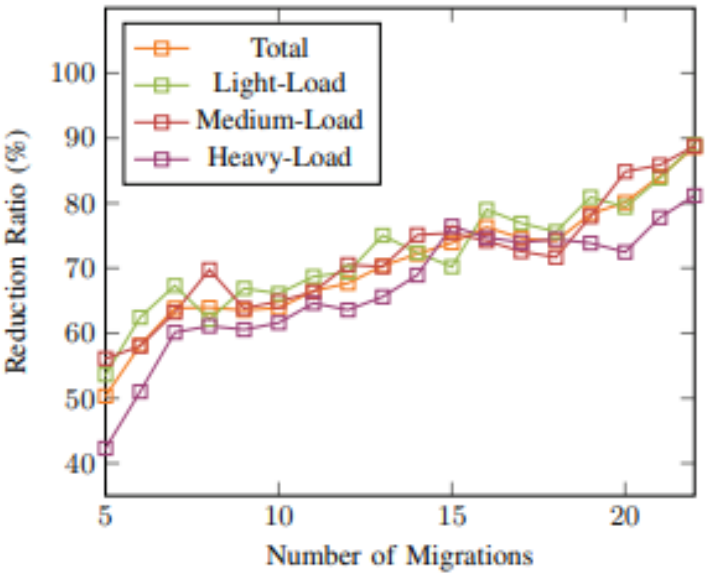

Figure 1.8: The reduction rate of workload and number of VNF migration [4].

1.8 Major findings and challenges

1.8.1 Major findings

Based on our research regarding consolidation in cloud infrastructure, we were able to rank the most important jobs according to shared criteria. These criteria allow us to conclude and determine the following major discoveries: Migration during consolidation can be a real problem of latency time whereas it can be controlled by specific control systems to minimize response time and cost e.g VNF-RM algorithm [59]. And, we can separate between dynamic placement and static placement by the migration factor. Controlled migration also presented a solution to the problem of hosting capacity and interference. Therefore, it may be necessary to balance between the importance of migration and the benefits of consolidation. Also, the consolidation directly affects the cost of the cloud network service for both the customer and the provider. Thus, consolidation serves to reduce the cost of service by reducing the number of active machines which costs more resources and consumes more power. On the quality side, there is a great focus on quality of service in most research [83] [9] [74] [84] with highlighting the constraints of Service Level Agreement (SLA) violation. Highlighting on services, allows us to talk about the relationship of SFC and VNF to provide a service (NAT, firewall, DPI, etc.), hence a well-ordered chaining is an SFC as illustrate in previous section.

1.8.2 Challenges

The VNF consolidation problem is a branch of the VNF placement problem, each one have their challenges as illustrated in Figure 1.9. The VNF placement challenges are the dynamic placement as in [32] [73], real time application [21] [27], chaining [31], big data [3] and allocation of resources [29] [46]. For VNF consolidation challenges, we can mention interference [9] [85], migration cost [41] [4], the balance in resource exploitation [29] and capacity of allocation.

Thus, the VNF placement problem is related to the general characteristics of the VNF architecture, while the consolidation problem is related to allocation technique.

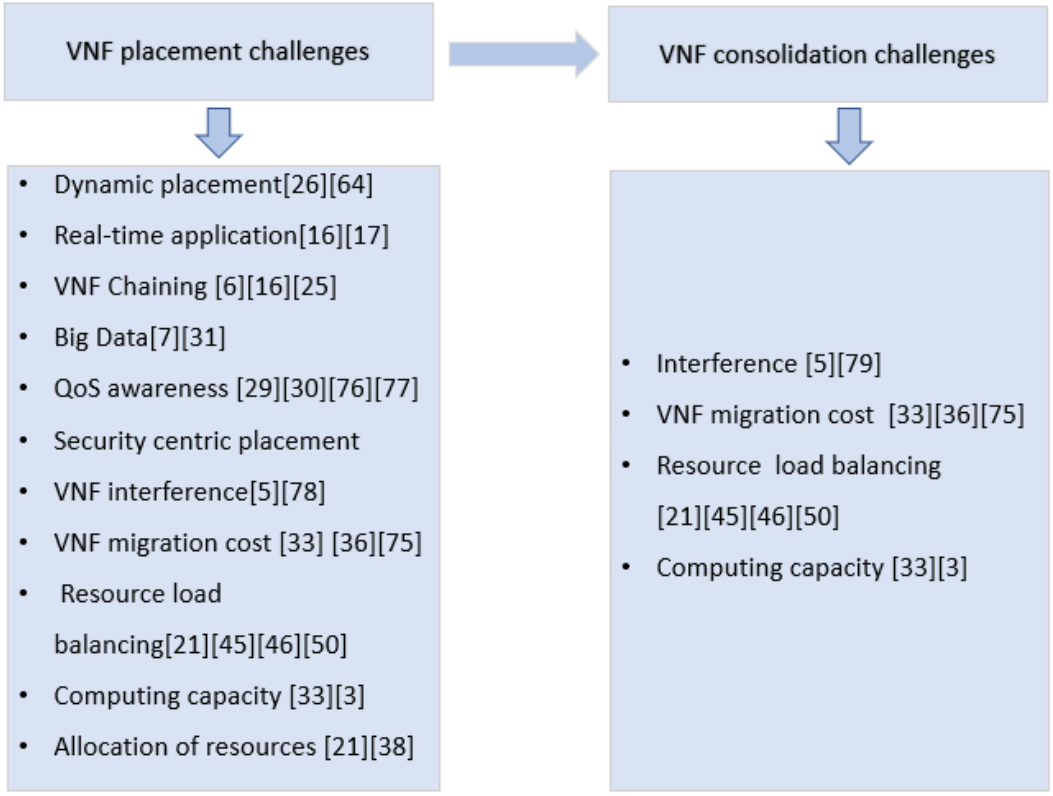

Figure 1.9: VNF placement and consolidation challenges

In fact, the conflicting in the objectives of consolidation as we seen in some works [3] [41] [45] [4] presents a big challenge; [3] treated with conflicting goals such as saving power, reducing bandwidth used, and reducing migration cost making implementation in practical environments difficult. Therefore, it proposed two

methods to create a favorable environment for consolidation. [41] involves the challenge of contradiction between accommodation capacity and reducing the number of active accommodation by using a control algorithm based on LP formulation and heuristic. In [45], the contradiction consists in bringing together the integration of several VNFs in a smaller number of resources and the desire to minimize latency. However, this can negatively affect the additional latency due to the sharing of same resources. [4] presented a real challenge to bring together low network latency and VNF migration by proposing a new VNF Real-Time Migration (VNF-RM) algorithm.

1.9 Summary of VNF placement and consolidation approaches

Table 1.4 illustrates some NFV investment approaches according to standard characteristics. This table has identified for each research output 1) the objective of the proposed approach, 2) the constraints that may appear when carrying out the approach, 3) the proposed algorithm / approaches / policies, 4 auxiliary environment that help to provide a favorable conditions to successful of approach; and finally 5) Suggestions for improving the proposed solutions.

Table 1.4: Classification of the main VNF placement approaches

Ref.	Objective	Constraints	Algorithm / Approach	Auxiliary environment	Suggestions
[10]	– reduce resource usage and optimize energy consumption through consolidation that serves to reduce the number of servers and switches used by prioritizing the most energy efficient hosts	– Lack of available physical resources – network complexity	– EE-TCA algorithm	– No consideration	– Instead of the 'Energy Efficient Tree algorithm search-based Chain placement Algorithm (EE-TCA), we propose an algorithm based on the neuron network because the chain structure of VNF is similar with it.
[21]	– Minimize response time for real-time applications – Minimize the cost of resource allocation (server, memory and storage)	– protection level – length of the service chain	– ILP and heuristic algorithm named Degree Based Heuristic (DBH)	– No consideration	– Adapting the security model to IoT level

Other than the placement problem, VNF encountered multiple problems such as the problem of orchestration, congestion, consolidation, etc. Consolidation addresses the problem of placement in very precise place of NFV architecture. However, there are also other pivotal criteria to consider in order to effectively manage and consolidate VNF such as:

– Architecture: we designate by architecture, the type of architecture of the approach (centralized, decentralized).

– Technique: this criterion designates the type of technique used in solving the consolidation problem of virtual machines (algorithm / methodology).

Ref	Objective	Constraints	Algorithm or Approach	Auxiliary environment	Suggestions
[27]	– Minimize the delay for real-time applications – Minimize the cost of resource allocation	– allocation of resources – deployment time	– DBH	– No consideration	– We can propose another algorithm which can reduce the number of assignments on the contrary of BDH that of dichotomous search algorithm which divides the assignments in half to facilitate the search.
[31]	– Reduce latency – Increase scalability and flexibility	– CPU – function service chaining	– CALVIN	– MEC	– Suggest a solution based on on extraction of statical features [86] with FCA classification.
[63]	– Minimizing access latency – Maximizing service availability	– Conflicting between their two objective	– Genetic algorithm	– MEC	– Suggest another solution for recent problems such as real-time and security application problems.
[77]	– reduce the power consumed in fog nodes and the costs of communication between applications.	– decentralization and haven't central coordination.	– Markov approximation approach.	– Fog computing.	– Suggest PSO algorithm to distribute the function.

– Use of resources: this concept is strongly linked to the problem of placing virtual machines. It indicates the number of dimensions supported by an algorithm to optimize resources usage in servers.

– Energy consumption: this criterion determines whether a solution meets the energy saving requirement in Cloud data centers and indicates if the case requires of energy saving that an algorithm targets to minimize.

– Placement / Planning: to separate between VNF problems into two main problems: planning problem and placement. Usually, the complexity of allocation and placement operation is related to the network architecture and the type of communication between VNF.

Table 1.5: VNF/VM consolidation approaches

Ref	**Architecture**	**Technical**	**Resource used**	**Energy consumption**	**Planning/ placement**	**Report (consolidation basis)**
[8]	centralized	G-VMP algorithm	4-dimensional	- in servers - In the communication network	placement	VMs and tasks
[9]	centralized	Heuristic algorithm AIA (Adaptative Interference Aware)	2-dimensional	in servers	placement	requests and offre
[3]	Centralized	Neural network and PSO algorithm	n-dimensional	in servers		VNF and servers
[46]	decentralized	A VM consolidation framework for Apache CloudStack	2-dimensional	in data-center cloud	planning	VM and plateform
[52]	decentralized	FCA	4-dimensional	in servers	planning	VM and tasks
[56]	centralized	ILP and greedy based heuristic	n-dimensional	in servers		VNF and severs
[87]	centralized	architecture for 5G controllers plane and graphic algorithm	n-dimensional	in servers	planning	VNFs and controllers
[74]	decentralized	an optimal placement algorithm based on the Tabu Search meta-heuristic in cloud and MEC infrastructure	2-dimensional	in servers	placement	VNFs and network cloud infrastructure

1.10 Conclusion

The wide diffusion of future NFV-based network models depends on several measures which should be taken into consideration when placement and consolidation of VNF is done. VNFs are characterized by their ability to be dynamically deployed according to requirements like the capacities of the NFV infrastructure nodes, which makes it the source of management and cloud control. Recently, the issue of VNF consolidation and location has turned on a popular topic in the field of cloud computing focusing on the active host cost issue as the right VNF placement can significantly minimize power consumption, workflow execution time and increase data centers efficiency. This challenge brings several NFV problems, among them consolidation, orchestration and resource allocation. This chapter discussed a full investigation of the VNF consolidation problem starting from placement. At overall, our aim is to provide clear planning with a detailed analysis of the relevant research related to VNF consolidation and including work focused on orchestration [88], resource allocation [89], energy efficiency [90] and VNF interference [91]. From this study, we note that all the approaches aim at minimizing energy consumption which generally corresponds to the gain in the first step whereas the quality of service is in the second step. In this context, we propose in the next chapter an algorithm that brings together the quality and the gain in consolidation of VNFs to minimize the number of active machines, reduce latency and cost and optimize energy consumption.

Chapter 2

Energy Aware VNF Placement and Consolidation in Cloud Data Centers

2.1 Introduction

The virtualized network function (VNF) is accountable for network services that operate on virtual machines and can establish connections individually or combine together to form a unified enclosure, utilizing all available resources within that enclosure. This adaptability enables the utilization of physical and virtual resources in a manner that guarantees control over power usage, resource distribution, and cost and latency reduction. Drawing on the findings from the preceding chapter regarding the latest advancements in VNF placement and consolidation optimization, our proposal introduces a novel Fuzzy-FCA approach for VNF placement, employing Formal Concept Analysis (FCA) and fuzzy logic in a mixed environment consisting of cloud data centers and Multiple access Edge Computing (MEC) architecture that ensure the distribution of compute resources to the end user. In the first step, VNF consolidates in NS using FCA algorithm to derive VNF candidate. In the subsequent step, candidate VNF migrates and consolidates into the most suitable active VM to reduce the count of active VMs and decrease energy usage by employing the Fuzzy FCA algorithm.

2.2 Fuzzy-FCA-based proposed approach

In this section, we present the issue of VNF placement and elaborate on its aspects and outcomes. The problem of VNF placement mainly involves arranging the physical and virtual resources in an uneven manner so that the virtual machines needed to carry out one or more tasks (VNF) are grouped together on a single server, or else on the fewest possible number of servers. Furthermore, the latency problem and cost associated with it necessitate careful consideration of the significance of deploying virtual network functions (VNFs) both at the edge and within the central cloud infrastructure [92].

2.2.1 Placement and consolidation strategy

We suggest a VNF consolidation strategy based on FCA and Fuzzy-FCA that takes into account the sequencing order of VNF in SFC and the number of accessible NS in order to address the problem of VNF placement. By deploying computing resources close to the end user, the MEC environment that we suggest can reduce end-to-end latency. Our initial goal is to reduce the cost, time (latency) required to optimize service function and energy consumption. Therefore, reducing the number of virtual machines and using MEC together directly results in less energy being used to execute a service. Our VNF placement approach's primary objective is to reduce the number of virtual machines needed to run a VNF as a result of maximizing resource usage and energy consumption. The problem of VNF consolidation in cloud data centers can be characterized as the interaction of two processes: the first process involves connecting VNF and NS, and the second consolidates VNF in VM by allocating the chain of service function and accounting for the number of available NS. The modeling of this issue is displayed in Fig 2.1.

Our technique (Fuzzy-FCA) is broken down into two processes to successfully handle the consolidation problem: the first process groups and places the set of VNF according to dedicated network service, and the second consolidates the set of VNF in virtual machines. In the first process, we propose an FCA method to robustly model the relationship between VNF and NS. The purpose of this method is to run a large number of NSs for each VNF. At the second process level, once the VNF groupings are defined, we apply the Fuzzy-FCA to place the VNF in the VM while

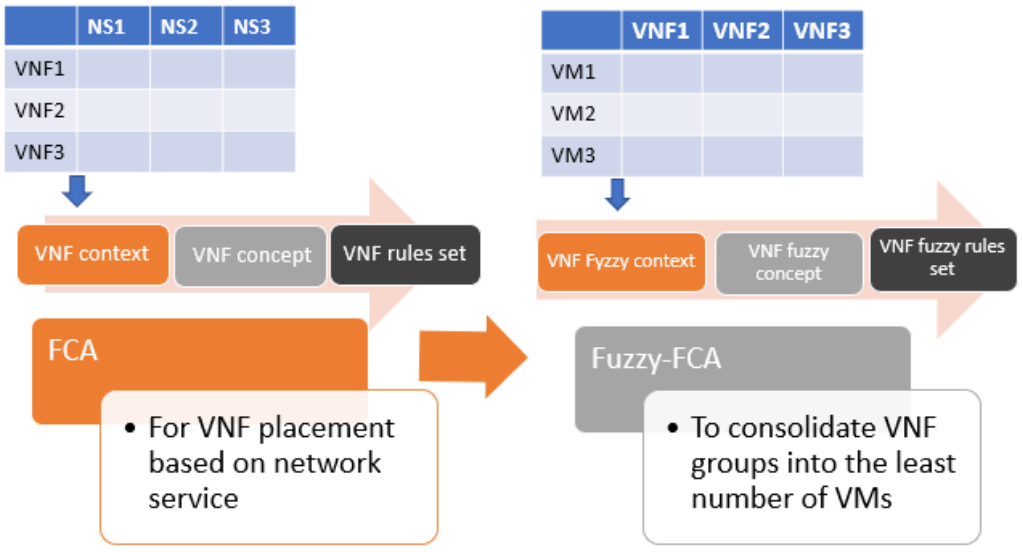

Figure 2.1: VNF placement and consolidation proposed process.

ensuring maximum and balanced exploitation of virtual and physical resources in all dimensions taking into account data uncertainty [64].

The first process determines the best placement of the VNF set by placing the VNFs in the NS based on the FCA. The second process consists of consolidating obtained VNF into the set of VM based on Fuzzy-FCA. After obtaining the FCA rules for the VNF, we apply Fuzzy-FCA on candidate concept and VM to consolidate VNF into VM respecting the sequence of the set VNF. In general, the proposed strategy is based on three main steps : (*i*) the first step consists on the organization of the VNF according to a hierarchy, described by a lattice of concepts, in order to group the VNF and the dependent NS in the same concept. (*ii*) the second step is the selection of the candidate concepts (cc.int)(see definition 9) of FCA and to attach them and use as VNF to the second process of Fuzzy-FCA and (*iii*) the third step is the consolidation of VNF in VM (Fusion by similarity degree function). To clarify the stages of our approach, we present some definitions and notions on which our approach is based.

2.2.2 Model description and problem formulation

We assume that $VNF = \{vnf_1, vnf_2, .., vnf_i\}$ is an attribute set, $VM = \{vm_1, vm_2, .., vm_j\}$ is an object set with the context $K = (VM, VNF, I)$, and I is the binary relation between a VM object and a VNF attribute in order to describe our method. A binary matrix under a table can be used to display this relationship. Knowing that FCA-Fuzzy strategy takes a two-dimensional array as input. In this situation, Table3.1 aids in presenting all the benchmarks taken into account when developing our fuzzy-FCA technique. We provide the definition of formal concept and partial order as follows to filter the relationship between objects and attributes provided in the context table. In which, the formal concept is represented by the pair (O_1, A_1) of the objects O_1 and A_1's attributes such that $O_1 \sqsubseteq O$, $A_1 \sqsubseteq A$.
The formal context's (O, A, I) Galois correspondence is made up of the two applications f and g. The function f is referred to as the dual of O, while the function g is referred to as the dual of A [93].

The pair $C = (O1, A1)$ is referred to as a formal concept of the formal context (O, A, I), if and only if $(O1) = A1$ and $(A1) = O1$, where $O1 \in O$ and $A1 \in A$. In this instance, $O1$ is known as the extension of the concept C and $A1$ is known as the intention of the same concept C. In this instance, $C2$ is referred to as a superconcept of $C1$ and $C1$ is a sub-concept of $C2$. $R(o)$ determines all the attributes a applied to the object o which are brought about by the partial order relation.

Let Aff be the function of assigning a VNF partition in virtual machines with $VM \times VNF \rightarrow Aff$. We formally define VNF placement problem by using the following triplet :

$$PP_{VNF} \equiv\prec VNF, VM, Aff \succ \tag{2.1}$$

Aff denotes the procedure of allocating the group VNF in the group of the virtual machine VM. Every $vm_i \in VM$ is necessary for the execution of $vnf_{vm_i} \in VNF$. Initially, we employ the FCA approach to identify potential VNF that will be merged into VM. This stage involves identifying the interdependence between two VNFs that rely on one or more network services NS in conjunction as follows:

Table 2.1: Description of parameters

FCA parameters	**Description**
$C = (O, A)$ is a formal concept where $O = (o_1, o_2, .., o_n)$ and $A = (a_1, a_2, .., a_k)$.	C is a formal concept which consist of object O and attribute A.
$C_1 = (O_1, A_1)$ and $C_2 = (O_2, A_2)$.	{C_1 and C_2 be two formal concepts of (O,A,I) where O_1, O_2: are the objects of set O and A_1, A_2:are the attributes of set A.
$C_1 \ll C_2$, if and only if $O_1 \sqsubseteq O$ and $A_1 \sqsubseteq A$	C_1 is partially ordered with respect to C_2
$C_i.Int$	$C_i.Int$ is the intention of the concept c_i which gives the attribute of concept.
$f(O) = \{a \in A/(o, a) \in I\}$ $g(A) = \{o \in O/(o, a) \in I\}$	f and g are two functions which constitute the Galois correspondence with the application f is called dual of O and, likewise, g is called dual of A
$VNF = \{vnf_1, vnf_2, .., vnf_i\}$, $VM = \{vm_1, vm_2, .., vm_j\}$ and $NS = \{ns_1, ns_2, .., ns_t\}$	VNF is set of virtual network function, VM is set of virtual machine and NS is set of network service with $i, j, t \in IN$
$Remove(c_i)$ $Eliminate(c_i)$	Function to delete a concept which includes an empty attribute or an empty object. Function of elimination of concepts which contain only one attribute
Fuzzy-FCA parameters	**Description**
α T φ	The extension of membership value. Given threshold value. The extension of object membership where the membership value between the object and the attribute is defined by the membership of the object.

$$Dependency^{VNF}_{vnf_i, vnf_j} = Count(ns_{vnf_i} \cap ns_{vnf_j}) \tag{2.2}$$

This dependency is determined by the count of network service NS that use both vnf_i and vnf_j. Similarly, the interdependence between two network services (NS) is contingent upon one or more virtual machines (VM), which can be expressed in

the following equation:

$$Dependency^{NS}_{ns_i,ns_j} = Count(vnf_{ns_i} \cap vnf_{ns_j}) \tag{2.3}$$

This dependency is the number of virtual network function VNFs that use both ns_i and ns_j.

Furthermore, the process of concept candidate extraction necessitates initially determining the weight of a concept and the maximum of coverage as outlined below. Suppose c_i represents a concept, where $ci.Int$ signifies the intention of concept c_i. The weight of concept c_i, indicated as $W(c_i)$, can be expressed in the following manner:

$$W(c_i) = \frac{|c_i.Int|}{|vnf_i|} \tag{2.4}$$

And the highest extent that covers the collection of VM objects referred to as MC is defined as follows:

$$MC = \begin{cases} 1 & \text{if } \bigcup_{i \in I} c_{i.Int} = 1 \\ 0 & \text{else} \end{cases} \tag{2.5}$$

Finally, let $C = \{cc_1, ..., cc_p\}$ be a set of concept. We define the candidate concept, that has a maximum weight and cover all VNF, noted by $Cand(CC)$, as follows:

$$Cand(cc) = \begin{cases} 1 & \text{if } \sum_{k=0} W(cc_p) = 1 \; and \; MC(cc) = 1 \\ 0 & \text{else} \end{cases} \tag{2.6}$$

2.2.3 Hierarchical organization of network service

To generate a formal context, a set of network services and VNF can generate a formal context from a triple (S, V, I), where S is a set of network services, V is a VNF set, and I is a binary relationship between two series S and V. The binary relation I takes the value 1 if the service ns_i requires the virtual network function vnf_j to run, 0 otherwise. Formal context is presented in the form of tables that show the relationships between services and modeled virtual network functions.

In the construction stage of the Galois lattice, the circles represent concepts, and the arcs between rectangles embody relationships from general (top) to specific (bottom). Several algorithms have been developed to construct Galois lattices. Galois lattice require simplification or reduction to eliminate unnecessary and redundant concepts without losing information. This simplification ensures that the main goal of minimizing the number of concepts is achieved by grouping the largest services and virtual network functions into the same concept. The various operations to simplify the Galois lattice consist in:

−Removing concepts that have an empty object set or an empty attribute set. Formally, a concept c_i is deleted if its objects set $c_i.Ext$ is empty or its attributes set $c_i.Int$ is empty:

$$Remove(c_i) = \begin{cases} 1 & \text{if} |ci.Ext| = 0 \quad \text{or} \quad |ci.Int| = 0 \\ 0 & \text{else} \end{cases} \tag{2.7}$$

−Eliminate all concepts having only one attribute:

$$Eliminate(c_i) = \begin{cases} 1 & \text{if} |c_i.Int| = 1 \\ 0 & \text{else} \end{cases} \tag{2.8}$$

A set of candidate concepts can be extracted from the generated and simplified Galois lattice. Concepts that have a maximum weight of 1 and cover the entire attribute set are called concepts candidate. Therefore, to extract the VNF of the concepts candidate, we use the maximum coverage formula as follows:

$$\bigcup c_i.Int = \{v_1, v_2, .., v_n\} \tag{2.9}$$

After extracting VNF from candidate concepts, we can consolidate them with virtual machines by applying Fuzzy-FCA. Therefore, Fuzzy-FCA is based on uncertainty information and specifies relationships between objects and attributes by precise real values belonging to $[0, 1]$.

2.2.4 Fuzzy Formal Concept Analysis

Fuzzy formal concept analysis (Fuzzy-FCA) is a technique that combines fuzzy logic and FCA technology which represents the uncertainty information as a real number belonging to $[0, 1]$. It is primarily based on relational analysis, which provides a hierarchical and weighted representation in the form of clusters of fuzzy-formal concepts [94]. Therefore, in this step, fuzzy FCA is the best method to place the obtained VNFs on the VM based on filtering [95].

Definition 2.1. Let O be a set of objects, A be a set of attributes and I be a fuzzy set on domain defined by: $O \times A$. The relation $(o, a) \in I$ between object and attribute has a membership value $\mu(o, a)$ in $[0, 1]$. We define a fuzzy formal context [96], noted Fuzzy-FC, by the triple given by the next equation:

$$Fuzzy - FC = (O, A, I = \alpha(O, A)) \tag{2.10}$$

A fuzzy formal context can also be a cross-table as shown in Table 2.2. The context has objects representing the VM, it also has attributes representing the VNF. The relationship between an object and an attribute is represented by a membership value between 0 and 1. The relations that have low membership values noted by threshold T can be eliminated. Table 2.3 shows the cross-table of the fuzzy formal context given in Table 2.2 with $T = 0.5$.

Table 2.2: A cross-table of a fuzzy formal context

O/A	vnf_1	vnf_2	vnf_3	vnf_4
vm_1	0.8	0.12	0.61	0.15
vm_2	0.9	0.85	0.13	0.11
vm_3	0.1	0.14	0.87	0.60
vm_4	0.7	0.10	0.50	0.20

Table 2.3: Fuzzy formal context in Table 2 with T =0.5.

O/A	vnf_1	vnf_2	vnf_3	vnf_4
vm_1	0.8	-	0.61	-
vm_2	0.9	0.85	-	-
vm_3	-	-	0.87	0.60
vm_4	0.7	-	0.50	-

Definition 2.2. Let (O, A, I) be a fuzzy formal context with a confidence threshold T. We define a fuzzy formal concept (or fuzzy concept), noted Fuzzy-FCA, by the next equation:

$$Fuzzy - FCA \equiv\prec (X_f = \varphi(X), B) \succ \tag{2.11}$$

where $X \sqsubseteq O$, $B \sqsubseteq A$ and $B^* = X$.

Let each object $O \in \varphi(X)$, we define the membership o by the next equation:

$$\mu_g = \min_{m \in B} \mu(o, a) \tag{2.12}$$

where $\mu(o, a)$ is the membership value between object o and attribute a, which is defined in I. Note that if $B = \{\}$ then $\mu o = 1$ for every o.

Definition 2.3. Let (A_1, B_1) and (A_2, B_2) be two fuzzy concepts of a fuzzy formal context (G, M, I). We define as follow the next notions:

- $(\varphi(A_1), B_1)$: is the sub-concept of $(\varphi(A_2), B_2)$, denoted as $(\varphi(A_1), B_1) \preceq (\varphi(A_2), B_2)$, if and only if $\varphi(A_1) \sqsubseteq \varphi(A_2) (\Leftrightarrow \ B_2 \sqsubseteq B_1)$.

