C02029: Doctor of Philosophy *CRICOS Code: 00099F*

Subject Code: C02029

February 2020

Resource Allocation and Optimal Scheduling of Virtual Network Functions in Software Defined Networks

Mahmoud Gamal Ahmed Bekhit

School of of Electrical and Data Engineering

Faculty of Engineering & Information Technology

University of Technology Sydney

NSW - 2007, Australia

Resource Allocation and Optimal Scheduling of Virtual Network Functions in Software Defined Networks

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

by

Mahmoud Gamal Ahmed Bekhit

to

School of Electrical and Data Engineering
Faculty of Engineering & Information Technology
University of Technology Sydney
NSW - 2007, Australia

February 2020

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Mahmoud Gamal Ahmed Bekhit* declare that this thesis, submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *School of Electrical and Data Engineering*, *Faculty of Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:
SIGNATURE: Signature removed prior to publication.

[Mahmoud Gamal Ahmed Bekhit]

DATE: 12th February, 2020

PLACE: Sydney, Australia

Abstract

One of the main challenges that faces the Network Functions Virtualization (NFV) deployment is to optimize the resource allocation of demanded network services in the NFV environment. In this study, new optimization models have been developed to find the near to optimal mapping and scheduling for the incoming Virtual Network Function (VNF) requests. The optimization models are formulated as a multi-objective problem in general where different objectives and constraints can be defined depending on the considered scenarios. In the first formulation, three objectives have been defined, namely, maximizing the number of accepted incoming service requests, optimizing link utilization and minimizing the overall processing time of service requests. The second development includes an optimization problem that considers the nonuniform arrival of the incoming service requests periodically. This optimization problem has been done by maximizing the number of accepted service requests, minimizing the number of bottleneck links, the overall processing time. In the third development, the optimization problem considers the expiry time for those incoming service requests to be processed in the VMs. Moreover, the model considers the uniform and non-uniform arrival of the incoming service requests. Four different objectives and five constraints have been considered to solve this optimization problem. Particularly, the model aims to maximize the acceptance rate, minimize the number of bottleneck links, the overall processing time and the relative processing time. In the fourth scenario, the optimization model has been developed to achieve three objectives functions, namely, minimizing the transmission delays occurring in every link, minimizing the processing capacity for every VM and minimizing the processing delay at every VM. The optimization model developed in the fifth formulation minimizes the processing time for every accepted service request, and at the same time maximizes the number of accepted service requests. All five scenarios have been treated as both single-objective and multi-objective optimization problems, where two different evolutionary algorithms based on a genetic algorithm have been applied for solving the resulting optimization problems. Via numerical simulations, it is shown that for the first three scenarios, the proposed algorithms solve the problem efficiently and converge to near to the optimal solution. Regarding the latter two scenarios, the numerical evaluations provide an evidence that the algorithms developed in this manuscript are scalable and they outperform the evolutionary algorithms proposed in the literature, namely genetic bandwidth link allocation (GA-BA) and genetic non-bandwidth link allocation (GA-NBA) algorithms.

DEDICATION

To My Loving Parents, My Lovely Wife, My Future Son, My Beautiful Brothers, My Supportive Family in law, My Amazing Friends And To Everyone That Reads This . . .

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my main supervisor and advisor A/Prof. Mehran Abolhasan for his motivation, suggestions, patience and for his limitless support which have guided me to overcome challenges more successfully. His guidance helped me several times in conducting my research, on writing my thesis and on presenting the research studies as clearly as possible. I am grateful for his effort for reviewing my writing and correcting the mistakes that have led me to publish my research in high-quality conferences and journals. It is a great honor and pleasure to work under his supervision and guidance.

Besides my main supervisor, I would like to thank my co-supervisor A/Prof. Justin Lipman, and external supervisor Dr. Wei Ni from Data61, CSIRO for their experienced supervision, valuable comments, constant encouragement and for their helpe to get results of better quality during my PhD period.

I would like to express my special thanks to my co-supervisor Dr. Saber Jafarizadeh for his genuine support, motivation and wonderful help in my research study. His vision leads me to extend my research work to solve a diverse of exciting real problems, he has taught me the way to formulate the mathematical model perfectly, provide a real example to explain the research problem clearly.

My genuine thanks also goes to Professor Tapabrata Ray from University of New South Wales (UNSW) and Dr Ahmed Salah from Hunan University for offering me all information that I need for my research method.