- (A_2, B_2): is equivalently the super-concept of (A_1, B_1).

Definition 2.4. Let K be a fuzzy formal context with a confidence threshold T, we define a fuzzy concept lattice [97] of a fuzzy formal context K as a set $F(K)$ of all fuzzy concepts of K with the partial order $\leq$ with the confidence threshold T.

Definition 2.5. Let $K_1 = (\varphi(A_1), B_1)$ be a fuzzy formal concept, $K_2 = (\varphi(A_2), B_2)$ is sub-concept of K_1. We define the similarity between K_1 and K_2, denoted as $Sim(K_1, K_2)$, by the following equation:

$$Sim(K_1, K_2) = |\frac{\varphi(A_1) \cap \varphi(A_2)}{\varphi(A_1) \cup \varphi(A_2)}| \tag{2.13}$$

where $\cap$ and $\cup$ refer intersection and union operators on fuzzy sets, respectively. Fig. 2.2 gives the traditional concept lattice generated from Table 2.2, without membership values. Fig. 2.3 gives the fuzzy concept lattice generated from the fuzzy formal context given in Table 2.3. As shown from the figures, the fuzzy concept lattice can provide additional information, such as membership values of objects in each fuzzy formal concept and similarities of fuzzy formal concepts.

A conceptual cluster of a concept lattice K with a similarity confidence threshold T_s is consider as a sub-lattice SK of K which is specified by the following properties:

1. SK has a supremum concept CS that is different of all its super-concepts.
2. Any concept C is different to CS in SK must have at least one super-concept $C' \in SK$ such that $E(C, C') \succ Ts$.

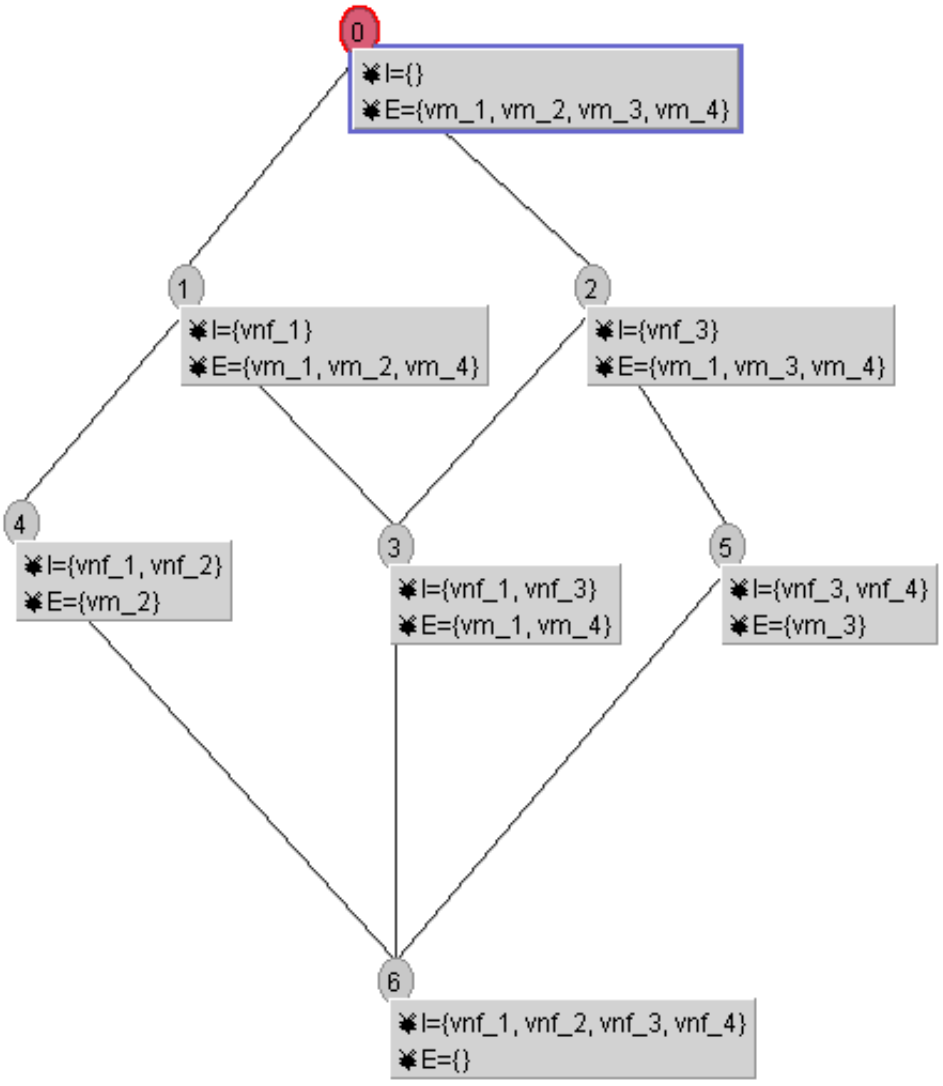

Figure 2.2: An example of concept lattice of traditional FCA.

The concept hierarchy reflects the taxonomy of VNF placement in the virtual machine, in which a VNF placement can be a super-area or sub-area of other VNF placement.

2.2.5 Proposed Fuzzy-FCA algorithm

The first process in this approach is to apply FCA to place the optimal NS network services on the VNFs. This algorithm states that the basic idea of our strategy is to first find the set of VNF candidate (scj.int) that have found their placement and then apply Fuzzy-FCA to them. This algorithm states that the principle of our strategy is to extract the VNF with a higher capacity from the network services

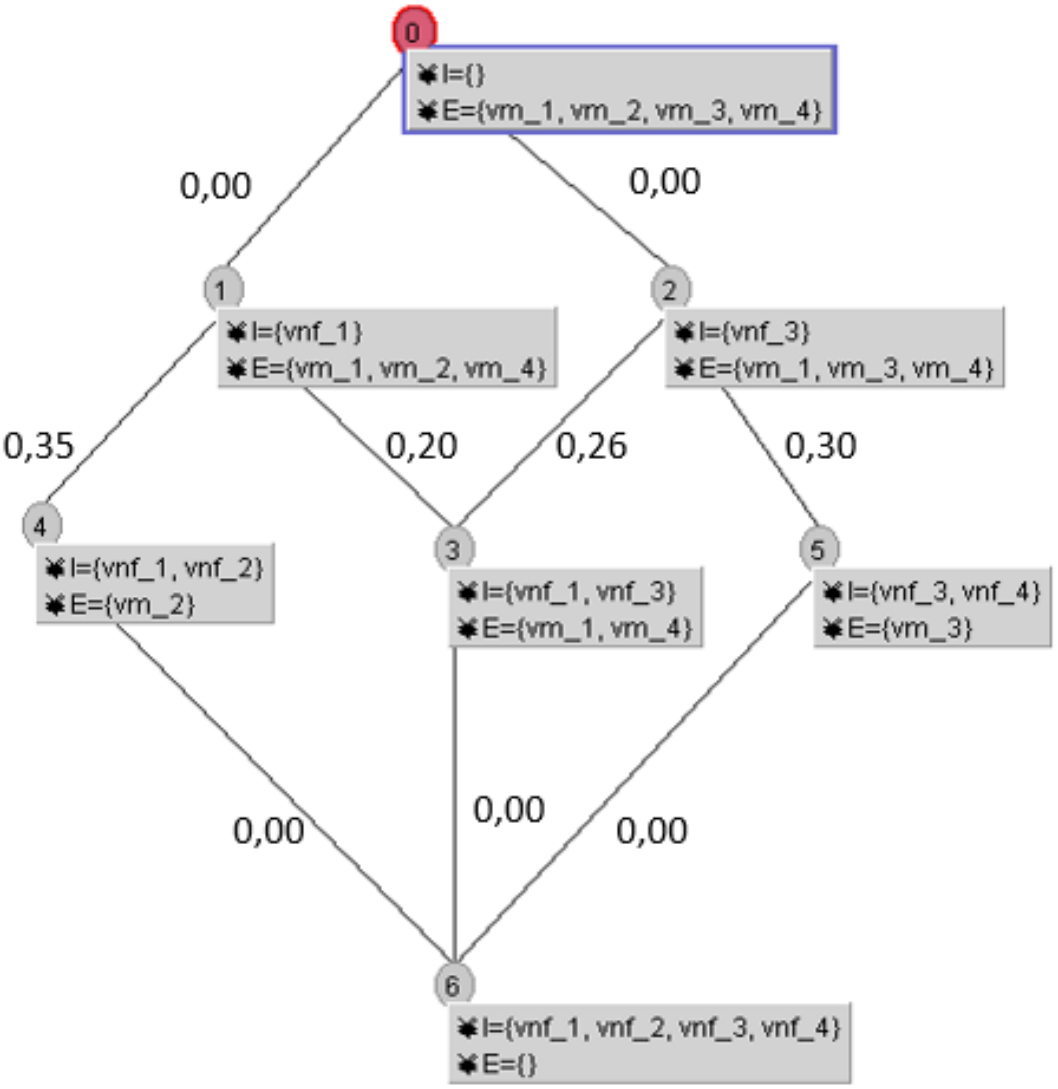

Figure 2.3: An example of fuzzy concept lattice of Fuzzy-FCA.

to host it. Our swarm intelligence-based Fuzzy-FCA is used to apply our VNF consolidation algorithm. Knowing that all set operators on Object Sets (VNF) are fuzzy [98] and those on Attribute Sets (VM) are net, both of which were previously extracted from the FCA. This algorithm's objective is to place a set of VNF into a minimum number of virtual machines in accordance with each VNF's individual choice while taking into account the operating balance of physical and virtual resources in all dimensions, latency, cost, and energy consumption. This algorithm creates conceptual clusters from a CS concept, known as the starting concept on a fuzzy concept lattice $F(K)$. We select CS as the supreme of $F(K)$, denoted $CS = sup(F(K))$, to generate all of the conceptual clusters of the fuzzy

concept network $F(K)$.

Our suggested algorithm can be written as shown in the following Algorithm 1.

Algorithm 1: Fuzzy-FCA-based VNF consolidation algorithm

Input: $VNF \leftarrow \{vnf_1, vnf_2, \ldots, vnf_i\}$ //set of virtual network functions
$VM \leftarrow \{vm_1, vm_2, \ldots, vm_j\}$ //set of virtual machines
$NS \leftarrow \{ns_1, ns_2, \ldots, ns_j\}$ //set of network service
F(K)//concept lattice
C_s // starting concept
T_s //threshold of similarity
Output: $Sc \leftarrow VM - VNF = \{\prec vm_i, vnf_j \succ\}$//set of solutions $\prec$ virtual machine, subset of virtual network functions $\succ$
begin
    // **Initialization**
    $VNF - NS \leftarrow$ empty
    $sc_j \leftarrow$ empty //subconcept VNF
    VNF = generate Fuzzy Formal Concepts ()
    Listweights = calculateweights (VNF)
    sort (VNF, Listweights)
    ListDouble Tendance
    for $i := 1$ *to numberVNF* **do**
        ListStimulus = calculateStimulus (VM)
        ListReponceIntrene = calculateInternalReponce (VNF (X), VM)
    end
    for *all* $vnf_i \in VNF$ **do**
        NS = generate Formal concept analysis()
        Listweights = calculate weights(VNF)
        Sort(NS, Listweights)
        listdouble tendance
        if *capacity of* $NS \succ VNF$ *requirements* **then**
            $VNF - NS.add(vnf_i, ns_j)$
            Find(scj.int)
        else
            return to 7
        end
        forall $vm_j \in VM$ **do**
            Tendance = calculateTrend (ListReponceIntrene, ListStimulus)
            Search_better_placement (Tendance, ListReponceIntrene, ListStimulus)
            for *subconcept C'of Cs in F(k)* **do**
                $F'(C') \leftarrow generateclusterconceptuel(C', F(k), Ts)$
                if $E(Cs, C') = |\frac{Cs \cup C'}{Cs \cap C}|$ **then**
                    $Sc \leftarrow Sc \cup F'(C')$
                else
                    add $F'(C)$ to $F'(k)$
                end
            end
        end
    end
end

2.3 Experimental study and results analysis

2.3.1 Data Collection

We will set up the test environments and working conditions before implementing our solution. A variable number of VM and VNF are included in our database so that we can analyze their impacts on cost, latency, and energy. We change the number of VMs and VNFs in each test with a difference of 10, while always keeping in mind that the number of VMs must always be lower than the number of VNFs, as shown in table 2.4.

Table 2.4: Sets of test data collections

Sets of tests	Number of VM	Number of VNF
S_1	90	110
S_2	100	120
S_3	110	130
S_4	120	140
S_5	130	150
S_6	140	160
S_7	150	170
S_8	160	180
S_9	170	190
S_{10}	180	200

These test bases are used by our graphical interface, which is required to generate resource requests for every VNF while taking into account the resource capacities of every VM. The requested resource values for each VNF in terms of memory, CPU and I/ O are chosen randomly in the range $[1000, 5000]$. While the resource capacity values for each virtual machine, CPU, and I/O are randomly selected from the range$[10,000, 50,000]$.

At the implementation level, we applied our solution of placement of VNF by the principle of Fuzzy-FCA based on intelligence in swarm, where each VNF is automatically migrated to the most appropriate VM on the basis of the received stimulus and its internal response. Based on the findings of our experiments, we can demonstrate the significance of our approach in placement ensuring resource use balance and optimizing energy consumption. We also demonstrate how using peripheral hosts (VM) reduces deployment latency and times. In terms of the

solution's quality and execution time, we contrast our outcome with that of the MultiSwarm approach [99].

2.3.2 Evaluation measures

We specify the following evaluation criterion in order to assess the effectiveness, efficiency, and scalability of our solution:
- The first criterion is packaging efficiency, which is demonstrated by the amount of a decrease in the number of active virtual machines and has an impact on overall energy consumption.
- The second criterion is how much energy is used by the server and network.
- The performance assessment is based on a comparison of the computational times of various solutions to a set of test problems with varying sizes and complexities.
- To determine efficiency, a test set of varying sizes and complexities is used to compare the computation times (convergence times) of various solutions.
- The scalability is evaluated in relation to the increase in its computation time and the size of the test problem.

While the VNF placement solution is considered effective and efficient if it meets the major constraints: minimal energy consumption, maximized and balanced use of resources in all dimensions, minimum latency and cost optimized.

2.3.2.1 Packaging Efficiency

The Packaging Efficiency, noted by $PacEff$, criterion reflected on other criteria such as energy consumption and unused resource. It is defined as the ratio of the number of vnf on the active number of vm as follow:

$$PacEff = \frac{Numberofvnf}{Numberofvm} \tag{2.14}$$

Therefore, this criterion can reflect on energy consumption, unused resource, cost and latency. So, the reduced packaging efficiency equals high energy consumption, cost, latency and unused resource.

2.3.2.2 Energy of server and energy of network

The server is among the major power consumer in the system. Formally, we define the power consumption pattern of VNF, $SerEne$, by the next equation:

$$SerEne = E_{CPU} + E_{memory} + E_{E/S} \tag{2.15}$$

where $SerEne$ denotes the total energy consumption in server; E_{CPU}, E_{memory}, $E_{E/S}$ represent the energy consumption of CPU, memory and input/output, respectively.

The network also consumes a significant amount of system power. Formally, we can evaluate the power consumption pattern of network as follows:

$$NetEne = E_{rack} + E_{network} \tag{2.16}$$

where $NetEne$ denotes the total energy consumption in network; E_{rack}, $E_{network}$ represent the energy consumption of rack and network, respectively.

2.3.2.3 Latency criterion

Our solution takes into consideration a mixed environment which consists of two types of hosts; cloud VM and MEC VM. In this case, the two types of hosts (VM) are grouped according to the latency value only without considering in which DC they belong. We consider $D = [d_1, ..., d_m]^T$ is a vector where d_j is the latency of the j_{th} VM and m is the total number of available VM across all DCs.

Let X is a specific assignment characterized by an n-dimensional latency vector $D' = [d_1, d'_2, ..., d'_n]$ whose value depends on the VM which are used to host the $nVNF$ composing the service, we define D' by the next formula:

$$D' = X.D \tag{2.17}$$

Therefore;

$$d'_i = \sum_{j=1}^{m} x_{ij} d_j \tag{2.18}$$

where d'_i is the latency of the VM which is hosting the i−th VNF (since a VNF is placed at exactly one VM, only a single term in the above sum will be non-zero). We define the average latency on all deployed VNF (average assignment latency X of elements D ') by:

$$L(x) = \frac{\sum_{i=1}^{n} d'_i}{n} \tag{2.19}$$

2.3.2.4 Cost criterion

We consider two types of cost; the fixed cost and overload cost of VM. The fixed cost consist of associated VM workload in terms of the number of vCPUs allocated to the VM and under the assumption that more application workload). The overload cost is the service/application requirement in terms of the number of virtual CPUs for VM_i. We also mention PM cost, which is a fixed overhead as the energy cost for keeping the physical machine in an operating state. So, X is the assignments of resources which can cost at the VM level where:

$$C_v(X) = e_v \sum_{i=1}^{n} p_i \tag{2.20}$$

with, p_i is the service/application requirement in terms of the number of virtual CPUs for VM_i.

2.3.3 Experimentation and results analysis

The experimental results are given on the basis of 10 tests, previously fixed. During these tests, we determine the evolution of the average latency, the cost of resources, the energy consumed at the server and network level, the exploitation of the used resources.

Figure 5 shows the variation in the number of active network services versus the total number of incoming network services to Cloud each time. The comparison figures between these two variations presented in Table 5 proves the effectiveness of the aggregation which is directed by FCA to reduce the number of active network service as well as to minimize the unused resources (UR) as shown in Figure 6. The curve of unused resources increases very slowly over the course of increase in

Table 2.5: Number of active NS compared to the total incoming NS number and Unused resources by our Fuzzy-FCA algorithm.

Sets of tests	Number of NS	Number of active NS	Unused resources
S_1	40	8	29.15
S_2	50	9	29.25
S_3	60	11	28.068
S_4	70	12	30.36
S_5	80	13	29.38
S_6	90	13	17.36
S_7	100	15	24.31
S_8	110	19	32.9
S_9	120	27	56.32
S_{10}	130	40	67.6

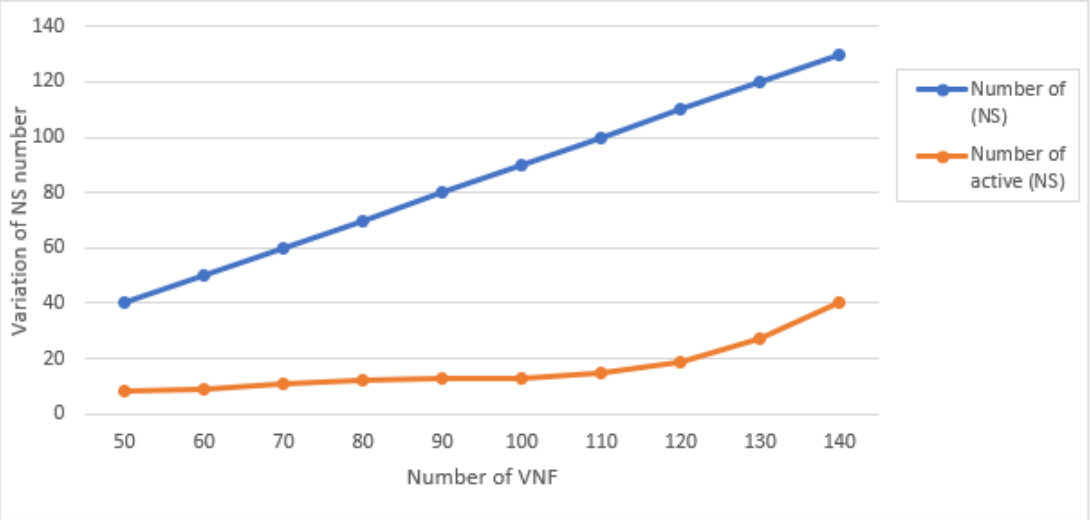

Figure 2.4: Number of active NS compared to the total incoming NS number by our Fuzzy-FCA algorithm.

number of incoming network service then it decreases by a very large value when the number of active NS equals 30.

Figure 7 shows a variation in average latency as a function of VM between the MEC environment and the cloud data center. As the figures in Table 7 also show, the average latency value decreases as the number of active virtual machines increases. This is explained by the increase in resources used, so the more VM and vCPUs increase, the more physical machines accelerate the latency time. Also, this is explained by the compute resources close to the end user deployed by MEC which can significantly reduce end-to-end latency.

1pt The decrease in latency with the increase in the number of virtual machines

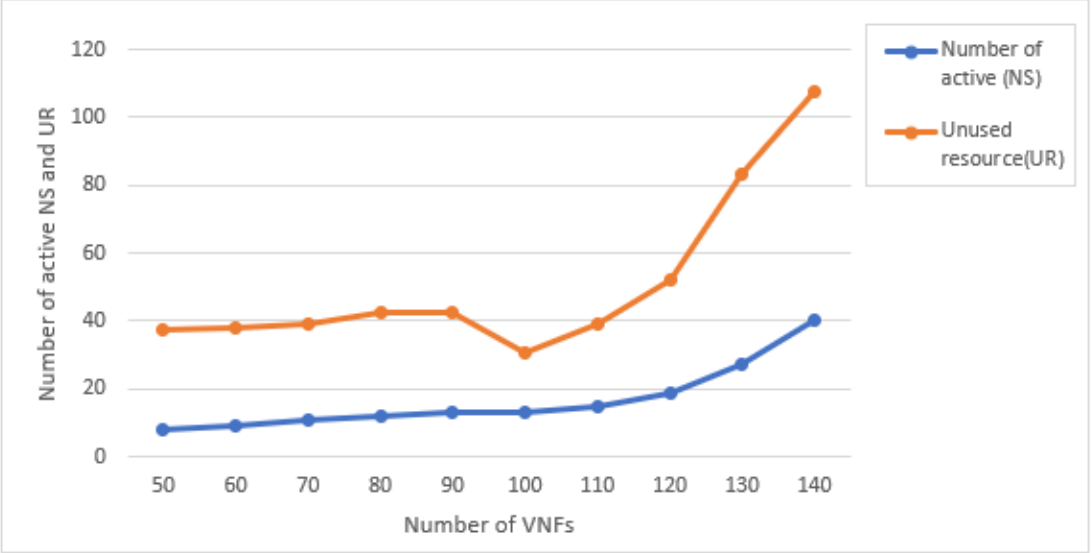

Figure 2.5: Evolution of unused resources by our Fuzzy-FCA algorithm.

Table 2.6: Average latency variation depending on number of VM by our Fuzzy-FCA algorithm.

Sets of tests	Number of VM	Average latency (ms)
S_1	8	0.125
S_2	9	0.111
S_3	12	0.083
S_4	15	0.066
S_5	17	0.058
S_6	18	0.055
S_7	20	0.050
S_8	21	0.047
S_9	22	0.045
S_{10}	28	0.035

can also be explained by the requirement of availability and flexibility, where the average latency does not exceed 0.043 ms for 22 virtual machines while it exceeds 0.12 ms for 8 virtual machines as mentioned in Figure 6. Here we can conclude that the decrease in latency during the increase in number of VM mainly returns to the sharing of the execution of the network functions on several virtual machines which decreases the overhead and the congestion at the level of a single Virtual machine.

Figure 8 illustrates the evolution of cost as a function of the number of VM considering the budget threshold indicated in Table 8, which also influences the result

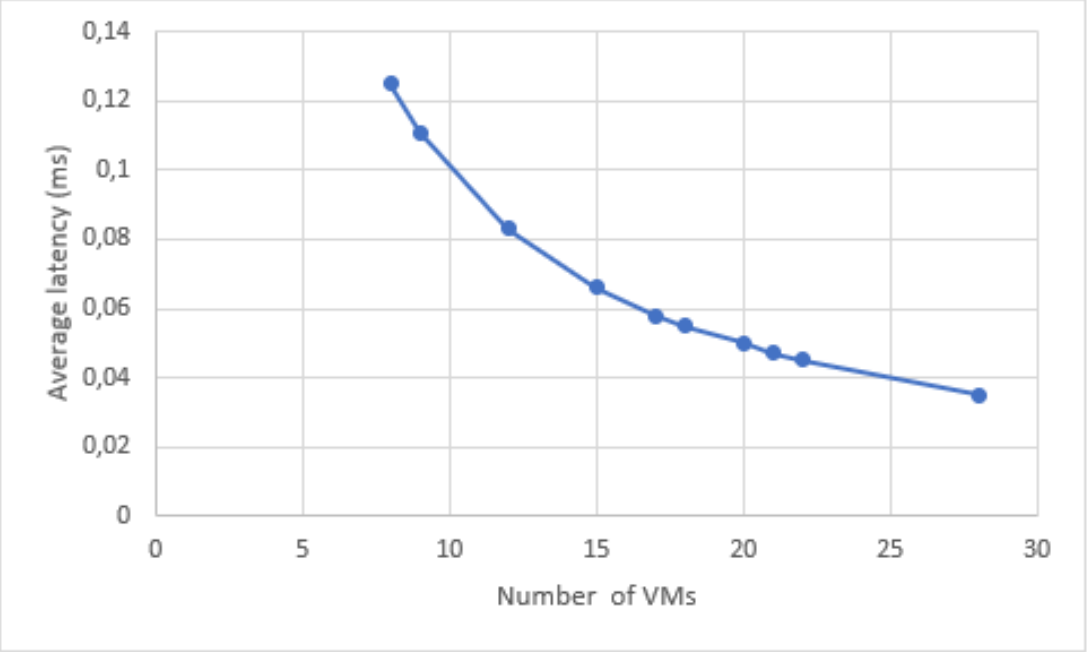

Figure 2.6: Average latency variation depending on number of VM by our Fuzzy-FCA algorithm.

Table 2.7: Cost optimization according to the number of VMs and the Cost in % of budget corresponding to 4 and 5 hundred dollars respectively given by our Fuzzy-FCA algorithm.

Sets of tests	Set of VM	Cost (in hundred dollars)	Cost in% of budget corresponding to 4 hundred dollars	Cost in% of budget corresponding to 5 hundred dollars
S_1	7	1.224924	31 %	24%
S_2	8	1.508356	38%	30%
S_3	12	2.407558	58%	47%
S_4	14	2.325032	60%	48%
S_5	17	2.891696	72%	58%
S_6	18	3.488612	87%	70%
S_7	21	4.157184	104%	83%
S_8	22	4.057636	101%	81%
S_9	24	4.818236	120%	96%
S_{10}	26	4.904336	163%	98%

of energy consumption as indicated in Figure 9. The coordinate axis represents the cost values as a part of the budget (which is the threshold equal 5, ie. 5 hundred dollars) and as a function of the number of virtual machines represented on the horizontal axis. The first and second straight line represents the maximum of budget 1 and budget 2 (4 hundred dollars and 5 hundred dollars) that can be fixed with the customer. For the first budget(4 hundred dollars), the cost increases

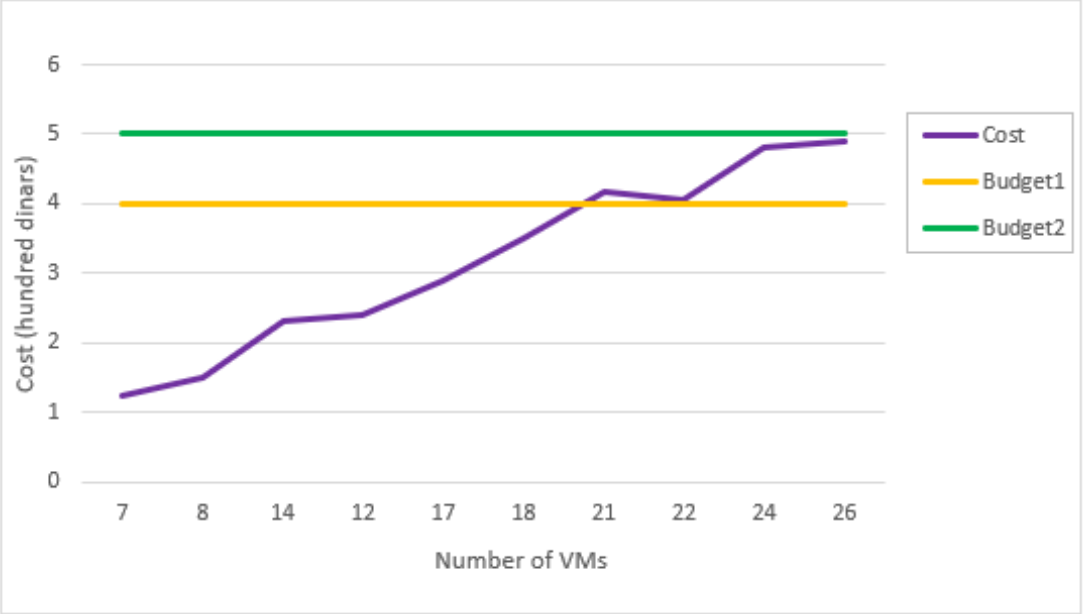

Figure 2.7: Cost optimization according to the number of VMs by our Fuzzy-FCA algorithm.

according to the number of virtual machines without exceeding the budget as in the case for 7 until 18 virtual machines, where the cost presents 98% of budget for 18 virtual machine. While the cost exceeds the budget when the number of VMs more than 18. For the budget 5 hundred dollars, the cost increase according to the number of VMs without exceed the budget. Although the number of VMs reaches the maximum (25), the cost does not exceed the budget.

Table 2.8: Comparison between Network Energy (NE) and Server Energy (SE) depending on the number of VMs by our Fuzzy-FCA algorithm.

Sets of tests	Number of VM	Network Energy (NE)	Server Energy (SE)	% (NE/SE)
S_1	9	4241	9222	45.99%
S_2	11	4577	12807	35.74%
S_3	12	4577	9495	48%
S_4	13	4577	14772	31%
S_5	14	4577	14772	27.57%
S_6	15	4577	18341	30.98%
S_7	16	4577	16599	24.95%
S_8	18	4577	20125	22.74%
S_9	19	4577	23795	20.73%
S_{10}	24	5249	15263	34%

Regarding the energy, Figure 9 and Table 8 show that the energy consumed at the network level is slightly increased with the increase in the number of virtual

machines while the energy consumed at the server level is increased remarkably with the increase in number of virtual machines. Therefore, our approach Fuzzy-FCA serves to minimize the number of virtual machines and to balance the use of resources by placing VNF to the most favorable virtual machines according to capacity, memory, vCPU, and so on. The basic idea is to consolidate VNF in the minimum number of virtual machines and put unused virtual machines in low power mode (standby) which decreases power consumption at data center and ensures balance in exploitation of resources. In our implementation, we set the number of VNF (60) and we vary the number of VM on each test to determine the number of active VM as shown in Table 9. With each test, we increase the number of VM and we see the reduction in number made by Fuzzy-FCA which shows the number of active virtual machines. Thus, according to the first test, the number of active VM is 14 with a reduction value equal to 6.

From Figure 10 we see that the packaging efficiency and unused resources evolve in terms of number of VM in a manner opposite to each other; thus when the packaging efficiency increases, the unused resources decrease and vice versa. In fact, the high packaging efficiency value reflects the maximization and the balance in exploitation of the resources and consequently the minimization of the losses. Figure 11 illustrates the difference between the simulation result of our Fuzzy-FCA proposal and the MultiSwarm approach in terms of average latency. Both average latency curves decrease with increasing number of VNF, but the latency

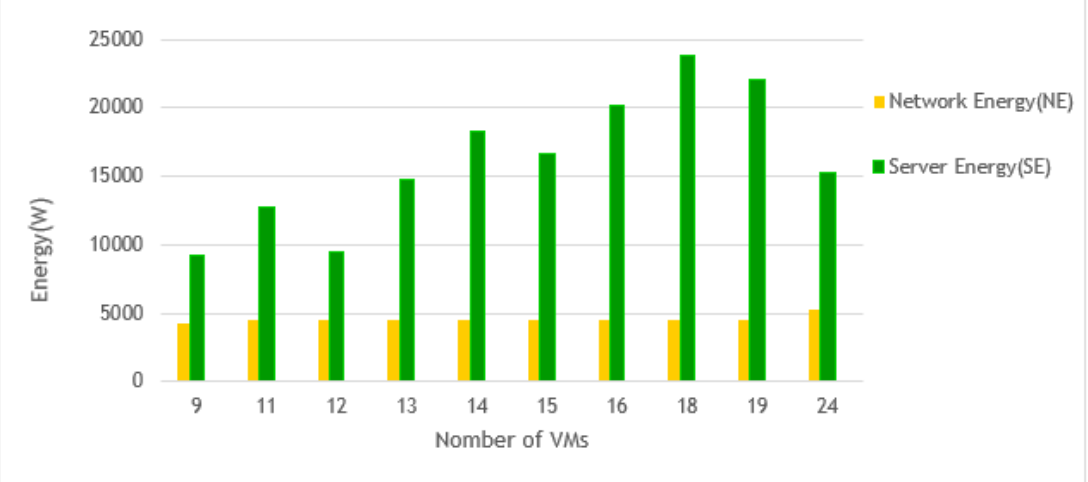

Figure 2.8: Comparison between Network Energy (NE) and Server Energy (SE) depending on the number of VMs by our Fuzzy-FCA algorithm.

of our approach (Fuzzy-FCA) proposal is still higher than that of MultiSwarm. The average latency of Fuzzy-FCA and MultiSwarm tends towards 0 when the number of VNF exceeds 200 which explains by the need at the most less response time (availability) to respond to the most number of VNF while exceeds 0.1 ms for MultiSwarm and 0.25 ms for our Fuzzy-FCA solution.

Figure 12 shows the difference between the execution time of our Fuzzy-FCA and MultiSwarm algorithm as a function of increasing number of virtual machines in

Table 2.9: Packaging Efficiency (PE) versus value of Unused Resources(UR) by our Fuzzy-FCA algorithm.

Sets of tests	Sets of VNFs	Total number of VM	Number of active VM	Packaging Efficiency (PE)	Unused Resources (UR)
S_1	50	40	8	7.14	29.15
S_2	60	50	9	6.66	29.25
S_3	70	60	11	6.36	28.068
S_4	80	70	12	6.66	30.36
S_5	90	80	13	6.92	29.38
S_6	100	90	13	7.69	17.36
S_7	110	100	15	7.33	24.31
S_8	120	110	19	6.31	32.90
S_9	130	120	27	4.31	56.32
S_{10}	140	130	30	4.66	58.08

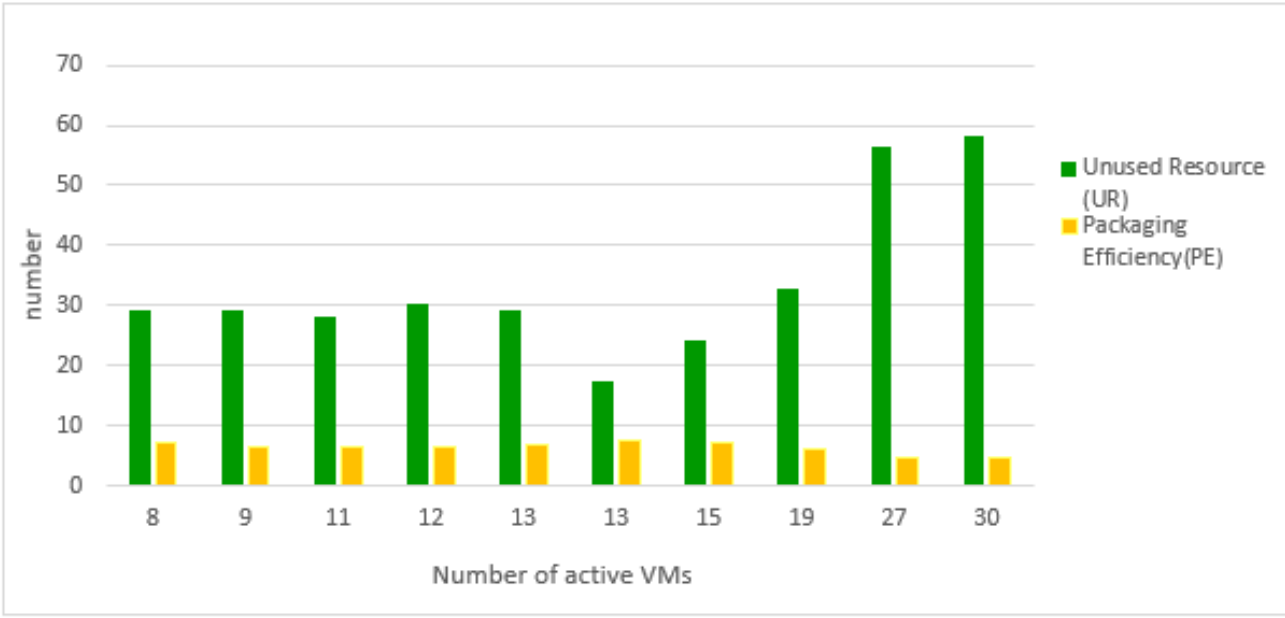

Figure 2.9: Packaging Efficiency (PE) versus value of Unused Resources(UR) by our Fuzzy-FCA algorithm.

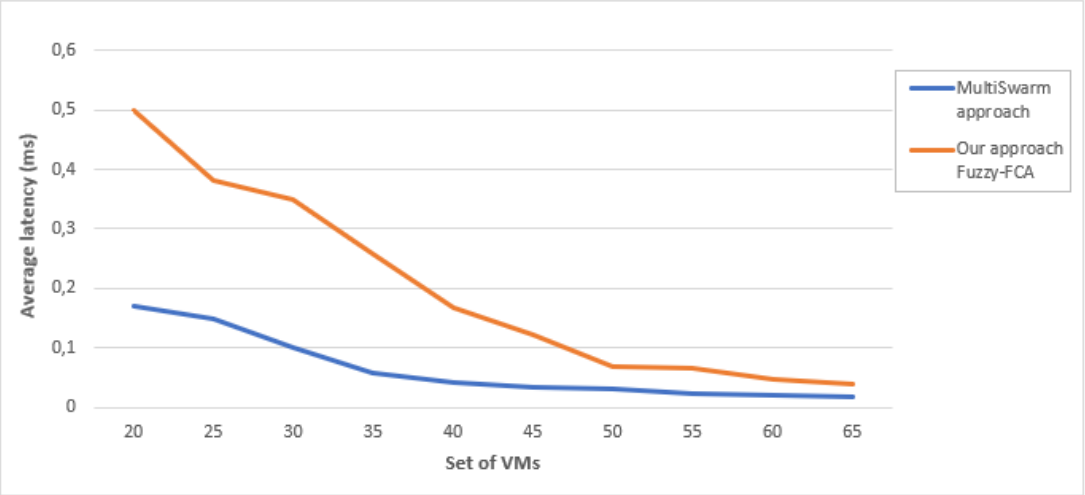

Figure 2.10: Latency depending on the number of VM for Fuzzy-FCA and MultiSwarm approach.

use. These experiments were performed on an Intel® Celeron® N4000 and 4 GB RAM machine (DELL Inspiron 15300). The curves show that the execution time of our approach (Fuzzy-FCA) is always lower than the MultiSwarm approach and increases with the increase in the number of virtual machines. It can be seen that there is a fluctuation in the Fuzzy-FCA curve explained by other internal factors (e.g network services).

Table 2.10: Variation of Average latency and time for Fuzzy-FCA and MultiSwarm algorithm.

		Algorithm			
		Fuzzy-FCA		MultiSwarm	
Sets of tests	Set of VM	Average latency(ms)	Time(ms)	Average latency(ms)	Time(ms)
S_1	20	0.33	3.3	0.17	3.5
S_2	25	0.23	3.35	0.15	3.55
S_3	30	0.25	3.14	0.1	3.57
S_4	35	0.2	2.96	0.059	3.58
S_5	40	0.125	3	0.042	3.6
S_6	45	0.09	3.04	0.033	3.65
S_7	50	0.038	3.019	0.03	3.67
S_8	55	0.041	3.1	0.024	3.69
S_9	60	0.027	2.9	0.021	3.7
S_{10}	65	0.02	3.06	0.018	3.75

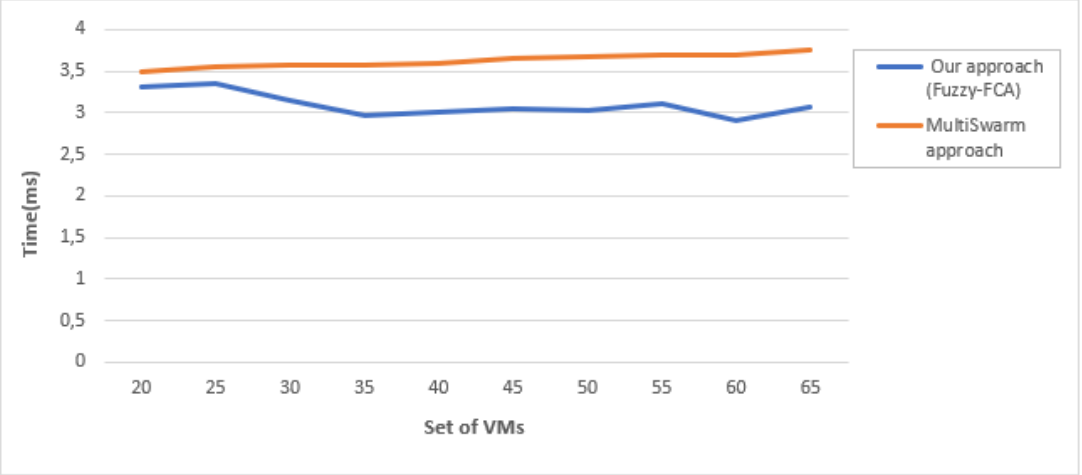

Figure 2.11: Execution time depending on the number of VMs for our approach Fuzzy-FCA and MultiSwarm algorithm.

2.4 Discussion

The performance and efficiency of our approach is well illustrated in the Table 10 in comparison with MultiSwarm algorithm. This efficiency shows at two parameters; the latency and time execution. Also, we used other three performance parameters based on our objective such as the maximization in exploitation of resources which is shown from PE, the balance in exploitation of resources from UR, and energy consumption optimization. The results of our approach confirm the efficiency and importance of FCA and Fuzzy-FCA in solving the problem of VNF placement compared to the MultiSwarm algorithm. Indeed, the MultiSwarm approach is based on best position taking into account the speed. Our virtual network function placement algorithm was able to guarantee a better reduction in power consumption while ensuring a low number of active virtual machines for different test problems. While the MultiSwarm algorithm mainly focuses on finding a specific region from the best swarm speed to place virtual network functions. So, we cannot talk about the reduction of the number of active machines and about the optimization of energy consumption in the MultiSwarm algorithm.

2.5 Strengths and limits of our approach

From the obtained results, we conclude that our approach succeeded in achieving the optimization of the placement of VNF by considering the balance and the maximization in exploitation of resources, the latency time, the cost and the consumed energy. However, these results may not be absolutely ideal since they are slightly bounded due to the uncertainty context. Most of the results show the importance of our proposed approach in reducing the number of active virtual machines and thus minimizing resource consumption, cost, latency and power consumption. Also, these results reflect the role of the confidence threshold value in specifying the capacity of each virtual machine from the packaging efficiency based on the uncertainty information. Also, the evolution of latency confirms the role of the integration of MEC with the Cloud environment as part of our approach. But, on the other hand, the fluctuations in some results show that our approach cannot achieve a perfect result from all sides considering the uncertainty information and the internal factors of the system. These fluctuations is clearly shown in Figure 6 such that the number of (UR) increases throughout the tests but it decreases for test 6 when VNF equals 100 from 29.38 to 17.36, then it comes back to increase later. Despite the importance and the multiplicity of possibility theorem, the uncertainty problem remains a great obstacle to achieve ideal results.

2.6 Conclusion and future work

In this chapter, we have applied the principle of FCA grouping by invoking the principle of labor division in swarm intelligence in a cloud environment. This approach is dedicated as a new solution for the placement of virtual network functions that takes into account the energy consumption in the Edge and cloud data centers, with minimized latency and the maximum and balanced exploitation of resources (memory, CPU, I/ O). We performed a series of tests to verify the performance and correspondence of our proposed algorithm with the sought conditions. The experimental results show that our virtual network function placement algorithm performed better in terms of our objective compared to the used comparison MultiSwarm algorithm. It guarantees the minimization of the number of active virtual machines for the allocation of VNF and the quantity of unused resources, gives

high packing efficiency and reliable computation time. The experimental results also show that our solution is based on a confidence value which gives a very efficient restriction allowing to minimize the number of concept, hence the number of active virtual machine.

The future perspectives for this work can be articulated around two new directions. The first direction is to conduct a more in-depth comparative study that can give academics and practitioners more knowledge on how to handle Virtual Network Functions Placement and Consolidation in Cloud Data Centers. The second direction is to extend this work in terms of VNF chaining criterion in relation to placement and with two important objectives to avoid: (i) the communication interference between different VNF and (ii) the congestion due to sharing the same resources.

Chapter 3

VNF Placement and Consolidation in Chain to Deploy SFC

3.1 Introduction

As the placement of VNF in SFC is in predefined order, this chapter represents a continuation of the problem of consolidation and placement of VNF taking into account the new conditions and requirements of the structure of SFC. The objective is to find the best VNF placement and guarantee the shortest SFC path taking into account the multiplicity of VNF instances. To avoid inefficient consolidation of VNF that does not meet the predefined order of SFC and host requirements, we propose a new method of placing VNF in chained VM to minimize end-to-end latency, minimize throughput and cost. In the following section, we treat and formulate the problematic. Then, we present in section 3.3 our model and proposal to solve the problem. In section 3.4, we describe the NFV model and system. In the next section 3.5, we use and present the training procedure which is based on PG policy gradient. In section 3.6, we present experimental results of simulations that help measure the performance of our proposal. Section 3.7 is devoted to conclusion.

3.2 Virtual Network Function (VNF) placement and chaining problem

The notion of Service Function Chain (SFC) adds new challenges to the VNF placement problem as long as it is made up of a set of VNF deployed in a predefined order. These challenges divide in two; the first one consists in finding the best placement of VNF taking into account multiple requirements (VM capacity, CPU, memory, instance), and the second is to select suitable paths which connects VNF in a predefined order while respecting routing requirements (bandwidth, delay, etc). Each SFC provides a network service (NF) according to a specific order of VNFs. For example the security flow can be ensured by the use of SFC which is constituted by the VNFs according to the following order: Firewall, DPI, and VPN. The deployment of this service is provided via commodity servers along path connecting source to destination in an infrastructure equipped with switches and routers. Figure 3.1 shows an example of placement and chaining of a set of VNFs in an NFV environment. Each SFC request is bounded by a source (S) and destination (D) which represent the physical locations on the corresponding NFV infrastructure for the flow source and target. Each VNF should be placed in the most suitable VM node considering the available resources. In Figure 3.1, VNF3 is placed in the VM3 along the path connecting VM7 which embeds the source S and VM8 embeds the destination D (red lines).

Some links can be overloaded and present a risk of congestion especially when they are used by several sessions. For example, the link (3,6) is shared by two sessions (session S1 and session S2) which overloads it. This risk of congestion can take place at the host node level when its resource capacities are insufficient. In this case, an effective VNF placement and chaining algorithm can solve these issues by finding the best VNF placement and traffic path. The main objective is to minimize the cost of resources, reduce the end-to-end delay (avoiding traffic congestion) and improve throughput. Generally, the problem of VNF placement and chain is a unification between the problem of placement and chaining in order to ensure a suitable location for the VNFs without affecting the predefined order of the set of VNFs constituting the SFC.

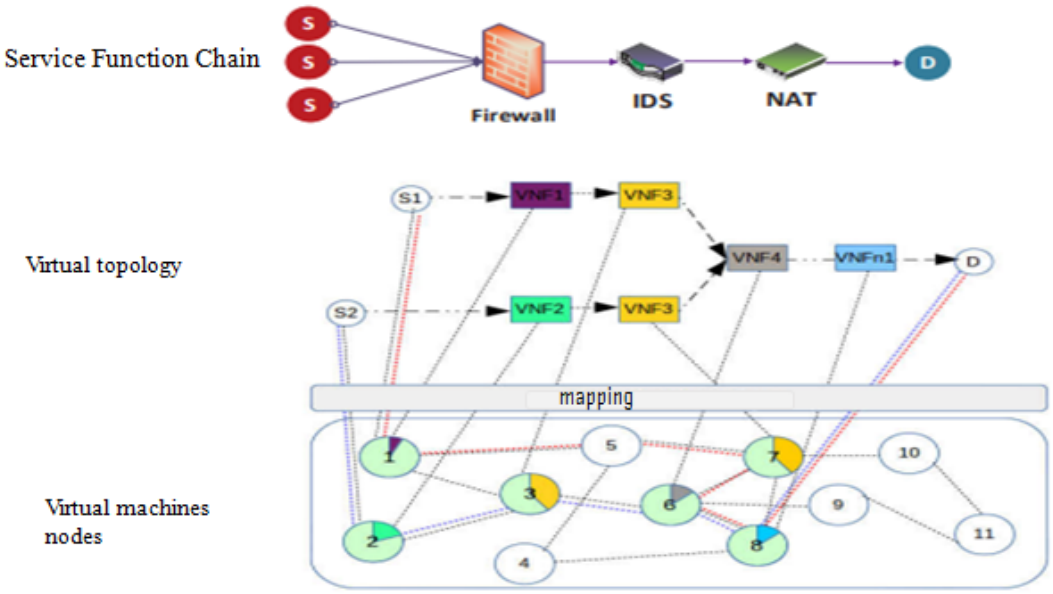

Figure 3.1: SFC request acceptance ratio in our approach

3.3 Model and problem formulation

To solve the problem of VNF placement and chaining, we propose an ILP modeling that describes the relationship between VNF placement and chaining order in VMs.

3.3.1 Model

Let $G_s = (\mathbb{V}_s, \mathbb{E}_s)$ be a physical infrastructure where $\mathbb{V}_s$ and $\mathbb{E}_s$ represent respectively the cluster of VMs (hosts or points of presence POPs) and the cluster of physical links connecting them. Each VM $u \in \mathbb{V}_s$ is equiped with various types of resources $R = \{1, 2, 3\}$ (memory, CPU and storage). For each type of resource $r \in R$, the associated capacity to VM u is denoted by c_u^r. Similarly, each physical link l_s is associated with a bandwidth capacity of b_{l_s} and a propagation delay of d_{l_s}. Each VM resource r is associated with a unit cost w_u^r and each physical link resource is associated with a unit cost of $w_{l_s}^{bw}$. The propagation delay on link l_s is denoted by d_{l_s} whereas the switching delay on VM u is denoted by d_u.
In the physical infrastructure, each couple of VMs $(u, v) \in {\mathbb{V}_s}^2$ are interconnected with the use of various paths $p_s \in \mathbb{P}$. The extremity nodes of path p_s are denoted by p_s^a and p_s^b.

Let $G_i = (\mathbb{V}_i, \mathbb{E}_i)$ be the required service function chain $i \in \mathbb{I}$, where $\mathbb{V}_i$ is the collection of VNFs and $\mathbb{E}_i$ is the collection of virtual links that connect them. $l_i = l_i^a l_i^b \in \mathbb{E}_i$ represents a virtual link that interconnects its extremity VNFs l_i^a and l_i^b. Besides, each *VNF* $V \in \mathbb{V}_i$ has a resource requirement $C_V^{r,\,i}$ and each virtual link $l_i \in E_i$ has a bandwidth requirement B_{l_i}. The flow processing time on each VNF V is denoted by d_V.

The end-to-end delay requirement for each service function chain i is given and denoted by d^i. In addition, the end nodes V_{s_i} and V_{t_i} of SFC i are mapped on the physical nodes u_{s_i} and u_{t_i}.

3.3.2 Problem formulation

Based on the notions listed in Table 1, we formulate the placement and chaining problem of VNFs considering all the network service requirements and constraints as follows:

Constraints:

$$\sum_{i \in \mathbb{I}} \sum_{V \in \mathbb{V}_i} x^i_{V,\,u} \times C_V^{r,\,i} \leq c^r_u \quad \forall u \in \mathbb{V}_s, r \in R \tag{3.1}$$

$$\sum_{u \in \mathbb{V}_s} x^i_{V,\,u} - M_i = 0 \quad \forall i \in \mathbb{I}, V \in \mathbb{V}_i \tag{3.2}$$

$$x^i_{V_{s_i},\,u_{s_i}} - M_i = 0 \quad x^i_{V_{t_i},\,u_{t_i}} - M_i = 0 \quad \forall i \in \mathbb{I} \tag{3.3}$$

$$\sum_{i \in \mathbb{I}} \sum_{l_i \in \mathbb{E}_i} \sum_{p_s \in \mathbb{P}} o^{l_s}_{p_s} \times B_{l_i} \times y^{l_i}_{p_s} \leq b_{l_s} \quad \forall l_s \in \mathbb{E}_s \tag{3.4}$$

$$\sum_{p_s \in \mathbb{P}} y^{l_i}_{p_s} - M_i \leq 0 \quad \forall i \in \mathbb{I}, l_i \in \mathbb{E}_i \tag{3.5}$$

$$x^i_{l_i^a,\,p_s^a} + x^i_{l_i^b,\,p_s^b} - y^{l_i}_{p_s} \leq 1 \quad \forall i \in \mathbb{I}, l_i \in \mathbb{E}_i, p_s \in \mathbb{P} \tag{3.6}$$

$$x^i_{l_i^a,\,p_s^b} + x^i_{l_i^b,\,p_s^a} - y^{l_i}_{p_s} \leq 1 \quad \forall i \in \mathbb{I}, l_i \in \mathbb{E}_i, p_s \in \mathbb{P} \tag{3.7}$$

$$\sum_{l_s \in \mathbb{E}_s} \sum_{l_i \in \mathbb{E}_i} \sum_{p_s \in \mathbb{P}} o_{p_s}^{l_s} \times d_{l_s} \times y_{p_s}^{l_i} + \sum_{u \in \mathbb{V}_s} \sum_{l_i \in \mathbb{E}_i} \sum_{p_s \in \mathbb{P}} o_{p_s}^{u} \times d_u \times y_{p_s}^{l_i} + \sum_{u \in \mathbb{V}_s} \sum_{V \in \mathbb{V}_i} d_V \times x_{V,\,u}^{i} \leq d^i \tag{3.8}$$

Objective:

$$min(\epsilon_1 A + \epsilon_2 B + \epsilon_3 C - \sum_{i \in I} M_i) \tag{3.9}$$

where: $\epsilon_1 + \epsilon_2 + \epsilon_3 << 1$

$$A = \sum_{i \in \mathbb{I}} \sum_{l_s \in \mathbb{E}_s} \sum_{l_i \in \mathbb{E}_i} \sum_{p_s \in \mathbb{P}} o_{p_s}^{l_s} \times d_{l_s} \times y_{p_s}^{l_i} + \sum_{i \in \mathbb{I}} \sum_{u \in \mathbb{V}_s} \sum_{l_i \in \mathbb{E}_i} \sum_{p_s \in \mathbb{P}} o_{p_s}^{u} \times d_u \times y_{p_s}^{l_i} + \sum_{i \in \mathbb{I}} \sum_{u \in \mathbb{V}_s} \sum_{V \in \mathbb{V}_i} d_V \times x_{V,\,u}^{i}$$

$$B = \sum_{i \in \mathbb{I}} \sum_{l_i \in \mathbb{E}_i} \sum_{p_s \in \mathbb{P}} \sum_{l_s \in \mathbb{E}_s} \frac{o_{p_s}^{l_s} \times B_{l_i} \times y_{p_s}^{l_i}}{b_{l_s}}$$

$$C = \sum_{i \in \mathbb{I}} \sum_{r \in R} \sum_{V \in \mathbb{V}_i} \sum_{u \in \mathbb{V}_s} w_u^r \times C_V^{r,\,i} \times x_{V,\,u}^{i} + \sum_{i \in \mathbb{I}} \sum_{l_i \in \mathbb{E}_i} \sum_{p_s \in \mathbb{P}} \sum_{l_s \in \mathbb{E}_s} w_{l_s}^{bw} \times o_{p_s}^{l_s} \times B_{l_i} \times y_{p_s}^{l_i}$$

The objective function maximizes the number of SFC requests in $\mathbb{I}$ to place and chain. Among the possible solutions, the objective function will choose the solution which optimizes the second part of the objective, namely a combination of (1) the delay, (2) the ratio of the bandwidth requested by the residual bandwidth on the links and (3) the cost of the resources. ϵ_1, ϵ_2 and ϵ_3 (very small values) make it possible to reduce or favor one or the other of the metrics compared to the others. For example, canceling the values of ϵ_2 and ϵ_3 results in minimizing the delay while canceling ϵ_1 and ϵ_3 will allow load sharing to avoid congestion of the physical infrastructure.