I extend my gratitude to to all my friends and staff who have helped me at University of Technology Sydney (UTS) in Australia. Specially, Abdelwahed Khamis, Firas Al-Doghman, Ahmed Fathy, Mohammed Jaddoa, Ahmed Saadeh, for their help and support in overcoming numerous obstacles I have been facing through my research.

A special feeling of gratitude to my loving parents, Gamal and Masrya whose words of encouragement and support, whose a look of happiness in their faces is my inspiration and who I work to make them be proud of me. Thank you for being my champions throughout the past 30 years.

I extend my gratitude to my lovely wife who support, encourage and always beside me facing hard times. She has been a constant source of support and encouragement also for her selfless love, care and dedicated efforts which contributed a lot for the completion of my thesis. Also, I would like to thank my future son who become the source of my motivation to work harder.

Special thanks to my lovely brothers, especially Ahmed for the great support and always standing beside me, Mostafa and Mohamed, I am truly thankful for having you in my life.

Thanks to my family-in-law especially, H.Hatta rahman and Hj. Suraida for everything you have done for the love and support. Thanks for always being there for me and help me with the best they know.

Thanks to my motivation Professors and Colleagues in Egypt for the generous care, encouragement and beautiful support during these years of my study both academically and officially.

Lastly, I would like to express my special thanks to all my amazing friends. I can’t thank him enough for everything that they have done for making this work a possible one. your helping hand paved a way to the completion of this thesis.

List of Publications

Related to the PhD Thesis :

1. Gamal, M., Abolhasan, M., Lipman, J., Liu, R.P. and Ni, W.,, 2018, *Multi Objective Resource Optimisation for Network Function Virtualisation Requests.*, Proc. 26th International Conference on Systems Engineering (**ICSEng 2018**), pp. 1-7, Sydney, Australia, 2018. IEEE.

2. Gamal, M., Jafarizadeh, S., Abolhasan, M., Lipman, J. and Ni, W.,, 2019, *Mapping and Scheduling for Non-Uniform Arrival of Virtual Network Function (VNF) Requests*, Proc. 90th Vehicular Technology Conference (**VTC2019-Fall**), Hawaii, USA, 2019. IEEE.

3. Gamal, M., Abolhasan, M.,Jafarizadeh, S., Lipman, J. and Ni, W.,, 2019, *Mapping and Scheduling of Virtual Network Functions using Multi Objective Optimization Algorithm*, Proc. 19th International Symposium on Communications and Information Technologies (**ISCIT 2019**), Ho Chi Minh, Vietnam, 2019. IEEE.

4. Gamal, M., Jafarizadeh, S., Abolhasan, M., Lipman, J. and Ni, W.,, *Resource Allocation and Optimization Scheduling for Virtual Network Function Requests.* IEEE Transactions on Vehicular Technology (**IEEE TVT**). (***under review***).

5. Gamal, M., Jafarizadeh, S., Abolhasan, M., Lipman, J. and Ni, W.,, *Optimal Mapping and Scheduling for Virtual Network Function in Software Defined Networks.*

The International Journal of Computer and Telecommunications Networking (**Computer Networks**). (***under review***).

Table of Contents

LIST OF FIGURES

List of Tables

LIST OF ALGORITHMS

Table 1: NOMENCLATURE

NFV	Network Functions Virtualization
SDN	Software Defined Networking
VM	Virtual Machine
NFV-RA	NFV Resource Allocation
ETSI	European Telecom Standards Institute
HVS	High Volume Server
NS	Network Service
VNF	Virtual Network Function
SFC	Service function chaining
TSP	Telecom Service Provider
NFV-MANO	NFV Management and Orchestration
ILP	Integer Linear Programming
NF	Network Function
SDN	Software Defined Networking
VNFs-SCH	VNFs Scheduling
VNF-FGE	VNF Forwarding Graph Embedding
VNF-FG	Virtual Network Function Forwarding Graph
VNFs-CC	VNFs Chain composition
DPI	Deep Packet Inspector
GA	Genetic Algorithm
DPI	Deep packet inspection
VNFR	Virtual Network Functions Request
GA-BA	Genetic Bandwidth Link Allocation
GA-NBA	Genetic Non-Bandwidth Link Allocation
NS	Network Service
NAT	Network Address Translation
VPNs	Virtual private networks
PGWs	Packet Data Network Gateways
IMSs	IP multimedia subsystems
LDMOAD/DE	The lowest delay multi-objective evolutionary algorithm based on decomposition algorithm.
RU-MOEA/D	Resource Utilization Multi-Objective Evolutionary Algorithm based on Decomposition
SM-MOEA/D	scheduling and mapping multi-objective evolutionary algorithm based on decomposition
SM-NSGA-II	scheduling and mapping Non-dominated Sorting Genetic Algorithm II
NFVI	Network Function Virtualization Infrastructure
NFV-MANO	NFV Management and Orchestration
TSP	telecommunications service provider
VNE	virtual network embedding
VNF-FGs	Virtual Network Function Forwarding Graphs
SFC	Service Function Chaining
SFs	service functions
IETF	The Internet Engineering Task Force
MIQCP	mixed integer quadratically constrained program