The constraints (3.1) and (3.4) guarantee that the resource demands of nodes and links must not exceed the resource capacities of nodes and physical links. The constraint (3.2) indicates that for any SFC placed, each of its VNFs must be placed on a single physical node to prevent the multiplicity of VNF instance. If the SFC i is placed, its sources and destinations must also be placed on given physical nodes fixed in advance (constraint (3.3)). The constraint (3.5) ensures that a virtual link can only be mapped to at most one physical path. Clearly, when both ends of a virtual link are mapped to the same physical node, the virtual link will be mapped to an empty path. Otherwise, the virtual link will be mapped to a non-empty physical path. The constraints (3.6) and (3.7) set the value of the variable $y_{p_s}^{l_i}$ according to the physical nodes on which the ends of the virtual link l_i are mapped: for different physical nodes, $y_{p_s}^{l_i}$ will be set to 1, otherwise $y_{p_s}^{l_i}$

will be zero. The inequality (3.8) is used to check the delay constraint: the sum of the propagation, switching and processing delays on the VNFs must be less than the value d^i provided as an input to the problem.

3.4 Model and NFV system description

Our model is mainly built on the NFV infrastructure and SFC requests. In this work, we try to meet the requirements of each SFC request by considering the different constraints mentioned in the previous section (latency, VM node capacity, bandwidth, and VNF instance) using Deep Reinforcement Learning (DRL). The NFV environment provides the essential infrastructure to build the Deep Neural Network (DNN) to represent the SFC service function chain topology in a set of VNFs as shown in Figure 3.2. Each incoming SFC request is mapped to a set of VNFs in a predefined order. Each VNF is placed in the most suitable server node according to its requirements. These nodes are connected via multilevel switches. In this case, the network of our model can be presented by $G = (N, L)$, where N is the set of server nodes and L is the set of links. With DRL methodology, the DNN corresponds to agent and PG. This agent extracts the NFV environment state and processes it, then makes the decision to accept and take the action of placing VNF in their corresponding node or to refuse it. If accepted, the NFV environment sends a reward to the agent, and the agent updates the policies according to the reward. This approach is characterized by the ability to adapt to the dynamicity of the environment. It is based on the MDP and LSTM model which captures dynamic network state transitions and sends them to the system agent for processing. DRL integrates PG policy to facilitate decision-making for high-dimension MDP states whereas historical states are guaranteed by LSTM.

3.4.1 Parallel Bi-state Module Deep Reinforcement Learning approach for VNF placement and SFC deployment

To summarize, this work presents a demonstration of a new approach for deep reinforcement (DRL) based on parallel two modules of MDP to extract and capture the dynamic state transitions of SFC and LSTM to detect the long-term historical trend of

Table 3.1:

Parameters, variables and constants

Network service physical infrastructure	**Description**
$G = (\mathbb{V}_s, \mathbb{E}_s)$	G is a graph representing the physical infrastructure, $\mathbb{V}_s$ is the set of hosts (VMs or POPs) and $\mathbb{E}_s$ corresponds to the set of physical links.
$\mathbb{P}$	$\mathbb{P}$ corresponds to a set of paths in the physical infrastructure. The paths $p_s \in \mathbb{P}$ interconnect 2 nodes in the physical infrastructure.
(p_s^a, p_s^b)	p_s^a and p_s^b correspond to the extremity nodes o path p_s.
R	$R = \{1, 2, 3\}$ is the set of resources. Here, we assumed 3 types of resources: memory, CPU and storage.
c_u^r	c_u^r corresponds to the capacity in resource $r \in R$ for the physical node $u \in \mathbb{V}_s$.
d_u, d_{l_s}	d_u and d_{l_s} correspond respectively to the switching time of the node u and to the propagation delay on the physical link l_s.
w_u^r, $w_{l_s}^{bw}$	w_u^r and $w_{l_s}^{bw}$ denote respectively the unit cost of resource r on the node u and the cost of 1 unit of bandwidth on link l_s.
$o_{p_s}^{l_s}$	$o_{p_s}^{l_s}$ is a constant that is set to 1 for physical link l_s belonging to the physical path p_s. $o_{p_s}^{l_s}$ is nil for $l_s \notin p_s$.
u_{s_i}, u_{t_i}	u_{s_i} and u_{t_i} correspond respectively to the physical nodes which embed the source and target of SFC i.
Parameters of service function chain request	**Description**
$G_i = (\mathbb{V}_i, \mathbb{E}_i)$	G_i is the requested service chain $i \in \mathbb{I}$ where $\mathbb{V}_i$ is the set of VNFs and $\mathbb{E}_i$ is the set of virtual links.
$\mathbb{I}$	$\mathbb{I}$ is the set of SFCs i arriving during the considered slot time τ.
d^i	d^i is the end-to-end delay requirement for the service function chain i
$C_V^{r,\,i}$, B_{l_i}	$C_V^{r,\,i}$ and B_{l_i} correspond respectively to the demand of SFC i for resource r on the VNF node V and to the demand for bandwidth on the virtual link l_i.
V_{s_i}, V_{t_i}	V_{s_i} and V_{t_i} correspond respectively to the source and target nodes of SFC i.
Binary variable	**Description**
M_i	M_i is set to 1 when SFC i is successfully embeded. Otherwise, M_i is nil.
$x_{V,\,u}^i$	$x_{V,\,u}^i$ is a binary variable indicating whether or not VNF V of SFC i is embedded into physical node u.
$y_{p_s}^{l_i}$	$y_{p_s}^{l_i}$ is set to 1 if the virtual link l_i is embedded into the physical path p_s. Otherwise, $y_{p_s}^{l_i}$ is nil.

SFC. Then, the Policy Gradient (PG) is used in the training stage to train the VNF environment agent and optimize reward.

Generally, our approach represents an adaptive VNF architecture based on PG that consists of a multi-layer fully connected DNNQ based on the back-propagation (BP) network as shown in Figure 3.3. DRL uses the PBDRL architecture which mainly consists of the NFV network as an environment including the nodes (servers or VM) and links between nodes in the network topology, the deep neural network as PBDRL agent, state information that combines the current state transitions and historical features of SFC, the action of the agent, reward, and policy. These components are shown in Figure 3.3 and work as follow:

- Environment provides space work for agents through which can move. The current state of the agent and action represent the input of the environment, whereas the agent's reward and its next state represent the output.
- Agent takes its current state from the environment and selects an action from a list of discrete and possible actions as a return.
- The environment measures the success or failure of an action. The reward is transferred to the agent if the action conditions are successful or backtracked in case of failure.
- Finally, the agent uses the policy according to reward to determine the next action. This procedure is repeated until reward converges.

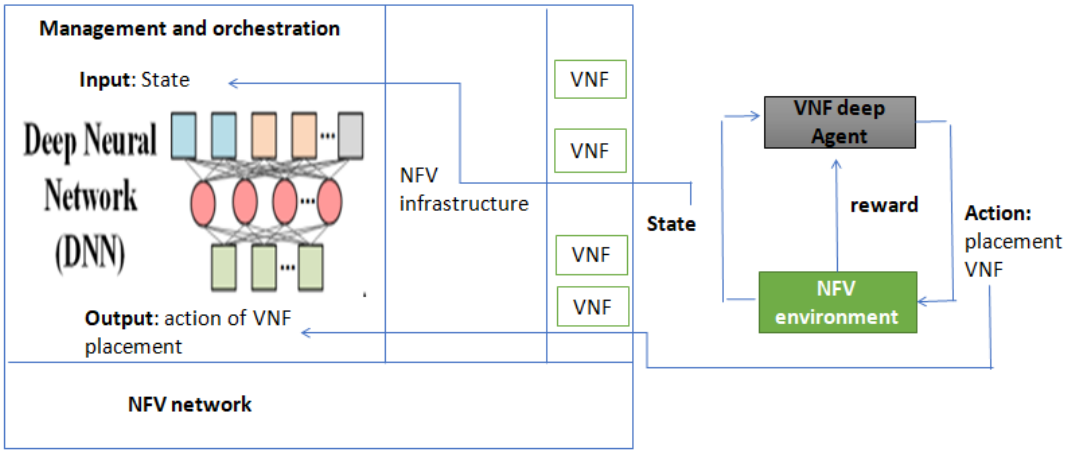

Figure 3.2: Adaptive NFV/SFC environment architecture with DRL

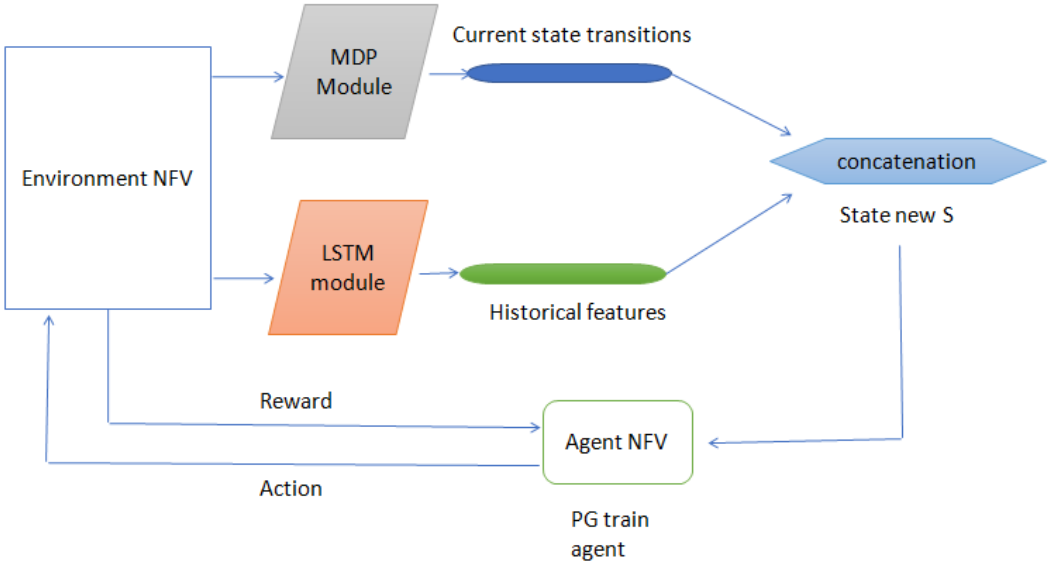

Figure 3.3: The architecture of PBDRL proposed approach

3.4.2 MDP and LSTM modules

To extract instantaneous (in real-time) network characteristics resulting from the probabilistic entry and exit of requests, we formally present the MDP model. We also use the LSTM model to support the data extracted by historical features. The MDP model is described by $\prec A, P, R, \gamma \succ$, where S is the set of discrete states, A is the set of discrete actions, $P : S \times A \times S$ is the transition probability distribution, $R : S \times A$ is the reward function, and $\gamma \in [0, 1]$ defines the discount factor that specific for future reward. Mainly, MDP aims to extract and capture the dynamic state moves of the network that can be expressed by the current state and the transition state as $S_{transition} = (s_t, a_t, r_t, s_{t+1})$, s_t is the current network state, a_t is the VNF placement action for the requested service r_t, and s_{t+1} is the new state of the network. The LSTM model aims at extracting the historical features of SFC placement which is defined as $S_{history} = LSTM(S3)$. This state is concatenated with the MDP state giving a new state: $S_t^{new} = \{S_{current}, S_{transition}, S_{history}\}$. Then the agent takes the characteristics of this state and processes it to return an action to the environment. With these various details of the environment's current state, the agent can find the appropriate policies through reciprocal and continuous exchange with the environment.

The PBDRL pseudo-code of our proposed approach which expresses these interactions is shown in Algorithm 1.

3.4.3 Description of MDP module

The model of our approach is modeled mathematically as a markovian decision process (MDP) by the following three components:

State: state capture is the most important element for extracting the characteristics of the NFV environment. Based on this component, we can determine the inputs of our approach as follows: let a state $s_t \in S$ be presented as vector (C_t, W_t, I_t): $C_t^s = (C_1^t, C_2^t, ..C_{|V|^t})$ represents the residual resources for each node, while $W_t = (W_1^t, W_2^t, ..W_V^t)$ is the remainder of the output bandwidth. $I = (W_{r_i}, T_{r_i}^j, N_{r_i}^j, C_{f_{i,j}}, P_{r_i})$ uncovers the properties of the current VNF in processing $f_{i,j}$, which includes W_{r_i} as the requested bandwidth, T as the residual latency space, $N_{r,i}^j$ as the number of undistributed VNFs in r_i, $C_{f_{i,j}}$ as request resources on servers, and P_r as TTL of request r_i.

Bi-state transition: $S_t^{new} = (S_{current} S_{history}, h_t)$ is the state transition of MDP and LSTM module, where $S_{current} = (s_t, a_t, r_t, s_{t+1})$, $S_{history} = \{P_{t,i}^k\}_{i=0}^n$ and h_t is the historical placement sequences of the VNF and related SFCs. We note that n is the overall number of VNF, $P_{t,i}^k$ is the placement sequence of VNF i with the look-back period k at time t and a_t is the action of VNF placement for responding to service requirements.

Action: In our proposal work, the action is the placement of VNF in the VM node for the deployment of the SFC requested. By associating to each VM node an integer setting $k = 1, 2, 3.., |N|$, an action on a given VNF corresponds to the setting of embedding VM. More precisely, if a VNF cannot be deployed, $a = 0$ otherwise the action a corresponds to the setting of an embedding VM. The set of actions A is thus determined as $A = \{0, 1, .., N\}$.

Reward action: the reward function represents the total expected cost of occupied VMs (the expense) to deploy the incoming request.

Algorithm 2: Parallel Bi-state Module Deep Reinforcement Learning (PBDRL) algorithm

Data: Input: Environment NFV
Result: Policy

Initialization;
initiate time slot $\tau \leftarrow 1$
while $R_\tau = \emptyset$ **do**
  $\tau \leftarrow \tau + 1$
end
use a request $r1$ *from* $R\tau$ *based on the arrival time*
$i \leftarrow 1$, $j \leftarrow 1$
for $t \leftarrow 1,\ T$ **do**
  Initialize state $S_{current}$ *and* $S_{history}$
  Concatenate states and get the new state
  Perform an action a from A to place $f_{i,j}$
  if $f_{i,j}$ *is accepted* **then**
    $j \leftarrow j + 1, s_t^{new} \leftarrow s_t^{new} + 1$
  end
  if r_i *is not accepted or* $j \succ |r_i|$ **then**
    if r_i *is not accepted and* $j \succ 1$ **then**
      back track the network state to $s_t^{new} - j + 1$
    end
    if R_τ *is all processed* **then**
      repeat
      $\tau \leftarrow \tau + 1$
      until $R_\tau \neq \emptyset$
      introduce a new request r_1' *from* R_τ
      restart $i \leftarrow 1$, $j \leftarrow 1$
    end
    else
      introduce request r_{i+1} *from* $R\tau$
      $i \leftarrow i + 1$
    end
  end
  Calculate request $U(s_t, a)$
  Move the state to s_{t+1}^{new} *and get the next VNF*
  Store transition $(s_t^{new}, a, U(s_t^{new}, a))$ *to PG batch memory*
end
for $t \leftarrow 1, T$ **do**
  $u_t \leftarrow \sum_{q=1}^{t} \gamma^{t-q}(s_q, a)$
end
for $i \leftarrow 1, 10$ **do**
  for $t \leftarrow 1, T$ **do**
    $\theta = \theta + \sum_t \alpha \bigtriangledown \sum_{i=1}^{t} \pi_\theta(s_i, t_i) U(t_i)$
  end
end

3.4.4 Description of LSTM module

LSTM [100] is a neural network structure based on the concept of 'cell state' in the network (see figure 3.4). It captures information from the previous steps and consists of different multi-gates; the input gate, the forget gate, and the output gate. The forget gate is considered as the most important gate in LSTM because it processes the hidden previous state h_{t-1} and the current input x_t by applying the 'sigmoid function' which extracts the final value for the interval between 0(forget data) and 1 (pass it through unchanged). Then the previous hidden state and inputs pass to the 'sigmoid' of the input gate while the hidden state and current inputs pass to the 'tanh' function, followed by multiplying 'sigmoid' output by 'tanh' output. The cell updates itself using these values by multiplying the cell state with the forget gate vector and the input gate vector to get the new, updated cell state value. In out-put gate level we can extract the value of the next hidden state which contains the information about previous inputs. First, the previous hidden state (h_{t-1}) and the current state are passed to the function 'sigmoid' as shown in Figure 3.4. Then the new cell state $S_history$ generated through multiplication $S_history - 1$ wih forget vector and add its results to new hiden state (multiplied by result of 'tanh' and 'sigmoid') is passed to 'tanh'. This final value lets to the network to decides which information the hidden state h_t should carry. This hidden state is used for prediction. The main goal of LSTM is to provide short-term memory for RNNs that can last for thousands of time steps, and thus long-term memory. LSTM is mainly used to classify and predict based on time chain data.

3.5 Modelization and Policy Gradient PG-based training procedure

The main objective of our PBDRL approach is to find the best VNF placement and SFC service function chain deployment in an NFV network taking into account real-time network dynamism and traffic, resource capacity, and bandwidth requirements. We exploit the characteristics of the DRL construct that are associated with the value (e.g DQN) of the VNF Chain neuron deepening and the NFV environment agent training policy (Reinforcement). Based on this training policy, the placement action of a VNF has been

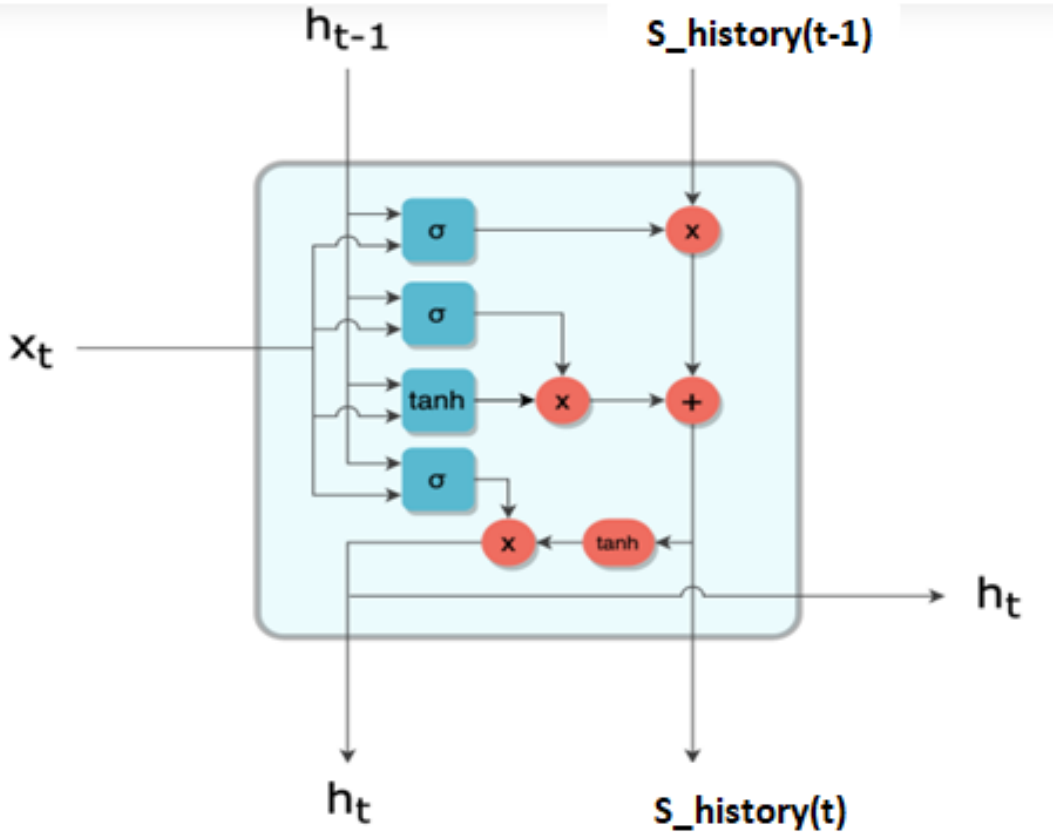

Figure 3.4: The Long Short-Term Memory (LSTM) cell and its task in the sequential data processing.

carried out with an estimated value calculated as a reward.

With PG, automatic feature architecture and end-to-end learning can be provided, thus the domain knowledge becomes not essential. Also, PG participates to improve training efficiency. For the problem of SFC deployment which requires a large action space, PG provides an action space with a high dimension and improves training efficiency and convergence toward the optimal solution. In fact, our approach integrates the Markov Decision Process (MDP) to deal with the movements and transitions of dynamic network state and capture them with ease. Therefore, we extract the resource currently used (i,e., CPU, bandwidth, memory, etc) and SFCs execution result as MDP state. Thus, we conclude that MDP state transitions describe dynamic network movements which can usually be automatic and continuous. Specifically, the PG uses its component as depicted in Figure 3.3, where the input layer represents the vector of states while the output layer represents the probability distribution of actions.

First, it is determined how many hidden layers there are and how wide each one is in $DNNQ^{\pi}$. Then, one hidden layer h is first used with S serving as the input layer and A serving as the output layer. To define the width of each layer, we exploit the next useful empirical equation $|h| =^2 \sqrt{|S||A| + \alpha}$. Each hidden layer has a number of nodes, the first hidden layer for DNN with two hidden layers has a $|S|^{2/3}|A|^{1/3} + \alpha$ nodes and the second has a $|S|^{1/3}|A|^{2/3} + \alpha$ nodes. In this manner, we can determine how many hidden layers are necessary for each set of nodes. For instance, less than 200 nodes only require three hidden layers, where the associated layers are characterized by the functions *relu* and *tanh*, which activate the network and give the neural network non-linear properties. To facilitate the training of the neural network, we normalize the input onto a tiny scale (i,e, from -1 to 1). Then, we divide the resource related inputs in state s by the maximum $C_{max} = \underset{v \in V}{max}(C_v)$. So that the demand resources $C_{f_{j,i}}$ of the current $VNF_{j,i}$ is normalized as follow:$(\frac{C_1^t}{C_{max}}, \frac{C_2^t}{C_{max}}, ..., \frac{C_{|v|}^t}{C_{max}})$. In the similar way, the bandwidth related inputs and delay related inputs are respectively divided by the maximum $W_{max} = \underset{v \in V}{max}(W_v)$ and maximum $d_{max} = \underset{i \in I}{max}(d^i)$ and normalized as follows: $(\frac{W_1^t}{Wmax}, \frac{W_2^t}{Wmax}, \cdots \frac{W_{|v|}^t}{Wmax}, \frac{d_1^t}{dmax}, \frac{d_2^t}{dmax}, \frac{d_{|i|}^t}{dmax})$.

3.5.1 VNF/SFC placement and deployment approach strategy based on dynamic environment features

To adapt our approach to the dynamic variations of the network, we divide each request into time slots. Generally, the deep reinforcement system in our approach selects first the requests, then puts all requests in R_t one after one and establishes a chain of decisions regarding whether to accept or reject each SFC. Finally, it does the updating of the network states. The deep reinforcement strategy of our approach is based on serialization and backtracking methods to the process of VNF processing in SFC. Our approach is mainly based on serialization and backtracking methods of VNF processing in SFC in order to minimize the large discrete action space. One VNF is treated within each MDP state transition. If any VNF cannot be placed due to insufficient resources, latency issues, or bandwidth limitations, the request can not satisfied. Thus, the deep reinforcement strategy of our approach backtracks to the previous state. Knowing that between every two-time slot, there are two cases:

(1) Intra time slot: our system takes one VNF of an SFC after another to conserve the

sequence order. MDP state is changed when a VNF is rejected or accepted. As shown in Figure 3.5, when two requests come (such as SFC1 and SFC2) in the time slot, VNF1 is processed to place it in the node considering the resource sufficiency of the hosting server or VM, and the bandwidth capacity by the NFV deep agent. An action is then passed to set the VNF1 on the appropriate VMs. No rewards are returned to our system agent as SFC1 is not fully deployed. Therefore, the system enters state s_{t+1}, where the reward $U(s_t, a) = 0$. In state $S_t + 1$, if no action is taken then $SFC1$ is rejected. This is either because there are no VM nodes with enough resources to embed the VNF2, or other constraints are not met (see ILP in pages 67 and 68). While, SFC2 is successfully deployed by the same method in $s_t + 3$ with a reward that is the throughput of SFC2 minus the cost of the resource consumption, $U(s_t + 3, a) = W_{r2}P_2 - Cost(S_{t+3}, a)$. To avoid multiple VNF instances, in order to improve the resource utilities and also to choose the nearest node for the VNF instance which minimizes the traffic and the latency, many SFCs can share the same VNF and in the same node. In this case, the same procedures are put in place to embed the VNFs into chain in intra and inter slot as shown in Figure 3.5.

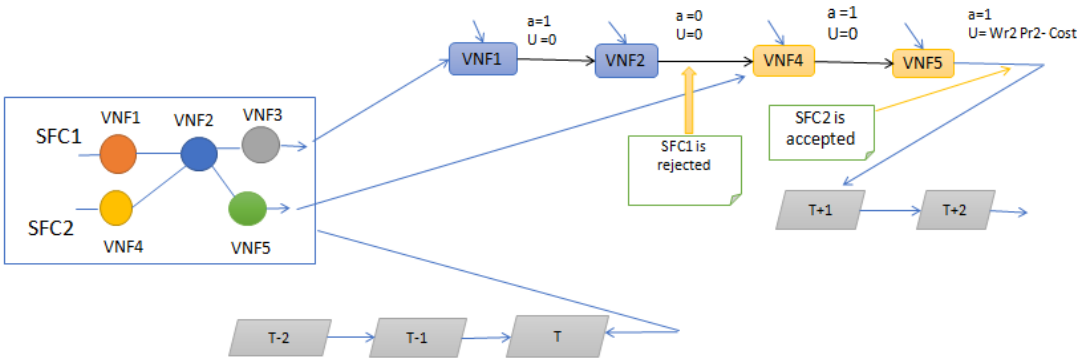

Figure 3.5: The procedure of our model in different time slots

(2)Inter time slot: it is the time in which no request is received through a number of consecutive time slots. So, the network state does not change and no action can happen. As shown in Figure 3.5, at every two-time slots (i.e, $\tau + k$,$\tau + k + 1$) the system removes the timeout request and retrieves the resources. In the next time slot, a new request arrives and the agent takes state transition and according to constraints gives an action

or rejected the SFC. Then, a reward is calculated in two cases as follow:

$$U(s_t, a) = \begin{cases} -\sigma_t C(s_t, a) + w_{r_i} P_{r_i}, & \text{si } r_i \text{ is accepted} \\ 0 & r_i \text{is rejected or not fully deployed.} \end{cases} \tag{3.10}$$

where σ_t is a binary variable that defines the time slot transitions to state s_t, and $C(s_t, a)$ is the total cost as defined in equation 3.9. In the same context, we introduce the reward function at state s_t taking into account its effect on future decision reward as:

$$U_{s_t} = \sum_{i=0} \gamma^i U(s_t, a) \tag{3.11}$$

with $\gamma \in [0, 1]$ is the future reward discount factor. Algorithm 1 summarized the whole procedure of our approach, allowing the NFV system to adaptively provision SFCs for different requests with different QoS requirements [101] based on the sequencing and backtracking methods.

3.5.2 Training procedure

The training procedure is divided into episodes. In each episode, all state transitions are saved in a buffer and used for training until that episode is over. During training, the PG adaptation optimizes the SFC deployment quality based on the gradient calculated from the deployment estimates. The main objective is to set a policy that optimizes the ultimate reward at the conclusion of a series of state transitions. The policy is denoted by the formula: $\pi_\Theta = P(a|s, \Theta)$. This equation expresses the probability of action a in the state s under parameter Θ. So the objective function is defined as follows:

$$\jmath(\Theta) = \sum_t \pi_\Theta(s, a) R(t) \tag{3.12}$$

It represents the final reward for each episode. The policy gradient which parameterized by Θ is formulated as follows:

$$\bigtriangledown_\Theta \jmath(\Theta) = (\frac{\partial \jmath(\Theta)}{\partial \Theta_1}, ..., \frac{\partial \jmath(\Theta)}{\partial \Theta_n}) \tag{3.13}$$

Where the gradient descent of the parameter Θ is updated as follows

$$\Theta_{i+1} = \Theta_i + \alpha \bigtriangledown \Theta_i \jmath(\Theta_i), \tag{3.14}$$

α is the learning rate, which can be modified according to the convergence speed of the training process. The ascending gradient related to the product sampling probabilities and the cumulative reward associated with selected actions based on the PG algorithm aims to maximize the final reward $\jmath\Theta$. Therefore, policies with high-reward actions are considered in the future. At the end of each episode, the total state reward s_t is calculated and transmitted to the agent. In these episodes, our system agent is trained until the convergence of the reward. During this period, the reward does not decrease, for that we add the discount factor $\gamma = 1$, and the noise machine to improve the training effect, as a probability $\epsilon \in (0, 1)$ to select a random action a_t or choose $a_t = P(a|s, \Theta)$.

3.6 Simulation and results

3.6.1 Network description and data-set

To carry out a simulation giving reasonable results, we propose an NFV network topology simulation based on tree architecture. We use stable-baseline3 [102] and the Proximal Policy Optimization (PPO) algorithm [103]. In our topology of network, there are three layers of switches for server connections. We count the number of VM nodes $|V|$(host or POP) from 24 to 500, with $[1000, 5000]$ of CPU resource and $[10.000, 50.000]$ of memory, each node have $[1, 500]$ units of CPU resource and $[1, 64]$ units of memory. We set also the intra-pod delay between 40 to 100 μs and inter-pod delay between 50 to 200μs.

In our approach of VNF placement and SFC request deployement, we used from 1 to 6 categories of VNFs to create from 20 to 200 SFC request. Whereas, there are 6 categories VNFs (i.e, firewall, NAT, IDS, Load balancer, WAN optimizer, and low monitor) [104] [105] [106]. These requests are characterized by Qos requirements including latency and throughput. We assume that each VNF instance has a fixed service rate μ ranging from 100 to 1000. In each episode, we use from 1000 to 60000 time slots. We assume that each request has a packet arrival rate ranging from 1 to 100 packets/s. To check the importance of our Parallel Bi-state module deep reinforcement learning approach,

we compared it with GPLL (a greedy-based policy to find a path with the lowest latency)algorithm and NGSP(non-recursive greedy) algorithm as a baseline which ensures placing VNF of the SFCs in the nodes used with high utilization rates of the resources. Our simulation platform is a Python-based framework and involves Tensorflow to build the architecture of the Parallel Bi-state module and to devote deep neural network. All the following experiments were performed on an Intel(R) Core(TM) i7-6850K with 6 cores 12 threads and Ubuntu as operating system.

3.6.2 Evaluation control parameters

We have three main parameters of evaluation control which are the number of nodes, the number of VNFs and the resources (CPU, memory):
- VM nodes number: the number of nodes that directly affects the cost of resources (CPU, memory) which controls the reward. Hence, when the number of nodes is decreased, the reward is increased.
- VNF number: the number of VNFs affects the number of trials and mainly the number of episodes. Indeed, when the number of VNFs increases in each request, the number of episodes also increases, as well as the requirement of each episode in bandwidth, latency, throughput, and resources.
- resources (CPU, memory): the resources also present themselves as a setting parameter which control the reward and the cost in our network.

While the evaluation of the effectiveness of our approach is carried out by measures dedicated to the execution environment we used (reinforcement deep learning) which are the reward and cost (see equation 3.15 and 3.16).

3.6.3 Evaluation measures

In order to evaluate the effectiveness of our approach, we calculate the total throughput of accepted requests. Also, the use of the PG algorithm in our approach additionally offers two evaluation measures which are cost and reward.

3.6.3.1 Reward

At each network, a captured environment state is processed by the agent. If this state meets all SFC request requirements (capacity, bandwidth, resources, etc.), a reward is returned and a cost to the agent with the appropriate selection of VNF placement in VM (host). As a result, an SFC is generated from the state s_t with reward U(st, a) with action a denoted as follows:

$$U(s_t, a) = W_r P_r - Cost(s_t, a) \tag{3.15}$$

Where $W_r P_r$ is the throughput of SFC, and $Cost(st, a)$ is the resource consumption cost.

3.6.3.2 Cost

Our work requires determining the cost of the consumed resources (CPU, memory, etc.), as well as the cost of the bandwidth occupied by the VM server to evaluate the performance of our approach. Generally, we can define the resource consumption cost by the sum of VNF placement cost in time slot τ and host VNF v as follows:

$$\sum_{\tau=1} \sum_{v \in V} z_{v,\tau}(cC_v + wW_v) \tag{3.16}$$

where c is the unit cost of VM resource, and w is the unit cost of bandwidth.

3.6.3.3 Throughput

The total throughput of accepted requests is a measure of the rate of accepted requests for bandwidth request W_r with marked Time to live (TTL) which can be defined as follows:

$$\sum_{r \in R} y_r W_r \tau_r, \tag{3.17}$$

with y_r is the binary variable set to 1 if request r is accepted, 0 otherwise.

	num_timeslots	num_requests	service_rate	num_vnfs	sfc_length	bandwidth	max_response_latency	cpus	memory	vnf_delays
count	2.0	2.000000	2.000000	2.0	2.000000	2.000000	2.0	2.000000	2.000000	2.0
mean	6000.0	110.000000	550.000000	6.0	3.500000	60.000000	1000.0	10.500000	7.500000	0.0
std	0.0	127.279221	636.396103	0.0	3.535534	56.568542	0.0	13.435029	3.535534	0.0
min	6000.0	20.000000	100.000000	6.0	1.000000	20.000000	1000.0	1.000000	5.000000	0.0
25%	6000.0	65.000000	325.000000	6.0	2.250000	40.000000	1000.0	5.750000	6.250000	0.0
50%	6000.0	110.000000	550.000000	6.0	3.500000	60.000000	1000.0	10.500000	7.500000	0.0
75%	6000.0	155.000000	775.000000	6.0	4.750000	80.000000	1000.0	15.250000	8.750000	0.0
max	6000.0	200.000000	1000.000000	6.0	6.000000	100.000000	1000.0	20.000000	10.000000	0.0

Figure 3.6: Evaluation data-set of request

3.6.4 Experimentation and results analysis

Training efficiency: To test the training operation in our approach, we apply it at different scales of networks with a different number of nodes or according to operating servers nodes as illustrated in Figure 3.6. We divide the learning network according to episode length to show the rewarding result according to operating server nodes.

Table 3.2: Variation of error ratio and reward for PBDRL, LSTM and NFV deep algorithm in mixed environment based on cloud data centers and MEC.

	Algorithm								
	PBDRL			First Fit			NFVdeep		
Episode length	Error ratio	Reward	Throughput	Error ratio	Reward	Throughput	Error ratio	Reward	Throughput
268	0.339	2.92	1.12	0.39	0.23	1.25	0.33	1.4	1.4
271	0.52	3.03	1.15	0.45	0.24	1.25	0.54	1.38	1.41
273	0.439	3.16	1.25	0.49	0.25	1.26	0.40	1.41	1.42
275	0.2	3.15	1.30	0.55	0.25	1.25	0.45	1.37	1.41
276	0.409	3.17	1.36	0.61	0.25	1.26	0.23	1.42	1.43
279	0.378	3.22	1.37	0.67	0.26	1.26	0.44	1.42	1.43
281	0.25	3.27	1.41	0.69	0.27	1.25	0.40	1.44	1.42
282	0.36	3.29	1.46	0.71	0.28	1.26	0.37	1.46	1.43

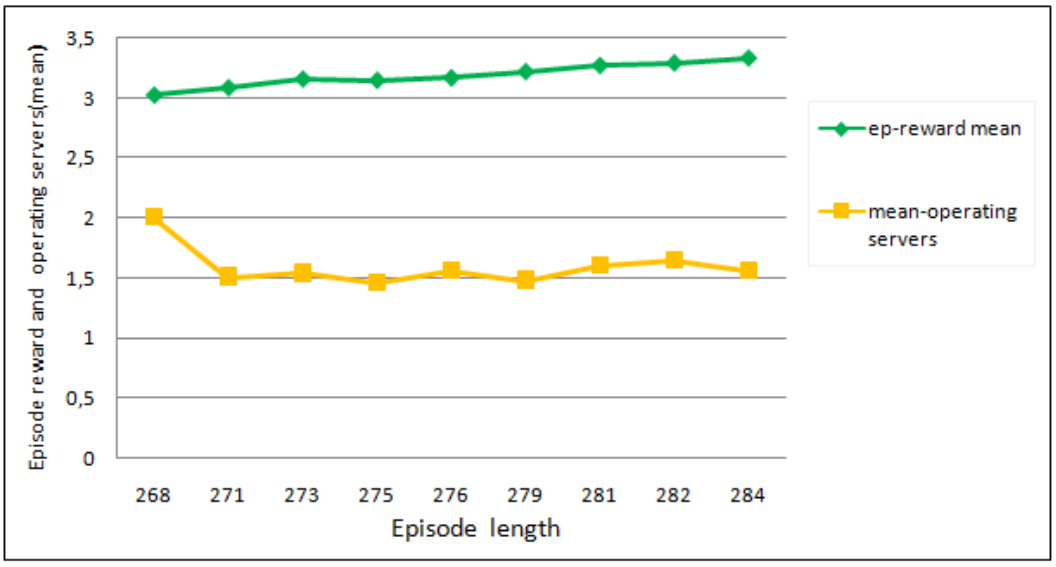

Figure 3.7: The evolution of episode reward as a function of episode length in our approach.

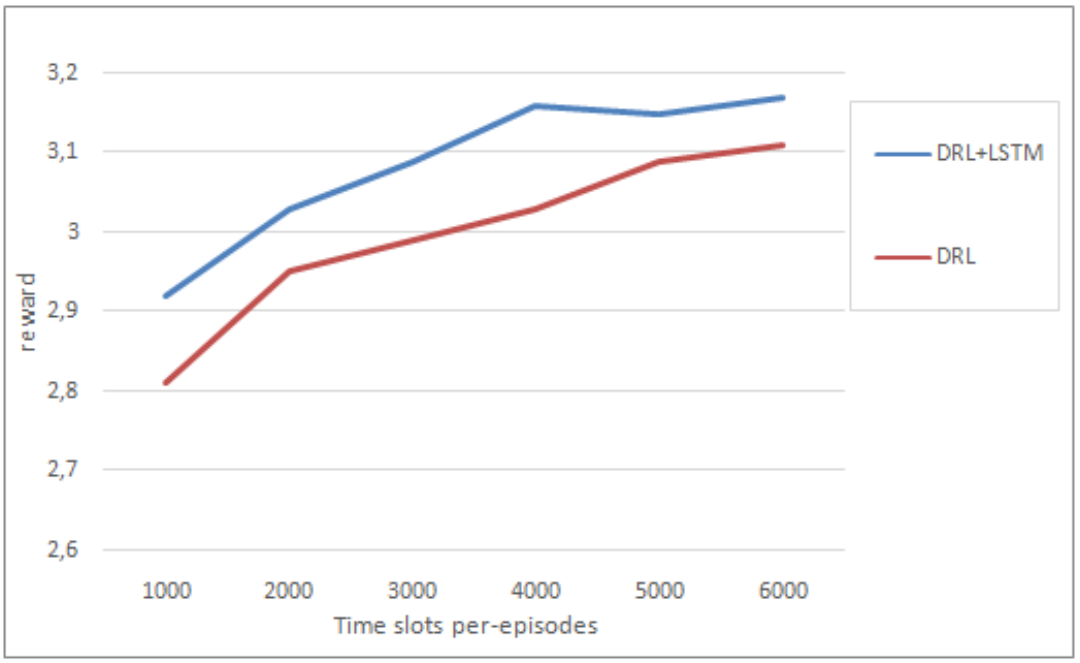

Figure 3.8: The effect of using LSTM module with MDP

The curves in Figure 3.7 show that the average episode reward is slightly increased as the average episode length increases while the average server operation (operating-server-mean) increases with a drop when peaking at the average episode length (ep-len-mean) level 271 which confirms that the means of operating server varies according to the quality of service offered and not according to the length of the episode. Moreover, Figure 3.8

confirms the importance of the integration of LSTM with MDP to guarantee the optimization of the reward results obtained compared to classic reinforcement learning(RL).

Total cost: Figure 3.9. can intuitively illustrate the evolution of server cost according to operating servers for each request. As is evident in this figure, the total cost increases as the average value of operating servers increase from 91.1 for 1.48 avg operating servers to 143 for 1.93 avg operating servers. This total cost is unevenly divided over 3 resources CPU cost, memory cost, and bandwidth cost. The CPU cost represents the highest percentage of total cost by 82% at the 1.48 operating servers level while the bandwidth and memory cost represent respectively 5.59% and 11.13% of the total cost at the same level. Confirming these results, Figure 3.10 shows that the CPU is always the most expensive among the other resources (memory, bandwidth) because it works on a server basis where each server processes data via its processor. This difference in the cost of resources can be explained by the possibility of controlling memory consumption, bandwidth (space) by reuse or integration while the processor could not share their functionality with more servers to control their cost.

Acceptance ratio: Figure 3.11 shows the SFC request acceptance rate according

Figure 3.9: The total cost of operating servers for each request in our approach

to the means of operating servers. The decreases in request acceptance rate when the average operating server increases are explained by the saturation of these servers and their inability to accept other requests. Also, the comparison between the actual value and new value(estimation value of PBDRL) shows the efficiency of our approach to improving the request SFC acceptance ratio. However, there is a slight increase in level 1.6 operating servers since the prediction error could not be avoided. Figure 3.11 adds

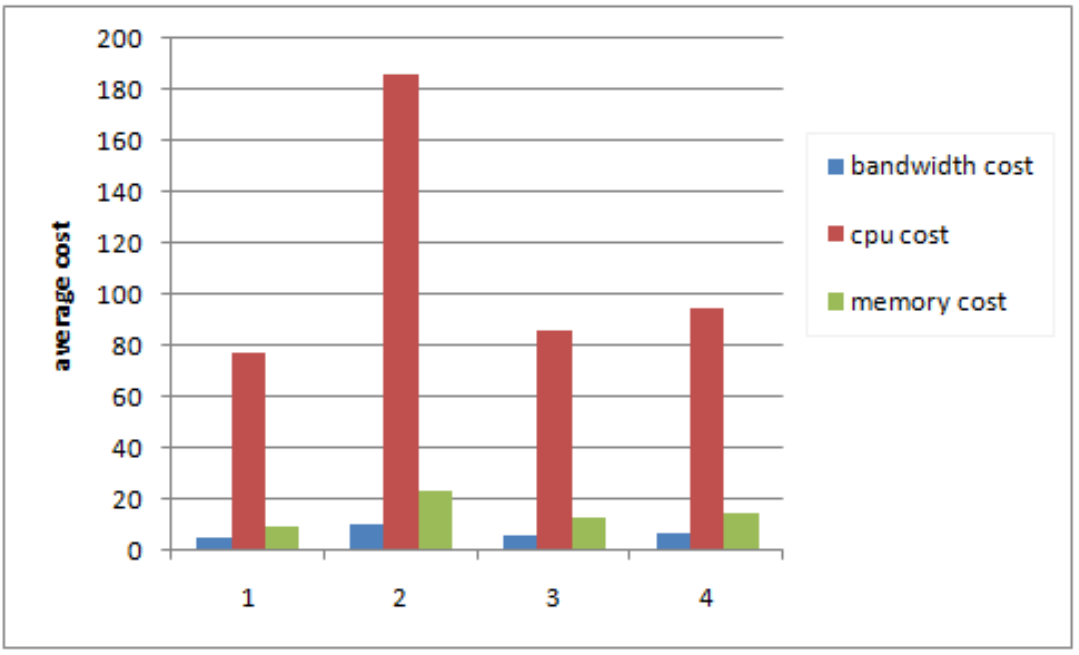

Figure 3.10: The average cost of different resources in our approach

a comparison between the acceptance rate of our approach and First Fit to show the importance of using LSTM as a module with deep reinforcement learning.

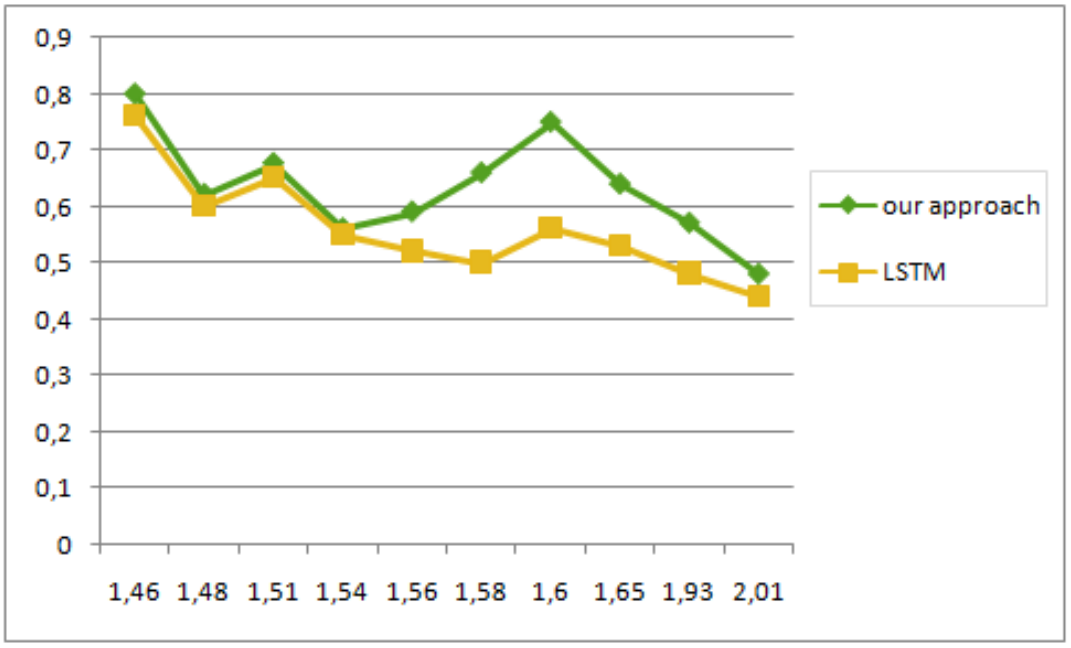

Figure 3.11: SFC request acceptance ratio

Reward: Concerning the reward, Figure 3.13 depicts the evolution of the reward as a function of bandwidth. The figure clearly shows that the reward increase with the

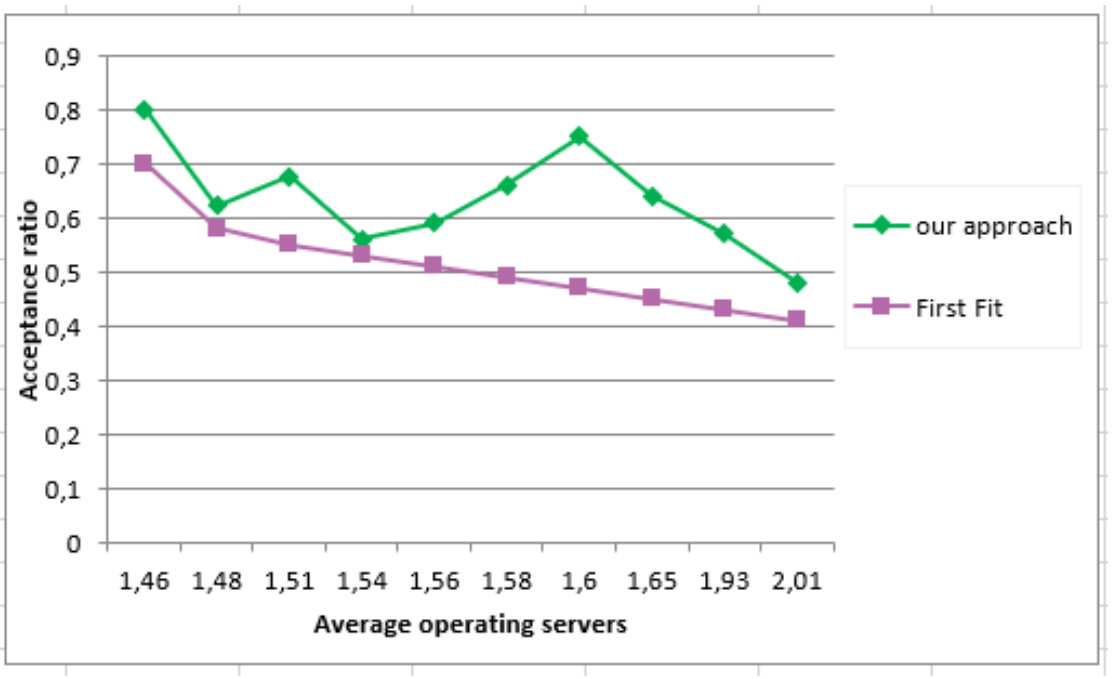

Figure 3.12: comparison of SFC request acceptance ratio between our approach and First Fit algorithm

bandwidth augmentation.

In Figure 3.14, we compare the reward of our approach with the NFV deep approach and First Fit. This comparison shows the effectiveness of our approach by 60% more than the NFV deep approach and 61% than the First Fit algorithm. So, for 1000 time slots per episode until 6000-time slots per episode, the reward of our approach increases quickly compared to NFV deep approach.

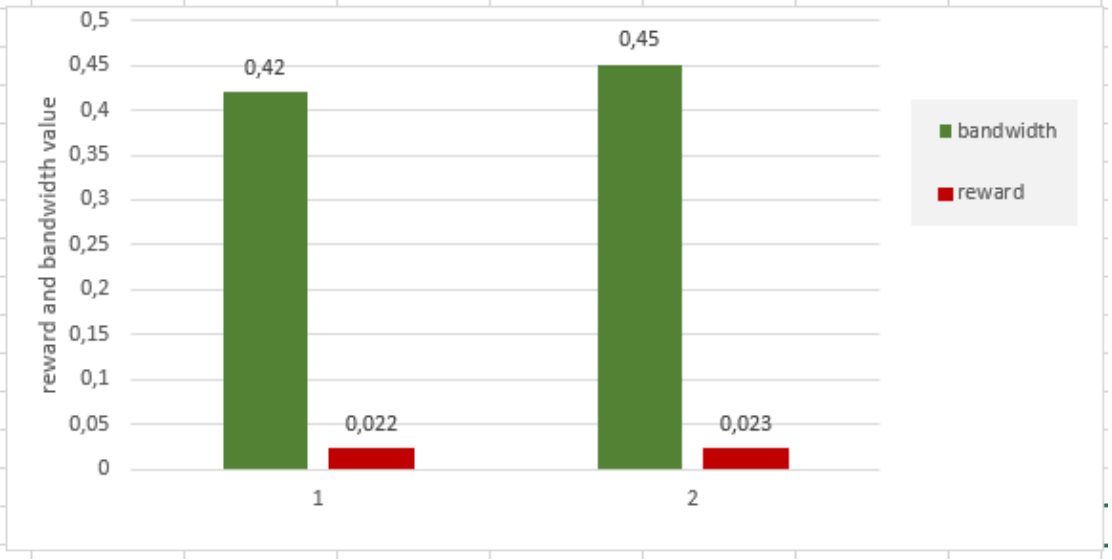

Figure 3.13: Average reward of the request according to bandwidth in our approach.

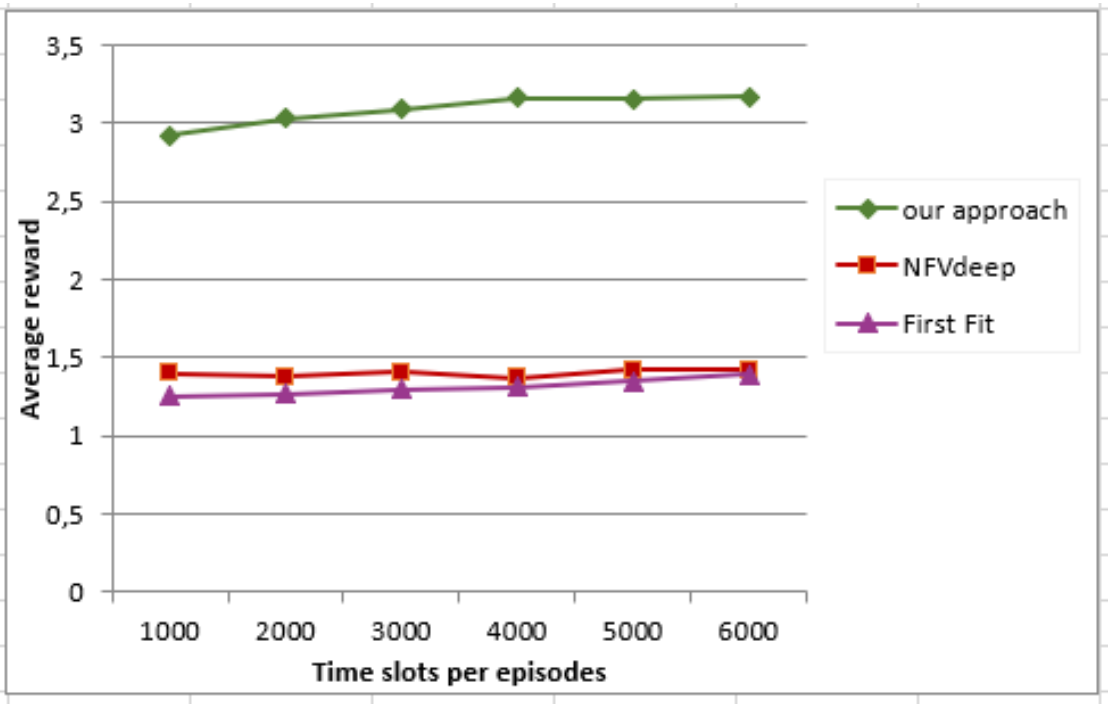

Figure 3.14: Average reward for each request in our approach compared to NFV deep algorithm and First Fit algorithm

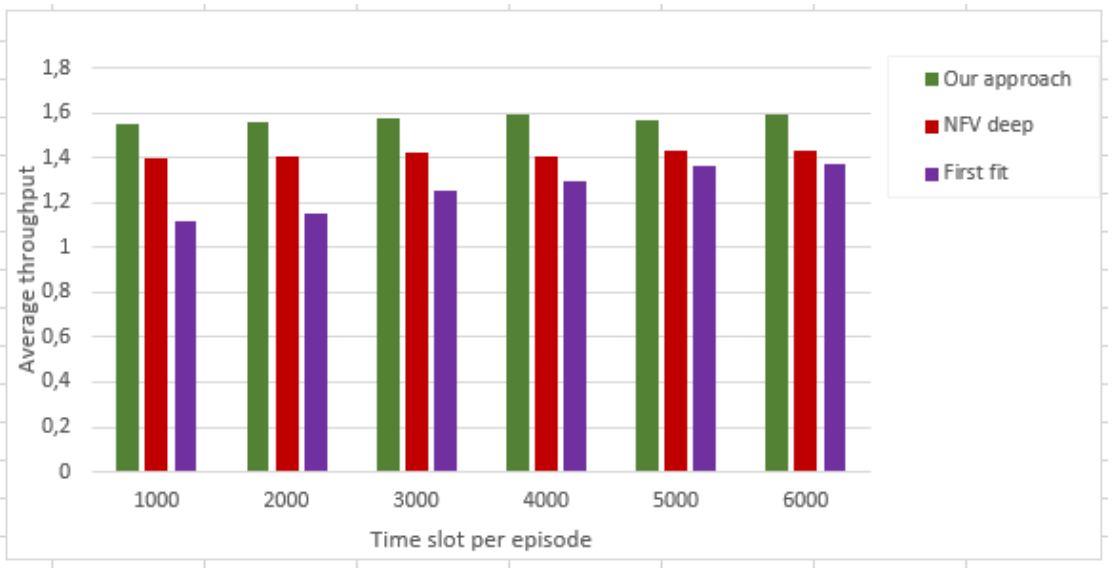

Figure 3.15: comparison between the SFC request throughput of our approach (PBDRL) and the algorithm of NFV deep and First Fit

Table 3.4: Sets of test data collections

Time slot per-episode / Average throughput	1000	2000	3000	4000	5000	6000
Our approach	1.55	1.56	1.58	1.59	1.57	1.59
NFV deep	1.4	1.41	1.42	1.41	1.43	1.43
First Fit	1.12	1.15	1.25	1.3	1.36	1.37

Average throughput: the average throughput as a quality of service parameter helps to illustrate the effectiveness of our approach to optimize the placement and deployment of SFC requests in order to minimize latency and network throughput in a cloud infrastructure. As shown in Figure 3.15 and Table 3.4, the average throughput of our approach is higher than other NFV deep and First Fit at each time slot per episode. So, the average throughput of our approach reaches 1.59 for 6000-time slots per episode, 1.43 for NFV deep approach, and 1.37 for First Fit approach.

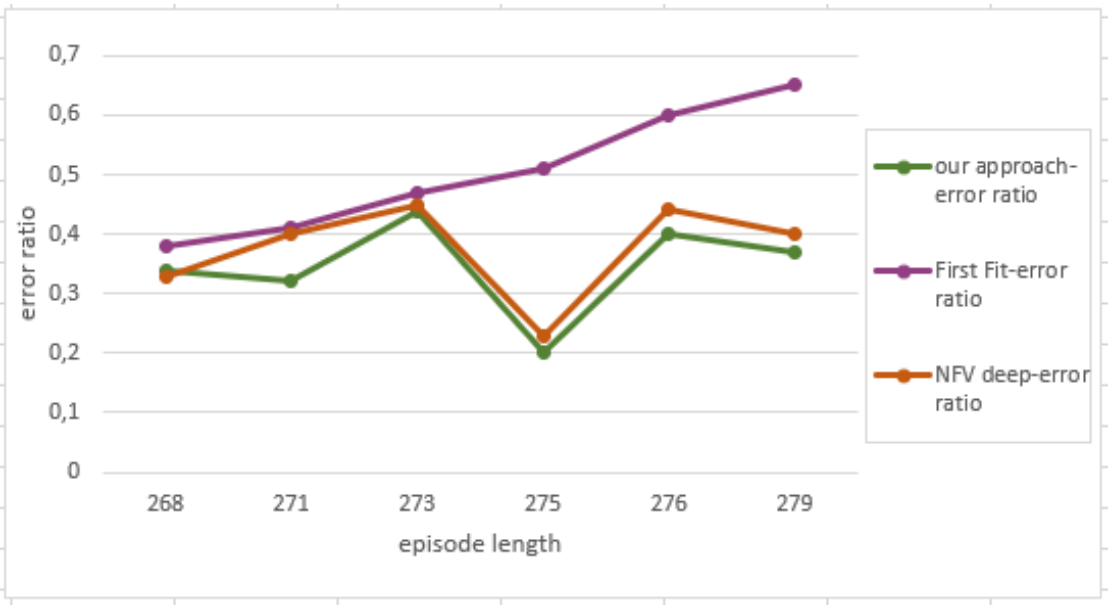

Figure 3.16: Error ratio for each request in our approach compared to NFV deep algorithm and First Fit algorithm

In Figure 3.16, we try to show the efficiency of our approach by extracting the error rate in comparison with the algorithm of NFV deep and First Fit. As illustrated in the experimental results, the error rate of our approach is low compared to the NFV deep and First Fit algorithm due to the high precision mechanisms of our approach. The error rate of our approach increases slightly with episode length, the drop made at episode length = 275 can be explained by the agent's ability at this point to respond and respect all the network constraints while minimizing the error ratio.

3.6.5 Limit of our approach

Our model is characterized by an integrated architecture that guarantees a very powerful and efficient prediction policy. This policy is based on the use of two modules in parallel to capture the state of the NFV environment which are: MDP to capture the spatial

state of the environment and LSTM to capture the temporal or historical long-term state. Also, our approach ensures high accuracy considering the filtering performed by the forget gate of LSTM and the policy gradient used by the agent to make the best decision. However, some issues are not solved by our approaches such as the problem of certainty and the problem of security which is a recurrent and classic problem.

3.7 Conclusion and prospects

3.7.1 Summary

In this work, we studied the consistency of the deep learning reinforcement strategy with the problem of VNF chaining and the SFC requests deployment in a cloud network. Our Parallel Bi-State Deep Reinforcement Learning approach decreases the latency by improving network throughput and SFC request acceptance rate. Concisely, our approach consists of (1) ILP formulation for VNF placement and chaining and (2) placement prediction, and deployment of SFC requests by deep reinforcement learning based on policy gradient (PG) for action decision-making (placement of VNF for an SFC request) considering the state of the environment and their history which is captured using MDP and LSTM.

3.7.2 Prospects

There are two possible directions for our future work. In order to provide practitioners and academics with more knowledge on how to solve the problem of the Reinforcement learning for SFC Placements and Deployments, we first intend to conduct a more in-depth comparative study, on other standard and recent data-sets. In the second direction, we intend to conduct further studies utilizing VNF data-sets for SFC Placements and Deployments data-sets from different sources.

Chapter 4

Hybrid approach for VNFs placement and chaining considering SFC migration cost

4.1 Introduction

The consolidation of VNFs in VM implicitly requires taking into consideration a predefined order of VNFs to avoid violating SLAs and to ensure proper functioning of the requested network service. Depending on the requested network service, an SFC which consists of a series of VNFs connected to one another in a predefined order is generated. For this, each placement or migration of new VNF should be maintains the same order in the SFC and SLAs are respected. In this regard, we propose a new approach combined FCA and Fuzzy Inference System (FIS) called Fuzzy-FCA to meet the constraints of VNF placement in VM and Bi-state Deep Reinforcement Learning (PBDRL) algorithm to place and migrate VNFs in chain. More precisely , our proposal aims at placing and consolidating the VNFs in a chain according to a predefined order and ensuring the SFC requested quality of service.

Fuzzy FCA aims at consolidating the VNF in the smallest number of VMs so that the number of active virtual machines is reduced and the exploitation of resources is enhanced by improving the load balancing. In this way, the latency, the cost of migration and resource and the consumption of energy are reduced. PBDRL aims at placing

and chaining the VNF according to predefined order of VNF so that end to end delay is minimized and the quality of service requirements are verified. The rest of chapter is organized as follows: In section 4.2, we first describe the VNF placement, VNF consolidation, VNF chaining and VNF migration problems before presenting our hybrid algorithm which combines the algorithms proposed in the precedent chapters to improve the placement and chaining. In section 4.3, we explain the operation of our proposal which uses fuzzification, inference, defuzzification to improve the resource utilization, reduce the delay and energy consumption, etc. Section 4.4 and 4.5 present simulation we done to measure and discuss the performance of our approach. Section 4.6 is devoted to conclusion.

4.2 Model Description

In this approach, we combine between two models we previously proposed in the following way: the Fuzzy FCA model is used to consolidate the VNFs in the most appropriate VM by considering the capacity constraints and resource availability before carrying out the chaining [11]. Next, we apply the PBDRL model to place and chain the VNFs that are already consolidated in the VMs according to the requested network services. This approach aims at improving the gain and the quality of service (Qos) while ensuring the minimization of the resource and migration costs, the energy consumption, the latency, and the throughput. For this, we first use the FCA to group the VNFs with the network services attached to them and exclude the others. Then, we apply Fuzzy-FCA to consolidate candidate VNFs into the smallest number of VMs most appropriate VMs considering their capacity and available resources. After that, we solve the problem of placing VNFs in a chain respecting the predefined order of VNFs in a SFC in order to meet the requirements of the migration and the requested network service. We propose the PBDRL algorithm based on neural network architecture to model the chaining aspect of VNFs in SFC and deep reinforcement learning to place VNFs in the most appropriate node that is chosen by the agent (PG). Figure 4.1 describes the roles of the combined approaches in our proposal.

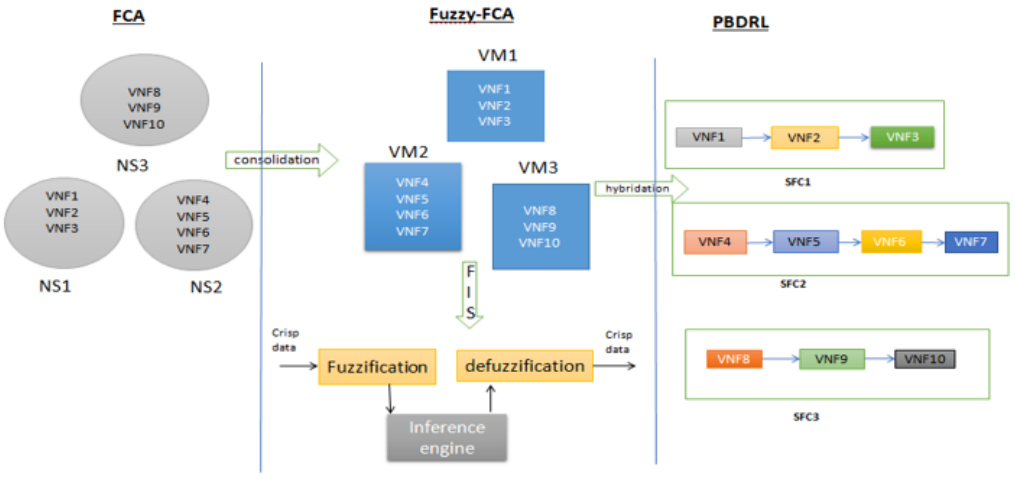

Figure 4.1: Hybrid VNF placement architecture.

4.2.1 Adopted algorithm for VNF placement and chaining

In this section, we identify the algorithms adopted for placement and chaining of VNFs for each SFC request. First, the VNFs should be place in the corresponding NSs according to their type by applying the FCA algorithm. Next, we extract the VNF candidates that are active and usable in the network service for use as an object in the second Fuzzy-FCA procedure. Second, VNF candidates are consolidated into the most appropriate VM considering capacity constraints and resource availability according to Fuzzy Inference System (FIS) based Fuzzy-FCA algorithm. To react with the environment, we use swarm intelligence algorithm which can perceive the indicated environment state by sending an internal stimulus. Finally, we use a PBDRL algorithm to place and chain the VNFs according to a predefined order which is given by the requested SFCs. The main objective is to solve the problem of VNFs placement respecting the chaining of VNFs and the multiplicity of VNF instance constraints, the congestion at the links and nodes levels, the resources, capacity and availability constraints.

4.2.2 VNFs placement and consolidation

As we have seen previously, each VNF is used to perform a specified network service. For this, we use an FCA algorithm to place the VNFs taking into account the relationship between each VNF and NS according to the context (O, A, I) disseminated by the

concept C. Then, we apply the rules of dependence, weight, maximum coverage and candidate concept in order to remove and eliminate unnecessary VNFs. In what follows, we apply the Fuzzy-FCA algorithm to consolidate the VNFs in the most appropriate VMs according to their efficiency determined by the parameter Packaging Efficiency which is computed as follows:

$$PackagingEfficiency = \frac{VNFnumber}{numberofVMactive} \tag{4.1}$$

The different rules of Fyzzy-FCA were applied in order to estimate the most suitable placement of VMs that is limited by a confidence threshold=0,6.

4.2.2.1 VNF placement and migration problem

The NFV network is characterized by a management and orchestration of predetermined resources which poses the problem of VNF placement optimizing the routing path that is used for SFC deployment. The arrival of new service requests requires a modification in the placement of some VNF instances and in the path of SFC to meet the requirements and optimize the exploitation of resources. These modifications are made by MANO, and called migration. The placement problem differs from the migration problem because the migration problem is starting from the current placement to find the new placement. Generally, the system operation requires more consideration of the migration than the placement. The placement and the migration problems are two problems that cannot be separated since both of them affect the network performance and cost. These two problems can be detailed as follows:

A. VNF and SFC placement problem: the VNF and SFC placement problem is defined in particular by an NFV network with a set of service requests, VNF placement strategy and SFC sought, considering dynamic routing to minimize the deployment cost under the SFC chaining constraints, time frame, and available resources. Notably, the placement problem arises during the initial deployment of VNFs and SFCs in the network.

B. Migration problem of VNFs and SFCs: the migration problem consists in particular of an NFV network with a set of service requests, **the current** VNF and SFC **placement matrix**, sought VNF and SFC placement strategy, considering dynamic routing to minimize the cost of deployment under the chaining constraints of SFC, the

deal and the available resources. This problem arises when it is necessary to move a VNF from one location to another for performance or resource management reasons.

4.2.2.2 SFC/VNF migration problem model

To identify migration problem constraints and to optimize the objectives, we propose to add the following lines to ILP model described in Chapter 3. The new parameters are described in table 4.1:

- $m = (m_{vv'di})$ is the migration decision matrix. If the i^{th} VNF is moved from v to v', $m_{vv'di} = 1$ else $m_{vv'di} = 0$

- $x = x_{ed}$ is the routing decision matrix. $x_{ed} = 1$ if request d uses link e, otherwise $x_{ed} = 0$.

- $z = z_{vdi}$ is the SFC placement matrix after migration. $z_{vdi} = 1$ if node v provides f_{di} else $z_{vdi} = 0$.

The migration of i^{th} request VNF $d(f_{di})$ is from node v to node v' if only node v provides f_{di} for the current SFC placement and for the new SFC placement. The current SFC allocation for the service request set is defined by $\pi = \pi_{vdi}$ according to the following equation:

$$\pi_{vdi} = \begin{cases} 1 & \text{if node } v \text{ provides VNF } fdi \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

with the following conditions:

$$m_{vv'di} \leq \pi_{vdi}, \forall v, \forall v', \forall d, \forall i. \tag{4.3}$$

$$m_{vv'di} \leq z_{v'di}, \forall v, \forall v', \forall d, \forall i. \tag{4.4}$$

$$\pi_{vdi} + z_{v'di} - 1 \leq m_{vv'di}, \forall v, \forall v', \forall d, \forall i. \tag{4.5}$$

Migration of VNF f_{di} from v to v' requires high cost. To minimize this cost, we first assumed that the system uses a minimum weight path for transporting the data and the VNF instance in the new placement. Then, we identify the path cost $\rho_{vv'}$ from v to v' and the data size $\Phi_f$for VNF instance f. We can derive the VNF instance migration cost as follows:

$$G_{vv'fdi} = \rho_{vv'}\Phi_{fdi} \tag{4.6}$$

Therefore, the migration cost solution is written according to the following equation:

$$A_{m1} = \sum_{v,v',d,i} m_{vv'di} G_{vv'fdi} \tag{4.7}$$

the migration problem also includes the migration deployment cost as follows:

$$A_{m2} = \sum_{d,v,i} b_d z_{vdi} n_{fdi} k_v \tag{4.8}$$

where n_f, k_f are respectively the number of cores required to process one traffic unit of VNF type f and the cost of the providing one core at node v. A_{m1} consists in minimizing the cost of migration, A_{m2} consists in minimizing the cost of deploying a new application. A_{m3} is the reallocation cost which is defined by the sum of the migration and deployment costs:

$$A_{m3} = A_{m1} + A_{m2} \tag{4.9}$$

Table 4.1: Migration parameters and their description

Parameters	Description
b_d	The bandwidth capacity of demand d
π_{vdi}	The state of the current SFC placement matrix: $\pi_{vdi=1}$ if node v provides $VNF\ f_{di}$ else $\pi_{vdi} = 0$.
$\rho_{vv'}$	the low weight path from v to v'
$G_{vv'fdi}$	The migration cost of f_{di} from v to v'
Φ_f	is the data size of VNF f type
f_{di}	the i^{th} VNF of demand d

4.3 VNF migration and chaining based on Fuzzy-FCA supported by FIS

After identifying the constraints and formulating the VNF migration and chaining problem, we propose Fuzzy-FCA approach to improve the VNF migration and chaining process during the placement. This approach uses Fuzzy Inference System (FIS)to select suitable VNFs according to service requirement and SFC. It is characterized by flexibility of decision which aims at minimizing uncertainty and finding best decision in minimum number of rounds. At the beginning, classification and grouping of VNFs with the appropriate network service is done by FCA which is based on binary values. Then, fuzzy inference system aggregates the various VNF migration and chaining constraints based on fuzzy values associated with linguistic terms to extract the output that describes whether the VNF placement is favorable or not. Finally, it identifies the best VNF placement in VM by defuzzification application. Fuzzy inference system consists of 3 elements:

1) Fuzzification is used to transform the input light values to fuzzy values.

2) Fuzzy Rule Matrix generates fuzzy number corresponding to replaceable output using fuzzy input variables.

3) Defuzzification is the process of transforming these values back to clear values again. This technique differs from other techniques in the input and output parameters which are fuzzy and which are referenced by description such as high cost, wide bandwidth, moderate flow, etc. In our system, there are 5 inputs and 2 outputs of FIS which are:

-Input1: constraint of average delay (D_m) is the total delay for all VNFs when an SFC

is deployed.

-Input2: constraint of requested delay (D_d) is the end-to-end delay for all VNFs included in the SFC.

-Input 3: constraint of cost requested (Cd) to meet the requirements of a service.

-Input 4: constraint of used bandwidth B which identifies the capacity of connections between nodes.

-Input 5: constraint of capacity C which corresponds to a capacity of VM.

-Output1: VNF delay (D_0) is the delay of a single VNF used to deploy the requested SFC.

-Output2: requested SFC service cost (C_s) which can measure system performance

We propose to set U as the universe of discourse, F are the fuzzy sets which belong to U and which can be represented by membership function $uF : x \in [0, 1]$. After identifying the inputs and outputs, we can move on to the next step of choosing appropriate membership function (MF) for each input and output. First, to model the average delay by fuzzy logic, we use three fuzzy sets: F1, F2, and F3 which present membership functions: $uF1(x)$, $uF2(x)$, and $uF3(x)$. These membership functions are expressed by the following linguistic labels: low, moderate and high, where x is number of VNFs. As shown in Figure 4.2, $uF1(x)$ is low for x=200, $uF2(x)$ is membership function moderate for $x = 500$, and $uF3(x)$ is membership function high for $x = 1000$. F4, F5, and F6 are the bandwidth fuzzy sets which include the membership functions $uF4(y)$, $uF5(y)$ and $uF6(y)$, where y is the number of VNFs. Where, $uF4(y)$ is limited for y=100, $uF5(y)$ is good for y=300, and $uF6(y)$ is exceeded for y=500. As shown in Figure 4.3, $F7$, $F8$, and $F9$ are the fuzzy sets to model the cost constraint that is quantified by membership functions $uF7(z)$, $uF8(z)$, and $uF9(z)$, where z is the number of VMs. These sets are associated by the following terms: acceptable, profitable and expensive. $F10$, $F11$, and $F12$ are the fuzzy sets used to model the delay constraint of each VNF in SFC (D0) which is quantified by membership function $uF10(q)$, $uF11(q)$, and $uF12(q)$ where q is the number of VNFs. These sets are defined by the terms: low, moderate, and high. As shown in Figure 4.4, the capacity constraint is modeled by F13, F14, and F15 which are associated with the membership functions $uF13(t)$, $uF14(t)$, and $uF15(t)$, where t is the number of VMs. These sets are classified in 3 linguistic terms: sufficient, largely sufficient and non sufficient. Profit constraint presents the material gain of placement of VNF in the most suitable VM which is modeled by the fuzzy set $F16$, $F17$, $F18$ and associated with the membership functions $uF16(s)$, $uF17(s)$ and $uF18(s)$ where s

represents the output cost as illustrated in Figure 4.4. These memberships are identified by the following linguistic labels no profitable, limited, and profitable. The profit constraint induces three consequences for fuzzy sets $F19$, $F20$ and $F21$ which is expressed respectively by conter-offer, reject and accept.

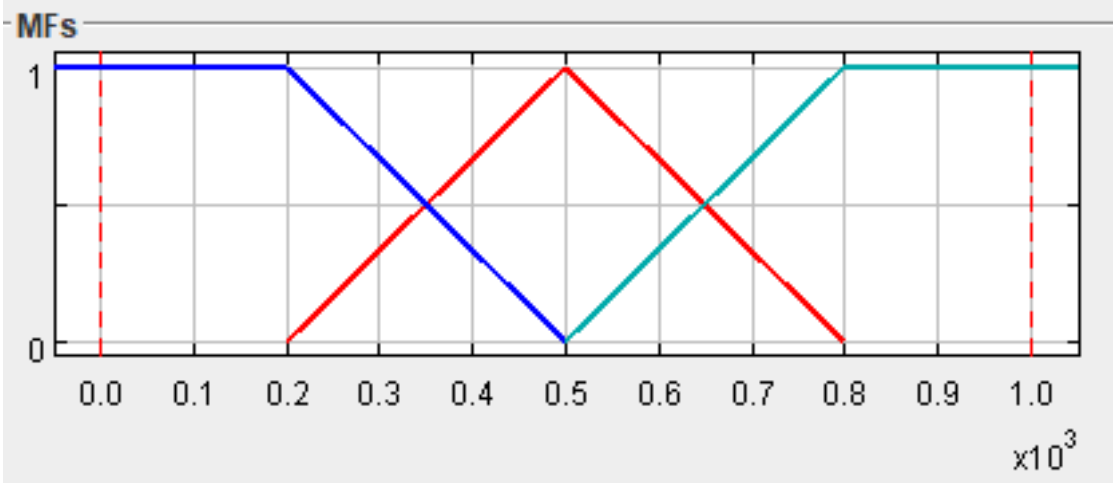

Figure 4.2: Delay memberships in term of VNF

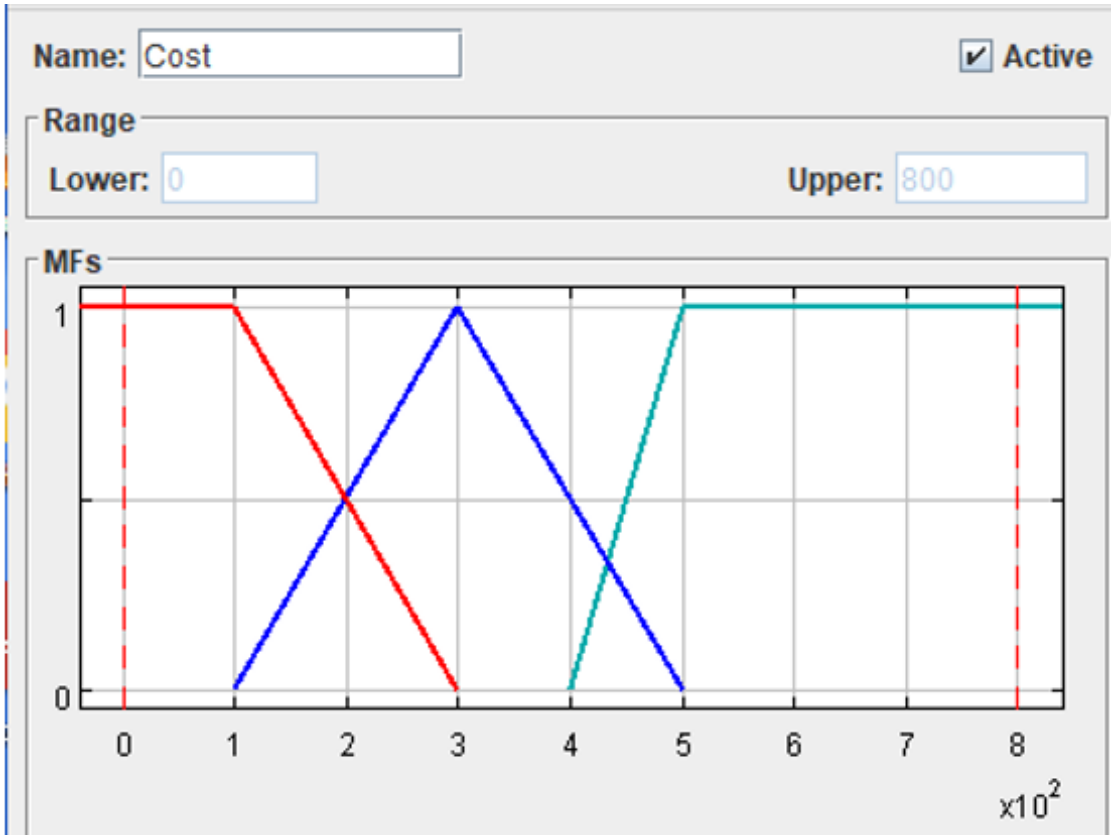

Figure 4.3: Cost memberships in term of VM

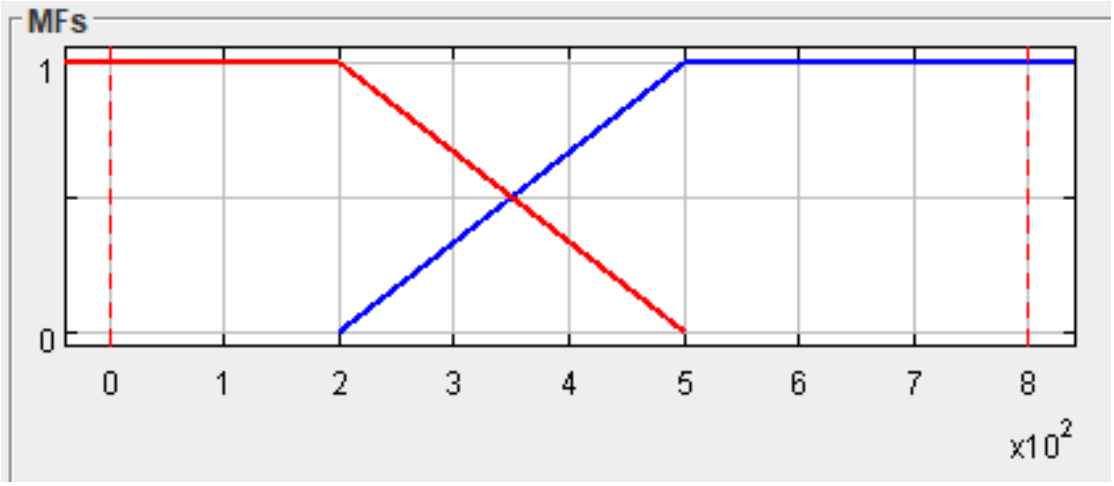

Figure 4.4: cCpacity memberships in term of VM

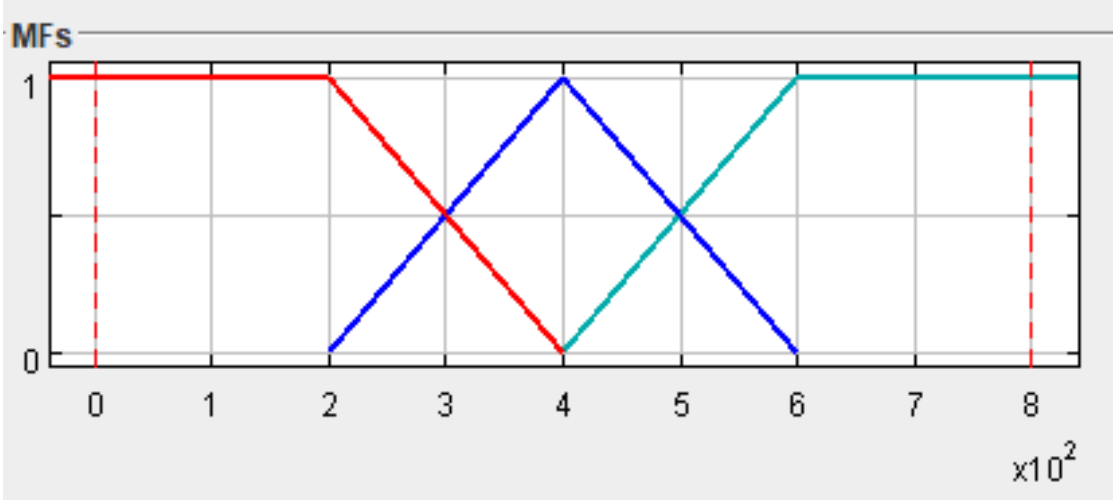

Figure 4.5: Cost output memberships in term of VM

4.3.0.1 Fuzzification

Fuzzification is the step of dividing input variables into categories that makes logical sense by determining the membership functions. These membership functions sets are determined and calculated according to the following equations:

$$uF1(x), uF4(y), uF7(z), uF10(q), uF13(t), uF16(s) = \begin{cases} 0 & if \quad x \succ d \\ \frac{d-x}{d-c} & if \quad c \leq x \leq d \\ 1 & if \quad x \prec c \end{cases} \tag{4.10}$$

$$uF2(x), uF5(y), uF8(z), uF11(q), uF14(t), uF17(s) = \begin{cases} 0 & if \quad x \leq 0 \\ \frac{b-x}{b-m} & if \quad m\ succx \succ b \\ 0 & if \quad x \geq b \end{cases} \tag{4.11}$$

$$uF3(x), uF6(y), uF9(z), uF12(q), uF15(t), uF18(s) = \begin{cases} 0 & if \quad x \prec a \\ \frac{x-a}{b-a} & if \quad a \leq x \leq b \\ 1 & if \quad x \succ b \end{cases} \tag{4.12}$$

where a and b are respectively the minimal and maximal of membership in the S-trapezoid shape. For triangular shape, a and b are the minimal membership functions and m is the maximal membership. In Z trapezoid shape, c is the minimal membership and d is the maximal membership.

Inference engine

FIS is a next step to apply the inference rules on the fuzzy input. It is used to generate fuzzy outputs using fuzzy set theory, if and then rules, fuzzy reasoning process. The rules of inference represent the computational function of the system which is based on the experiences and the previous observations and the role of experts. Each rule consists of two concepts; If contains the circumstances and Then contains the consequences. We can present the inference engine for VNF placement as follows:

R1:if uF1(x) and uF4(y) and uF7(z) and uF10(q) then F17

R2: if uF1(x) and uF4(y) and uF7(z) and uF11(q) then F17

R3: if uF1(x) and uF4(y) and uF8(z) and uF10(q) then F17

R4: if uF1(x) and uF4(y) and uF8(z) and uF11(q) then F17

R5: if uF1(x) and uF4(y) and uF9(z) and uF10(q) then F17

R6: if uF1(x) and uF4(y) and uF9(z) and uF11(q) then F17

R7: uF1(x) and uF5(y) and uF7(z) and uF10(q) then F17

R8: if uF1(x) and uF5(y) and uF7(z) and uF11(q) then F17

R9: if uF1(x) and uF5(y) and uF8(z) and uF10(q) then F17

R10: if uF1(x) and uF5(y) and uF8(z) and uF11(q) the F17

R11: if uF1(x) and uF5(y) and uF9(z) and uF10(q) the F17

R12: if uF1(x) and uF5(y) and uF9(z) and uF12(q) the F17

R13: if uF1(x) and uF6(y) and uF7(z) and uF10(q) then F17

R14: if uF1(x) and uF6(y) and uF7(z) and uF11(q) then F17

R15; if uF1(x) and uF6(y) and uF8(z) and uf10(q) the F17

R16: if uF1(x) and uF6(y) and uF8(z) and uF11(q) then F17

R17: if uF1(x) and uF6(y) and uF9(z) and uF10(q) then F17

R18: if uF1(x) and uF6(y) and uF9(z) and uF11(q) then F17

R19: if uF2(x) and uF4(y) and uF7(z) and uF10(q) then F17

R20: if uF2(x) and uF4(y) and uF7(z) and uF11(q) then F17

R21: if uF2(x) and uF4(y) and uF8(z) and uF10(q) then F17

R22: if uF2(x) and uF4(y) and uF8(z) and uF11(q) then F17

R23: if uF2(x) and uF4(y) and uF9(z) and uF10(q) then F17

R24:if uF2(x) and uF5(y) and uF9(z) and uF11(q) then F17

R25: if uF2(x) and uF5(y) and uF9(z) and uF10(q) then F17

R26: if uF2(x) and uF5(y) and uF7(z) and uF11(q) then F17

R27: if uF2(x) and uF5(y) and uF8(z) and uF10(q) then F17

R28: if uF2(x) and uF5(y) and uF8(z) and uF11(q) then F17

R29: if uF2(x) and uF5(y) and uF9(z) and uF10(q) then F17

R30: if uF2(x) and uF5(y) and uF9(z) and uF11(q) then F17

R31:if uF2(x) and uF6(y) and uF7(z) and uF10(q) then F17

R32: if uF2(x) and uF6(y) and uF7(z) and uF11(q) then F17

R33: if uF2(x) and uF6(y) and uF8(z) and uF10(q) then F17

R34:if uF2(x) and uF6(y) and uF8(z) and uF10(q) then F17

R35:if uF2(x) and uF6(y) and uF9(z) and uF10(q) then F17

R36: if uF2(x) and uF6(y) and uF9(z) and uF11(q) then F17

R37: if uF3(x) and uF4(y) and uF7(z) and uF10(q) then F17

R38: if uF3(x) and uF4(y) and uF7(z) and uF11(q) then F17

R39: if uF3(x) and uF4(y) and uF8(z) and uF10(q) then F17

R40: if uF3(x) and uF4(y) and uF8(z) and uF11(q) then F17

R41: if uF3(x) and uF4(y) and uF9(z) and uF10(q) then F17

R42: if uF3(x) and uF4(y) and uF9(z) and uF11(q) then F17

R43: if uF3(x) and uF5(y) and uF7(z) and uF10(q) then F17

R44: if uF3(x) and uF5(y) and uF7(z) and uF11(q) then F17

R45: if uF3(x) and uF5(y) and uF8(z) and uF10(q) then F17

R46: if uF3(x) and uF5(y) and uF8(z) and uF11(q) then F17

R47: if uF3(x) and uF5(y) and uF9(z) and uF10(q) then F17

R48: if uF3(x) and uF5(y) and uF9(z) and uF11(q) then F17

R49: if uF3(x) and uF6(y) and uF7(z) and uF10(q) then F17

R50: if uF3(x) and uF6(y) and uF7(z) and uF11(q) then F17

R51: if uF3(x) and uF6(y) and uF8(z) and uF10(q) then F17

R52: if uF3(x) and uF6(y) and uF8(z) and uF11(q) then F17

R53: if uF3(x) and uF6(y) and uF9(z) and uF10(q) then F17

R54: if uF3(x) and uF6(y) and uF9(z) and uF11(q) then F17

Defuzzification

After obtaining the inference membership functions outputs, we go to the defuzzification step to convert the fuzzy outputs to a crisp values. In this step, the fuzzy outputs of the fuzzification step are transformed to net values. Output value is specified by x=500, y=500 , z=400 and q=400 according to an expert determining the clear values of VMs in terms of capacity and bandwidth and VNF in terms of delay and cost. After the aggregation, we extract the control action by the following values:

u_{ca}=1,1,1, 0.5, 0,0,0,0,0,0

Finally, we apply center of gravity defuzzification to determine crisp output as follows:

$$Cr - output = \frac{\sum_{n=1}^{N} I_n u_n}{\sum_{n=1}^{N} u_n} \tag{4.13}$$

4.3.1 VNF chaining for SFC request problem

In this chaining step, we propose a deep reinforcement learning-based algorithm which uses MDP and LSTM in parallel to capture the current and historical environment transitions that it send to the agent for processing and making decision whether to execute a placement action or not according to the indicated conditions. If the action is *accepted* the environment returns a reward. If the action is *not accepted*, the agent does a back-track. Each SFC request is divided into an episode which includes the placement

of VNFs in a node. During training, the PG adaptation optimizes the SFC deployment quality based on the gradient calculated from the deployment estimates. The main objective is to set a policy that optimizes the ultimate reward at the conclusion of a series of state transitions. The policy is denoted by the formula: $\pi\Theta = P(a|s, \Theta)$. $\pi\Theta$ expresses the probability of action a in the state s under parameter Θ. Figure 4.6 shows two VNF chaining modes for an SFC request. In mode 1, two VNFs from different VM nodes are linked. In mode 2, the link is between two VNFs in the same VM node. Note that node 2 improves the quality of service and communication efficiency, at the cost of interference in power consumption and congestion at node level or at the link level as shown in the figure 4.6. Therefore, we should find an effective solution to placing and consolidating VNFs in a chain without exceeding SLA. Our approach deals with these issues and with the dynamic variations of the network in different time slots. A serialization and backtracking method is used to process a single VNF at each MDP and LSTM transition state. Our system moves to the processing of next VNF in SFC if the conditions are favorable, otherwise it returns to the previous network state. Our PBDRL approach processes each SFC request in different time slot. At every two time slots, two cases can be defined as:

(1) Intra time slot: when the requests arrive, the NFV system starts by, firstly, removing the timeout requests and refreshing the network state through freeing the resources occupied by these requests. Then, it processes these requests one after the other according to the time of their arrival. The agent processes the incoming network state and finds the VM node with sufficient resources to place the VNF. Afterwards, the agent executes the action of placing the VNF in the found VM node candidate. The system moves to the next state, the agent does the same processing until the last VNF of SFC is placed. If a VNF does not find the necessary requirements to be placed in a VM node, no action is taken and SFC is rejected. The system backtracks to the state s_t and releases the resource occupied by the previous placed VNFs. The following SFCs are treated in the same way.

(2) Inter time slots: when no SFC request arrived at two time slots, the system removes the timeout request and releases the occupied resources. In this case, no action is taken and no state transition is performed.

In the case where no action is to be taken, the state is not changed. When the new requests are received, a state transition occurs and the reward calculation is performed

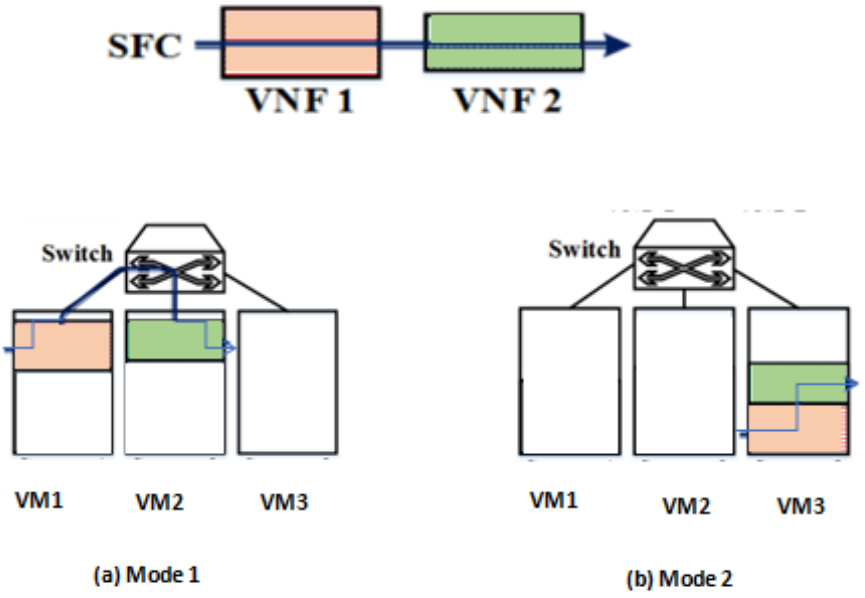

Figure 4.6: VNF consolidation

on acceptance or rejection over a series of time slots as follows:

$$R(s_T, a) = \begin{cases} -\sigma_T cost(s_T, a) + w_{r_i} P_{r_i}, & \text{si } r_i \text{ is accepted} \\ 0 & r_i \text{is rejected} \end{cases} \tag{4.14}$$

Where $w_r P_r$ is the throughput of SFC, and σ_T is a binary variable indicating whether the time slot changes after state s_t.

4.4 Implementation and experiment

In this section, we show the effectiveness of our VNF placement and chaining approach in cloud infrastructure in terms of minimizing the cost of physical and virtual resources, and delay and augmenting throughput. To implement our approach, we used a synthetic data-set which is randomly generated. This data-set consists of a number of VM nodes which varies from 100 to 1300, with [1000, 5000] of CPU resources and [10.000, 50.000] of memory. Each each node have [1, 500] units of CPU resource and [1, 64] units of memory. We also set the intra-pod delay between 40 to 100μs and inter-pod delay

between 50 to 200μs. We compared our approach with other solutions to properly assess its effectiveness based on 4 parameters which are: cost of resources expressed in Equation(3.16), reward expressed in Equation(3.15), and throughput expressed in Equation(3.17).

4.4.1 Parameter of SFC requests

In our simulation, we use from 24 to 1000 SFC requests. Each request consists of 1 to 7 VNFs among firewall, NAT, router, IDS, load balancer (LB), WAN optimizer and flow monitor. Also, each request requires a specific quality of service in time and throughput. We identify the service rate of each VNF between 100 and 1000. In each episode, we use from 1 to 60 time-slots with packet arrival rate from 1 to 100 packet/s.

4.4.2 Simulation setup

To carry out the experiments, we use the Cloud-Sim simulator with fuzzy logic toolbox. We use the Netbeans java language and the TensorFlow Java API library to build our NFV environment and deep neural network. For configuration, we simulate the NFV network topology based on 3 layers of switches.

4.4.3 Result and evaluation

Table 4.2: The operation cost

Algorithms / Number of migration	ILP	PBDRL	Hybrid algorithm
10	13	14	11.2
30	16	19	12.5
50	19	19.5	15.6
100	21	23	17
150	22.5	24	19
200	26.2	29	21.4

To evaluate the performance of our approach, we compare it with algorithms that include it such as ILP, deep reinforcement learning and PBDRL. We use ILP to provide SFC migration optimization for a service request. PBDRL uses the optimal solution obtained from the ILP for estimating the best solution for the SFC chaining and migration

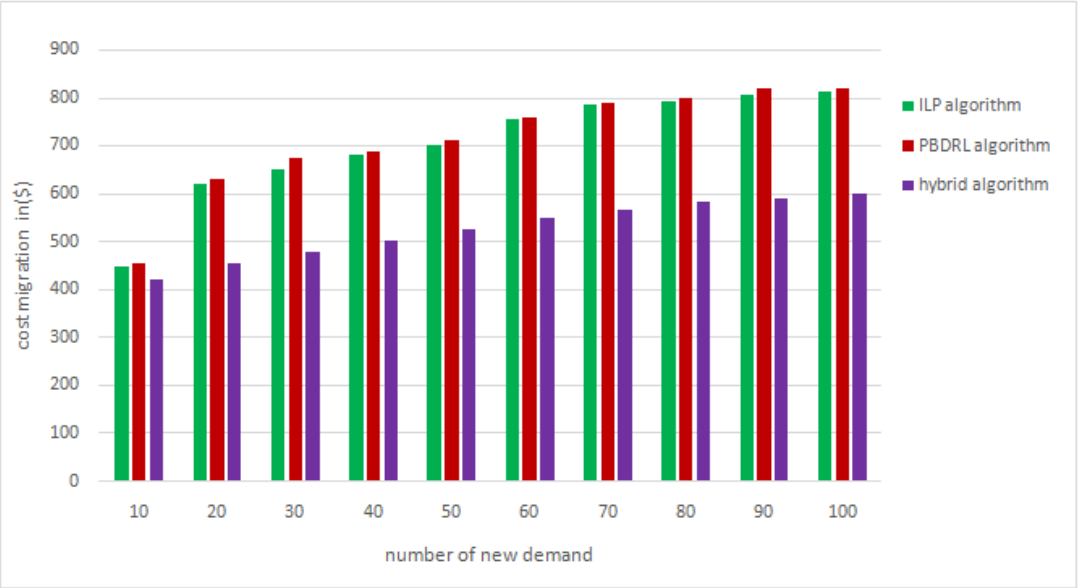

Figure 4.7: The comparison of the migration cost for our approach with ILP and PBDRL algorithm.

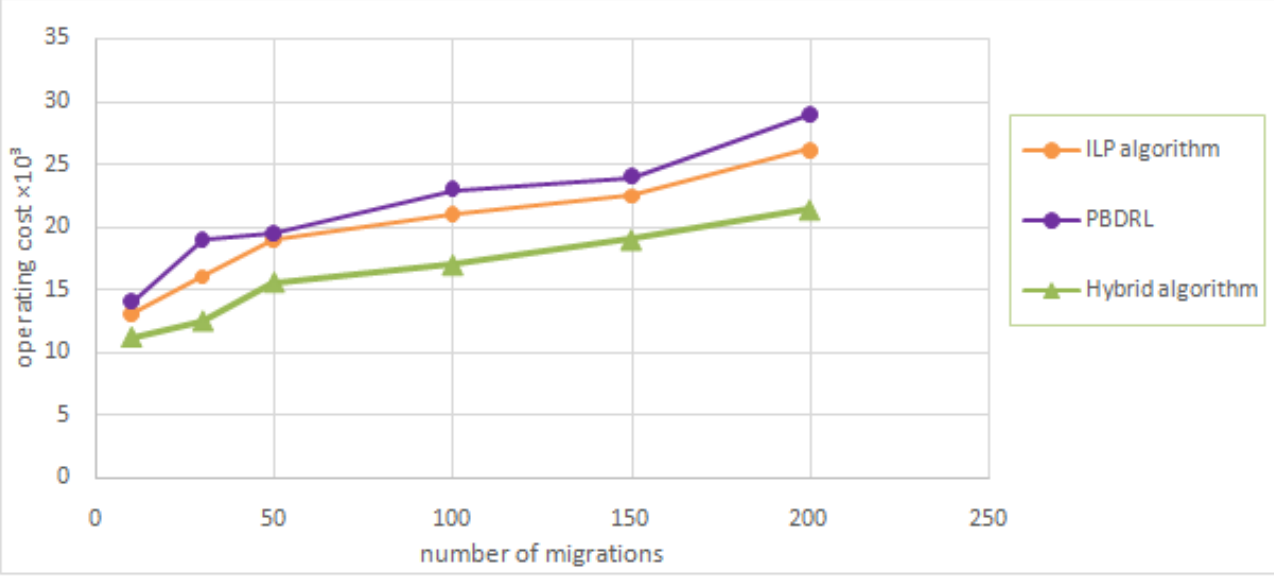

Figure 4.8: The comparison of the operation cost for our approach with ILP and PBDRL algorithm.

problem.

A.Topologies of network: We evaluate our approach on 6 topologies[107]. These topologies consist of synthetic network Barabási-Albert (BA), Waxman (WA) and Erdős-Rényi (ER) models. BA is generated by 4 primary nodes and 4 links. WA is created with density probability =0.7. ER is created with probability generation limit=0.2. We identify the small topologies by BAS, WAS, ERS and the large topologies by BAL, WAL,

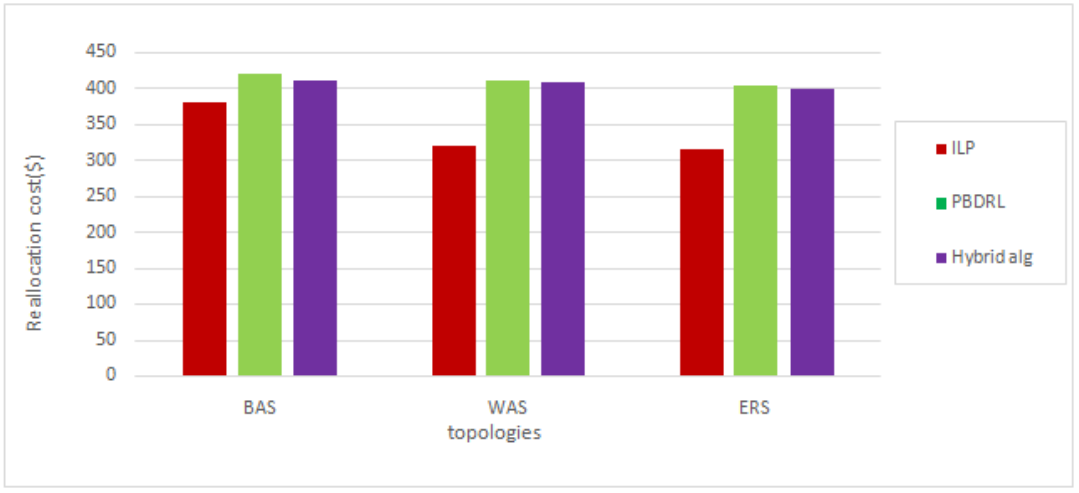

Figure 4.9: Reallocation cost of our approach in term of different topology.

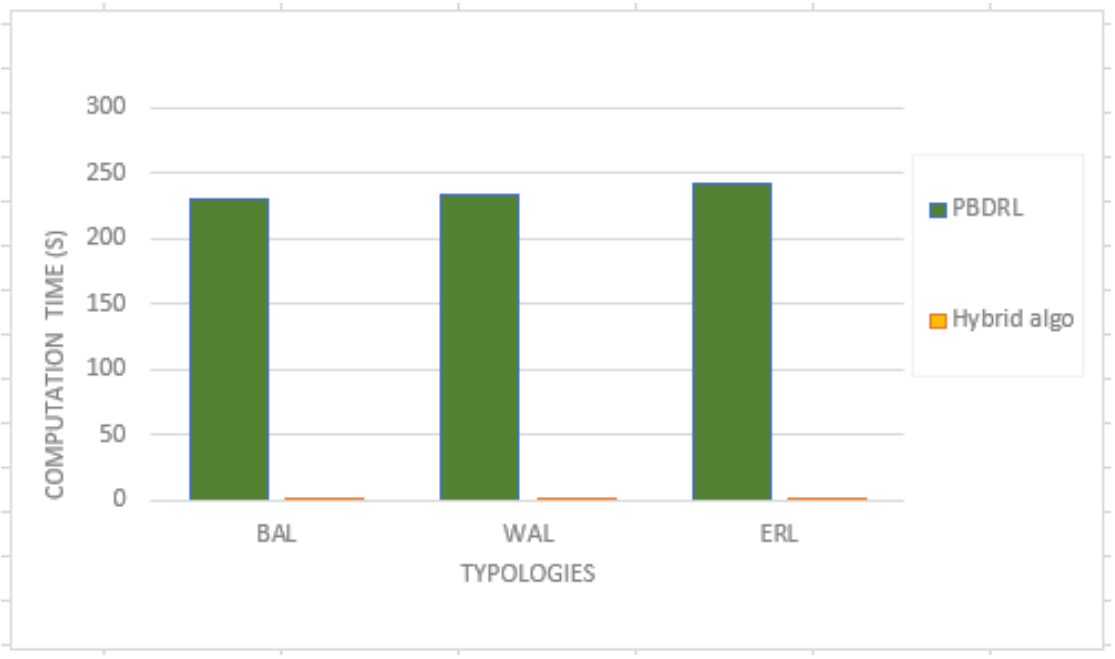

Figure 4.10: Computation time for our hybrid approach

ERL.

B. Performance of our approach: Figure 4.7 shows that ILP realizes a migration cost optimization greater than PBDRL algorithm and Hybrid algorithm regardless of the number of request changes. Also, this figure reflects the overload impact on the cost of migration and reallocation during the arrival of new request. In figure 4.8, we observe that the operation cost of hybrid algorithm is lower than ILP and PBDRL. The operating cost increases when the number of migrations increases, while it is lower when the number of migrations is limited because the operating cost increases when the cost

of deployment increases in case of migration is rare. As illustrated in Table 4.2, this cost varies between 11.2 and 21.4 for the hybrid algorithm and between 14 and 29 for PBDRL.

Figure 4.9 shows that ILP achieves the best reallocation cost optimization for ERS, WAS and BAS topology respectively whereas the hybrid algorithm ensures the lowest computation time for large topology, as shown in Figure 4.10. This can be explained by the role and ability of the different algorithms to provide the solution which facilitates and accelerates the training of reinforcement algorithm and to minimize the computation time.

Table 4.3: Table of average SFC delay in three typologies

Algorithm	ERS	BAS	WAS
Hybrid algorithm	29	30	27
ILP	27	29	26
Before migration	24	26	24

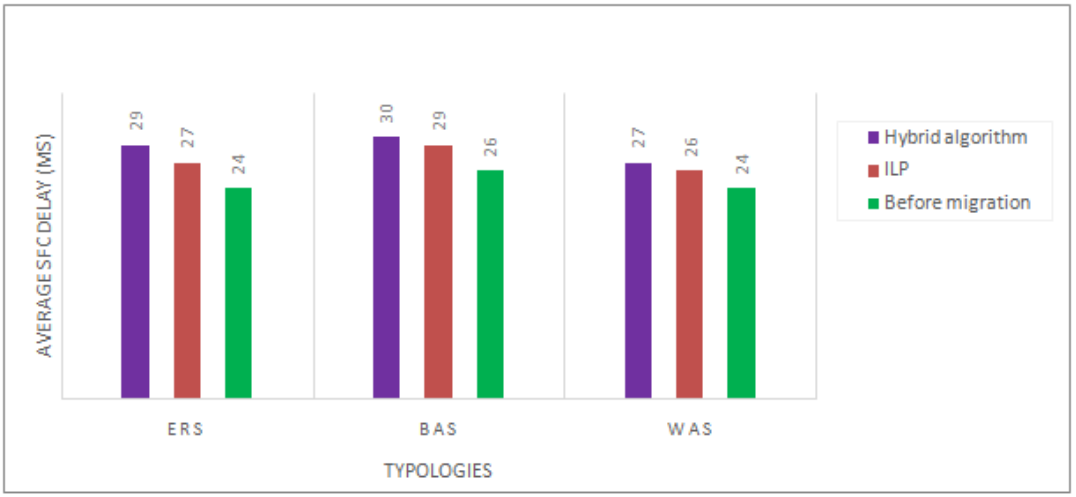

Figure 4.11: Average SFC delay in three topologies

In terms of delay, Figure 4.11 and Table 4.3 show that our algorithm has the largest efficiency in reducing the SFC delay before the migration which decreases slightly after the migration in 3 different typologies. This increase is explained by the complexity of the hybrid algorithm which is based on neural network to find the best placement and chaining of VNFs, as well as time taken for training and policy. Also, we observe that ILP realizes more optimized delay than the hybrid algorithm after migration which confirms their importance in the minimization of delay in the hybrid algorithm

4.5 Discussion

We can deduce that our approach is a combination of two complementary algorithms Fuzzy-FCA and PBDRL, each dedicated to specific objectives. The Fuzzy-FCA algorithm aims to minimize the number of active virtual machines, the cost and the latency, to balance and maximize the exploitation of resources in order to minimize the energy consumption. While the PBDRL algorithm aims to find the best SFC request path, minimize delay and throughput taking into account bandwidth availability, congestion and VNF instance multiplicity. However, this combination negatively affects the efficiency of some algorithms, for example, the efficiency of ILP weakens in terms of delay and computation time when combined in the hybrid algorithm as illustrated in figure 4.6 and figure 4.7.

4.6 Conclusion

In this work, we tackled the VNF placement problem considering chaining and migration as two factors that can improve the resource utilization while reducing the delay and augmenting the throughput. We have proposed a hybrid algorithm which consists in a combination of Fuzzy-FCA algorithm and PBDRL algorithm. We first used Fuzzy-FCA to extract the most functional VNFs in the requested network service and consolidate them into the smallest number of VMs while respecting the resource capacity constraints. Then we applied the PBDRL algorithm to find the best path to place and deploy chained VNFs for an SFC request taking while avoiding congestion, increasing throughput, and reducing delay. The evaluation results showed that our hybrid approach improves the performance, particularly migration cost, computation time and delay for different topologies and scales.

Chapter 5

Conclusion and future research directions

5.1 Introduction

This chapter presents a general summary of the most important contributions of this thesis and discusses the work that will carry it out in the future. In Section 5.2, we recall the contributions and objectives achieved during this thesis. In Section 5.3, we present some ideas and proposals for future works.

5.2 Conclusion and discussion

In this thesis, we tackled the VNF placement problem gradually. At first, we treated it as a backpack problem and worked to find the placement of VNFs according to the conditions and constraints that may be encountered. In the second step, we extended the placement problem and treat it considering the VNF chaining. The objective is to put the placement problem in parallel with the chaining problem to obtain the exact and suitable placement for each VNF in terms of resource exploitation and in terms of proper functioning since both are dependent on each other. In this case, the placement of VNFs should follow a predefined order of the VNFs which constitute an SFC and a specified network service. At first, we studied the problem of placement of VNFs in all dimensions and ramifications arriving at their extension as an SFC. We have classified it according to

several axes and algorithms used to process it and also in terms of existing technologies. From this study, we determined the limits, weak points and the most important missing in the previous work. To fill the gap, we proposed various approaches aiming at solving the placement, consolidation and chaining of VNFs. As a first contribution, we dealt with the VNF placement problem without considering the VNF chaining. We proposed two successive algorithms, namely FCA and Fuzzy FCA. FCA is used to group and place VNFs for a network service, whereas Fuzzy FCA is used to consolidate the VNF candidate which are extracted from FCA process into the most appropriate VMs. Our solution aims at minimizing the number of active virtual machines, energy consumption, latency and resource cost while balancing the exploitation of resources in cloud data centers. As an extension work, we tackled the problem of placement and consolidation of VNFs by considering the chaining relationship between the different VNFs which should follow a predefined order. We thus proposed a new approach based on deep reinforcement learning using two modules LSTM and MDP in parallel to capture the state of NFV environment transitions. LSTM is used to capture historical transition states and MDP is used to capture current transition states. An agent processes these states on the basis of policy gradient VNF placement in VM node (the action is executed if the conditions are favorable. The approach succeeded in following a smart strategy able to estimate the best route for the SFC request in which each VNF placed in the VM node meets the requirements of this node (otherwise back track is executed). We also dealt with the problem of migration to improve the VNF placement and chaining. In this context, we have proposed the hybridization of Fuzzy-FCA and PBDRL algorithms in order to minimize the cost and the end-to-end delay. All of our proposals are validated by simulation showing the enhancements achieved with our approaches compared to the existing works.

5.2.1 Future research directions

Based on the results obtained during this thesis, we propose here some ideas and research directions to complete our study:

1. VNF consolidation, reoptimize the VNF placement off-line. Generally, the VNF placement and consolidation is done on-line. The arrival and departure of SFCs modify the resource availability. In this way, reoptimizing in an off-line manner the placement of VNF which incurs the migration of VNFs is a challenging problem.

2. Protection of SFCs against the failures. To cope with the failure, the VNF can replicated. In this way, various by pass routes should be configured so that the requirements are respected even after failures.

Bibliography

[1] Shankar Lal, Tarik Taleb, and Ashutosh Dutta. Nfv: Security threats and best practices. *IEEE Communications Magazine*, 55(8):211–217, 2017.

[2] Bo Yi, Xingwei Wang, Keqin Li, Min Huang, et al. A comprehensive survey of network function virtualization. *Computer Networks*, 133:212–262, 2018.

[3] Dandan Qi, Subin Shen, and Guanghui Wang. Virtualized network function consolidation based on multiple status characteristics. *IEEE Access*, 7:59665–59679, 2019.

[4] Daewoong Cho, Javid Taheri, Albert Y Zomaya, and Pascal Bouvry. Real-time virtual network function (vnf) migration toward low network latency in cloud environments. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 798–801. IEEE, 2017.

[5] NFVISG ETSI. Gs nfv-man 001 v1. 1.1 network function virtualisation (nfv); management and orchestration, 2014.

[6] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications surveys & tutorials*, 18(1):236–262, 2015.

[7] Mohammad Shojafar, Claudia Canali, Riccardo Lancellotti, and Enzo Baccarelli. Minimizing computing-plus-communication energy consumptions in virtualized networked data centers. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 1137–1144. IEEE, 2016.

[8] Zaki Brahmi and Faten Ben Hassen. Communication-aware vm consolidation based on formal concept analysis. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE, 2016.

[9] Qixia Zhang, Fangming Liu, and Chaobing Zeng. Adaptive interference-aware vnf placement for service-customized 5g network slices. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 2449–2457. IEEE, 2019.

[10] Oussama Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. Energy efficient algorithm for vnf placement and chaining. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 579–588. IEEE, 2017.

[11] Wided Khemili, Jalel Eddine Hajlaoui, and Mohamed Nazih Omri. Energy aware fuzzy approach for placement and consolidation in cloud data centers. *Journal of Parallel and Distributed Computing*, 161:130–142, 2022.

[12] Leila Helali and Mohamed Nazih Omri. A survey of data center consolidation in cloud computing systems. *Computer Science Review*, 39:100366, 2021.

[13] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N Calheiros, and Rajkumar Buyya. Virtual machine consolidation in cloud data centers using aco metaheuristic. In *European conference on parallel processing*, pages 306–317. Springer, 2014.

[14] Yongqiang Gao, Haibing Guan, Zhengwei Qi, Yang Hou, and Liang Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of computer and system sciences*, 79(8):1230–1242, 2013.

[15] Christina Terese Joseph, K Chandrasekaran, and Robin Cyriac. A novel family genetic approach for virtual machine allocation. *Procedia Computer Science*, 46: 558–565, 2015.

[16] Maolin Tang and Shenchen Pan. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural processing letters*, 41 (2):211–221, 2015.

[17] Moreno Marzolla, Ozalp Babaoglu, and Fabio Panzieri. Server consolidation in clouds through gossiping. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–6. IEEE, 2011.

[18] Jason Nikolai and Yong Wang. Hypervisor-based cloud intrusion detection system. In *2014 International Conference on Computing, Networking and Communications (ICNC)*, pages 989–993. IEEE, 2014.

[19] Ilias Mavridis and Helen Karatza. Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing. *Future Generation Computer Systems*, 94:674–696, 2019.

[20] Junzo Watada, Arunava Roy, Ruturaj Kadikar, Hoang Pham, and Bing Xu. Emerging trends, techniques and open issues of containerization: a review. *IEEE Access*, 7:152443–152472, 2019.

[21] Abdelhamid Alleg. *Service Function Placement and Chaining in Network Function Virtualization Environments.* PhD thesis, Bordeaux, 2019.

[22] Fethi Fkih and Mohamed Nazih Omri. Irafca: an o (n) information retrieval algorithm based on formal concept analysis. *Knowledge and Information Systems*, 48(2):465–491, 2016.

[23] Quan Thanh Tho, Siu Cheung Hui, Alvis Cheuk M Fong, and Tru Hoang Cao. Automatic fuzzy ontology generation for semantic web. *IEEE transactions on knowledge and data engineering*, 18(6):842–856, 2006.

[24] Fatiha Naouar, Lobna Hlaoua, and Mohamed Nazih Omri. Information retrieval model using uncertain confidence's network. *International Journal of Information Retrieval Research (IJIRR)*, 7(2):34–50, 2017.

[25] Sedef Demirci and Seref Sagiroglu. Optimal placement of virtual network functions in software defined networks: A survey. *Journal of Network and Computer Applications*, 147:102424, 2019.

[26] Network Functions Virtualisation. Etsi industry specification group (isg),"etsi gs nfv-man 001 v1. 1.1: Network functions virtualisation (nfv); management and orchestration,", 2014.

[27] Abdelhamid Alleg, Riad Kouah, Samira Moussaoui, and Toufik Ahmed. Virtual network functions placement and chaining for real-time applications. In *2017 IEEE 22nd international workshop on computer aided modeling and design of communication links and networks (CAMAD)*, pages 1–6. IEEE, 2017.

[28] Fatma Ben Jemaa. *Design and optimization of next-generation carrier-grade wi-fi networks.* PhD thesis, Université Pierre et Marie Curie-Paris VI, 2016.

[29] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13 (3):518–532, 2016.

[30] Sean C Mondesire, Anastasia Angelopoulou, Shehan Sirigampola, and Brian Goldiez. Combining virtualization and containerization to support interactive games and simulations on the cloud. *Simulation Modelling Practice and Theory*, 93:233–244, 2019.

[31] Zuo Xiang, Frank Gabriel, Elena Urbano, Giang T Nguyen, Martin Reisslein, and Frank HP Fitzek. Reducing latency in virtual machines: Enabling tactile internet for human-machine co-working. *IEEE Journal on Selected Areas in Communications*, 37(5):1098–1116, 2019.

[32] Gao Zheng, Anthony Tsiopoulos, and Vasilis Friderikos. Dynamic placement of vnf chains for proactive caching in mobile edge networks, 2018.

[33] Yong Li and Min Chen. Software-defined network function virtualization: A survey. *IEEE Access*, 3:2542–2553, 2015.

[34] Sevil Mehraghdam, Matthias Keller, and Holger Karl. Specifying and placing chains of virtual network functions. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 7–13. IEEE, 2014.

[35] Evgenia Kapassa, Marios Touloupou, and Dimosthenis Kyriazis. Slas in 5g: A complete framework facilitating vnf-and ns-tailored slas management. In *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 469–474. IEEE, 2018.

[36] Taekhee Kim, Siri Kim, Kwonyong Lee, and Sungyong Park. A qos assured network service chaining algorithm in network function virtualization architecture. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 1221–1224. IEEE, 2015.

[37] Khaled Alwasel, Rodrigo N Calheiros, Saurabh Garg, Rajkumar Buyya, Mukaddim Pathan, Dimitrios Georgakopoulos, and Rajiv Ranjan. Bigdatasdnsim: A simulator for analyzing big data applications in software-defined cloud data centers. *Software: Practice and Experience*, 51(5):893–920, 2021.

[38] Mohammad Shojafar, Claudia Canali, Riccardo Lancellotti, and Enzo Baccarelli. Minimizing computing-plus-communication energy consumptions in virtualized networked data centers. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 1137–1144, 2016. doi: 10.1109/ISCC.2016.7543890.

[39] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[40] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[41] Tiago C Ferreto, Marco AS Netto, Rodrigo N Calheiros, and César AF De Rose. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, 27(8):1027–1034, 2011.

[42] Waltenegus Dargie. Estimation of the cost of vm migration. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE, 2014.

[43] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *ACM/IFIP/USENIX international conference on distributed systems platforms and open distributed processing*, pages 243–264. Springer, 2008.

[44] Gourav Prateek Sharma, Wouter Tavernier, Didier Colle, and Mario Pickavet. Vnf-aap: Accelerator-aware virtual network function placement. In *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–4. IEEE, 2019.

[45] Marco Savi, Massimo Tornatore, and Giacomo Verticale. Impact of processing-resource sharing on the placement of chained virtual network functions. *IEEE Transactions on Cloud Computing*, 9:1479–1492, 2019.

[46] Tanasak Janpan, Vasaka Visoottiviseth, and Ryousei Takano. A virtual machine consolidation framework for cloudstack platforms. In *The International Conference on Information Networking 2014 (ICOIN2014)*, pages 28–33. IEEE, 2014.

[47] Abdelquoddouss Laghrissi and Tarik Taleb. A survey on the placement of virtual resources and virtual network functions. *IEEE Communications Surveys & Tutorials*, 21(2):1409–1434, 2018.

[48] Satyam Agarwal, Francesco Malandrino, Carla Fabiana Chiasserini, and Swades De. Vnf placement and resource allocation for the support of vertical services in 5g networks. *IEEE/ACM Transactions on Networking*, 27(1):433–446, 2019.

[49] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53(17):2923–2938, 2009.

[50] Mayank Mishra and Anirudha Sahoo. On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 275–282. IEEE, 2011.

[51] Anton Beloglazov and Rajkumar Buyya. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. *MGC@ Middleware*, 4(10.1145):1890799–1890803, 2010.

[52] Xin Li, Zhuzhong Qian, Ruiqing Chi, Bolei Zhang, and Sanglu Lu. Balancing resource utilization for continuous virtual machine requests in clouds. In *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 266–273. IEEE, 2012.

[53] Xin Li, Zhuzhong Qian, Sanglu Lu, and Jie Wu. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. *Mathematical and Computer Modelling*, 58(5-6):1222–1235, 2013.

[54] Dewang Gedia and Levi Perigo. Performance evaluation of sdn-vnf in virtual machine and container. In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–7. IEEE, 2018.

[55] Eugen Feller, Louis Rilling, and Christine Morin. Energy-aware ant colony based workload placement in clouds. In *2011 IEEE/ACM 12th International Conference on Grid Computing*, pages 26–33. IEEE, 2011.

[56] Tao Wen, Hongfang Yu, Gang Sun, and Liu Liu. Network function consolidation in service function chaining orchestration. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

[57] Zoltán Ádám Mann. Resource optimization across the cloud stack. *IEEE Transactions on Parallel and Distributed Systems*, 29(1):169–182, 2017.

[58] Jialei Liu, Shangguang Wang, Ao Zhou, Jinliang Xu, and Fangchun Yang. Sla-driven container consolidation with usage prediction for green cloud computing. *Frontiers of Computer Science*, 14(1):42–52, 2020.

[59] Djillali Boukhelef, Jalil Boukhobza, Kamel Boukhalfa, Hamza Ouarnoughi, and Laurent Lemarchand. Optimizing the cost of dbaas object placement in hybrid storage systems. *Future Generation Computer Systems*, 93:176–187, 2019.

[60] Mohamed K Hussein, Mohamed H Mousa, and Mohamed A Alqarni. A placement architecture for a container as a service (caas) in a cloud environment. *Journal of Cloud Computing*, 8(1):1–15, 2019.

[61] Tao Shi, Hui Ma, and Gang Chen. Energy-aware container consolidation based on pso in cloud data centers. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.

[62] Alain Tchana, Noel De Palma, Ibrahim Safieddine, and Daniel Hagimont. Software consolidation as an efficient energy and cost saving solution. *Future Generation Computer Systems*, 58:1–12, 2016.

[63] Louiza Yala, Pantelis A Frangoudis, and Adlen Ksentini. Latency and availability driven vnf placement in a mec-nfv environment. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[64] Asma Omri, Karim Benouaret, Djamal Benslimane, and Mohamed Nazih Omri. Towards an understanding of cloud services under uncertainty: A possibilistic approach. *International Journal of Approximate Reasoning*, 98:146–162, 2018.

[65] Aleksey Buzmakov and Amedeo Napoli. How fuzzy fca and pattern structures are connected? In *5th Workshop" What can FCA do for Artificial Intelligence?"(FCA4AI'2016)*, 2016.

[66] Marcelo Caggiani Luizelli, Leonardo Richter Bays, Luciana Salete Buriol, Marinho Pilla Barcellos, and Luciano Paschoal Gaspary. Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 98–106. IEEE, 2015.

[67] Enfeng Yang, Yong Zhang, Lei Wu, Yulong Liu, and Shijun Liu. A hybrid approach to placement of tenants for service-based multi-tenant saas application. In *2011 IEEE Asia-Pacific Services Computing Conference*, pages 124–130. IEEE, 2011.

[68] Mahzabeen Emu, Peizhi Yan, and Salimur Choudhury. Latency aware vnf deployment at edge devices for iot services: An artificial neural network based approach. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2020.

[69] Hajlaoui Jalel Eddine, Mohamed Nazih Omri, and Djamal Benslimane. Performance and scalability appraisal of four directed weighted graph matching algorithms: A survey. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 392–398. IEEE, 2017.

[70] Fei Hao, Guangyao Pang, Zheng Pei, Keyun Qin, Yu Zhang, and Xiaoming Wang. Virtual machines scheduling in mobile edge computing: a formal concept analysis approach. *IEEE Transactions on Sustainable Computing*, 5(3):319–328, 2019.

[71] Rihab Derouiche, Zaki Brahmi, and Mohamed Mohsen Gammoudi. Fca-based energy aware-data placement strategy for intensive workflow in cloud computing. *Procedia Computer Science*, 159:387–397, 2021.

[72] Zaki Brahmi, Sahar Mili, and Rihab Derouiche. Data placement strategy for massive data applications based on fca approach. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE, 2016.

[73] Xing Wu, Jing Duan, Mingyu Zhong, Peng Li, and Jianjia Wang. Vnf chain placement for large scale iot of intelligent transportation. *Sensors*, 20(14):3819, 2020.

[74] Aris Leivadeas, George Kesidis, Mohamed Ibnkahla, and Ioannis Lambadaris. Vnf placement optimization at the edge and cloud. *Future Internet*, 11(3):69, 2019.

[75] Marwa Mokni, Sonia Yassa, Jalel Eddine Hajlaoui, Rachid Chelouah, and Mohamed Nazih Omri. Cooperative agents-based approach for workflow scheduling on fog-cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–20, 2021.

[76] Carla Mouradian, Somayeh Kianpisheh, Mohammad Abu-Lebdeh, Fereshteh Ebrahimnezhad, Narjes Tahghigh Jahromi, and Roch H Glitho. Application component placement in nfv-based hybrid cloud/fog systems with mobile fog nodes. *IEEE Journal on Selected Areas in Communications*, 37(5):1130–1143, 2019.

[77] Paridhika Kayal and Jörg Liebeherr. Autonomic service placement in fog computing. In *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–9. IEEE, 2019.

[78] Paridhika Kayal and Jörg Liebeherr. Distributed service placement in fog computing: An iterative combinatorial auction approach. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 2145–2156. IEEE, 2019.

[79] Lidia Ruiz, Ramón J Durán, Ignacio De Miguel, Pouria S Khodashenas, Jose-Juan Pedreno-Manresa, Noemí Merayo, Juan C Aguado, Pablo Pavon-Marino, Shuaib Siddiqui, Javier Mata, et al. A genetic algorithm for vnf provisioning in nfv-enabled cloud/mec ran architectures. *Applied Sciences*, 8(12):2614, 2018.

[80] Qixia Zhang, Fangming Liu, and Chaobing Zeng. Online adaptive interference-aware vnf deployment and migration for 5g network slice. *IEEE/ACM Transactions on Networking*, 29(5):2115–2128, 2021. doi: 10.1109/TNET.2021.3080197.

[81] Wenrui Ma, Oscar Sandoval, Jonathan Beltran, Deng Pan, and Niki Pissinou. Traffic aware placement of interdependent nfv middleboxes. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[82] Ayaz Ali Khan, Muhammad Zakarya, Rajkumar Buyya, Rahim Khan, Mukhtaj Khan, and Omer Rana. An energy and performance aware consolidation technique for containerized datacenters. *IEEE Transactions on Cloud Computing*, 9:1305–1322, 2019.

[83] Dejene Boru Oljira, Karl-Johan Grinnemo, Javid Taheri, and Anna Brunstrom. A model for qos-aware vnf placement and provisioning. In *2017 IEEE Conference*

on Network Function Virtualization and Software Defined Networks (NFV-SDN), pages 1–7. IEEE, 2017.

[84] Xi Chen, Zonghang Li, Yupeng Zhang, Ruiming Long, Hongfang Yu, Xiaojiang Du, and Mohsen Guizani. Reinforcement learning–based qos/qoe-aware service function chaining in software-driven 5g slices. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3477, 2018.

[85] Sabrine Amri, Zaki Brahmi, Rocío Pérez de Prado, Sebastian García-Galán, Jose Enrique Muñoz-Expósito, and Adam Marchewka. Interference-aware virtual machine placement: A survey. In *International Conference on Image Processing and Communications*, pages 237–244. Springer, 2018.

[86] Sarra Ouni, Fethi Fkih, and Mohamed Nazih Omri. Toward a new approach to author profiling based on the extraction of statistical features. *Social Network Analysis and Mining*, 11(1):1–16, 2021.

[87] Irian Leyva-Pupo, Cristina Cervelló-Pastor, and Alejandro Llorens-Carrodeguas. The resources placement problem in a 5g hierarchical sdn control plane. In *International Symposium on Distributed Computing and Artificial Intelligence*, pages 370–373. Springer, 2018.

[88] Stuart Clayman, Elisa Maini, Alex Galis, Antonio Manzalini, and Nicola Mazzocca. The dynamic placement of virtual network functions. In *2014 IEEE network operations and management symposium (NOMS)*, pages 1–9. IEEE, 2014.

[89] Walter Cerroni and Franco Callegati. Live migration of virtual network functions in cloud-based edge networks. In *2014 IEEE International Conference on Communications (ICC)*, pages 2963–2968. IEEE, 2014.

[90] Eric Masanet. The energy efficiency potential of cloud-based software: A us case study. 3:1–5, 2013.

[91] Chaobing Zeng, Fangming Liu, Shutong Chen, Weixiang Jiang, and Miao Li. Demystifying the performance interference of co-located virtual network functions. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 765–773. IEEE, 2018.

[92] Marwa Mokni, Jalel Eddine Hajlaoui, and Zaki Brahmi. Mas-based approach for scheduling intensive workflows in cloud computing. In *2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE.

[93] Prem Kumar Singh, Cherukuri Aswani Kumar, and Abdullah Gani. A comprehensive survey on formal concept analysis, its research trends and applications. *International Journal of Applied Mathematics and Computer Science*, 26(2):495–516, 2016.

[94] Xiaoyu Wu, Jianying Wang, Li Shi, Yong Gao, and Yu Liu. A fuzzy formal concept analysis-based approach to uncovering spatial hierarchies among vague places extracted from user-generated data. *International Journal of Geographical Information Science*, 33(5):991–1016, 2019.

[95] Haithem Mezni and Taher Abdeljaoued. A cloud services recommendation system based on fuzzy formal concept analysis. *Data & Knowledge Engineering*, 116: 100–123, 2018.

[96] Abner Brito, Laécio Barros, Estevao Laureano, Fábio Bertato, and Marcelo Coniglio. Fuzzy formal concept analysis. In *North American Fuzzy Information Processing Society Annual Conference*, pages 192–205. Springer, 2018.

[97] Hongliang Lai and Dexue Zhang. Concept lattices of fuzzy contexts: Formal concept analysis vs. rough set theory. *International Journal of Approximate Reasoning*, 50(5):695–707, 2009.

[98] Mohamed Nazih Omri. Fuzzy ontology-based querying user's requests under uncertain environment. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 14(3):41–59, 2020.

[99] Xuewen Xia, Ling Gui, and Zhi-Hui Zhan. A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting. *Applied Soft Computing*, 67:126–140, 2018.

[100] Sepp Hochreiter. Ja1 4 rgen schmidhuber (1997).“long short-term memory”. *Neural Computation*, 9(8).

[101] Jalel Eddine Hajlaoui, Mohamed Nazih Omri, Djamal Benslimane, and Mahmoud Barhamgi. Qos based framework for configurable iaas cloud services discovery. In *2017 IEEE international conference on web services (ICWS)*, pages 460–467. IEEE, 2017.

[102] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

[103] Ehsan Etezadi, Carlos Natalino, Renzo Diaz, Anders Lindgren, Stefan Melin, Lena Wosinska, Paolo Monti, and Marija Furdek. Deepdefrag: a deep reinforcement learning framework for spectrum defragmentation. 2022.

[104] Bernardetta Addis, Dallal Belabed, Mathieu Bouet, and Stefano Secci. Virtual network functions placement and routing optimization. In *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pages 171–177. IEEE, 2015.

[105] Meihui Gao, Bernardetta Addis, Mathieu Bouet, and Stefano Secci. Optimal orchestration of virtual network functions. *Computer Networks*, 142:108–127, 2018.

[106] Ruben Solozabal, Josu Ceberio, Aitor Sanchoyerto, Luis Zabala, Bego Blanco, and Fidel Liberal. Virtual network function placement optimization with deep reinforcement learning. *IEEE Journal on Selected Areas in Communications*, 38 (2):292–303, 2019.

[107] Mikhail Drobyshevskiy and Denis Turdakov. Random graph modeling: A survey of the concepts. *ACM computing surveys (CSUR)*, 52(6):1–36, 2019.

List of Publications

Full papers:

- Wided Khemili, Jalel Eddine Hajlaoui, Mohamed Nazih Omri, "Energy Aware Fuzzy Approach for Placement and Consolidation in Cloud Data Centers".
Journal of Parallel and Distributed Computing, Elseiver, 2022. **Q1**, IF=4.54 (2022)

Submitted:

- Wided khemili, Jalel Eddine Hajlaoui, Mohamed Nazih OMri, "A Survey of Consolidation and Placement Investigation to Optimize Resources and Power Consumption in Cloud Environment", Journal of "Engineering Applications of Artificial Intelligence".

- Wided Khemili, Mohand Yazid Saidi,Jalel Eddine Hajlaoui, Ken Chen, Mohamed Nazih Omri, "Parallel Bi-State Deep Reinforcement Learning Approach for SFC Placements and Deployments", journal of "Neural Computing And Application".

- Wided Khemili, Jalel Eddine Hajlaoui, Mohand Yazid Saidi, Mohamed Nazih Omri, Ken CHEN "Deep Reinforcement Learning for VNF placement and chaining of Cloud Network services", 37-th International Conference on Advanced Information Networking and Applications 2023.