Chapter 1

Introduction

1.1 Introduction

In this chapter, we present an overview for the content of the thesis. Firstly, section 1.2 provides the research background of SDN and NFV concepts including the architecture of each concept. Secondly, section 1.3 explains briefly the problem statement and provides the research motivation for the NFV Resource Allocation (NFV-RA) that led us to embark upon the development of new algorithms to optimize the resource allocation in the NFV environment. Thirdly, section 1.4 displays the aims of this research along with the main objectives. Fourthly, section 1.5 shows in details the research gap. Fifthly, section 1.6 depicts the significance of the proposed research problem. Finally, section 1.7 presents the organization and structure of the thesis.

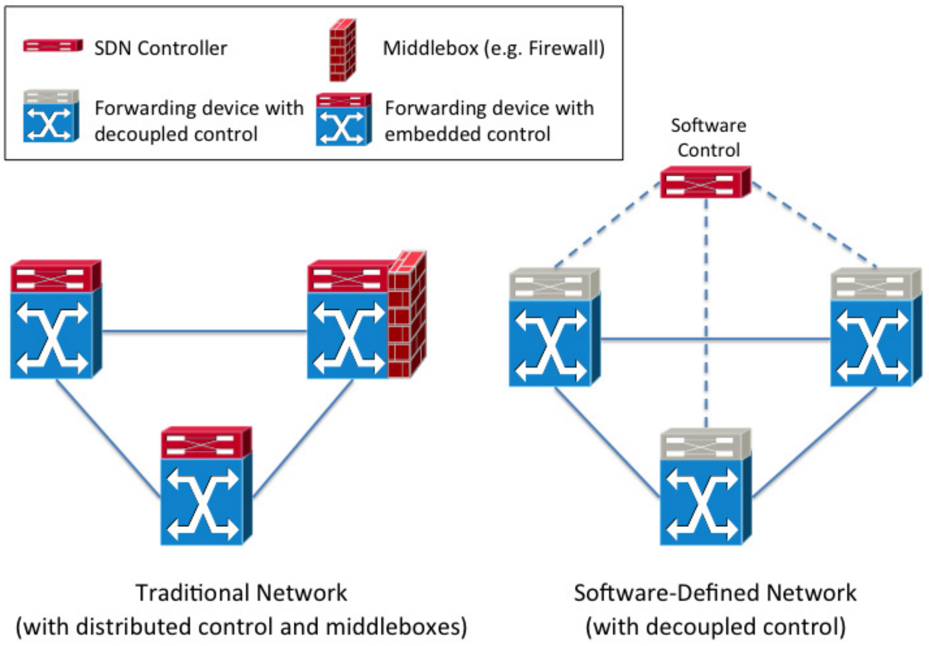

SDN Controller
Middlebox (e.g. Firewall)
Forwarding device with decoupled control
Forwarding device with embedded control
Software Control
Traditional Network (with distributed control and middleboxes)
Software-Defined Network (with decoupled control)

Figure 1.1: Traditional Network and Software-Defined Network architecture as proposed in [1]

1.2 Background

SDN separates the forwarding planes (muscle) from the network control (brains) and provides a central vision of the distributed network for automation of network services and effective orchestration. Control plane is used to establish the forwarding tables of the data plane elements, in other words, determine how the network traffic will be handled. The Data plane is used to forward the network traffic according to the decisions made by the control plane.

The main benefit of SDN is that it provides a centralized controller and network programmability. The important outcome of the SDN concept is the separation between the network policies implementation in the switching hardware, and the forwarding of traffic. This separation breaks the network control problem into manageable pieces, simplifies network management and leads to great flexibility in the network system [2].

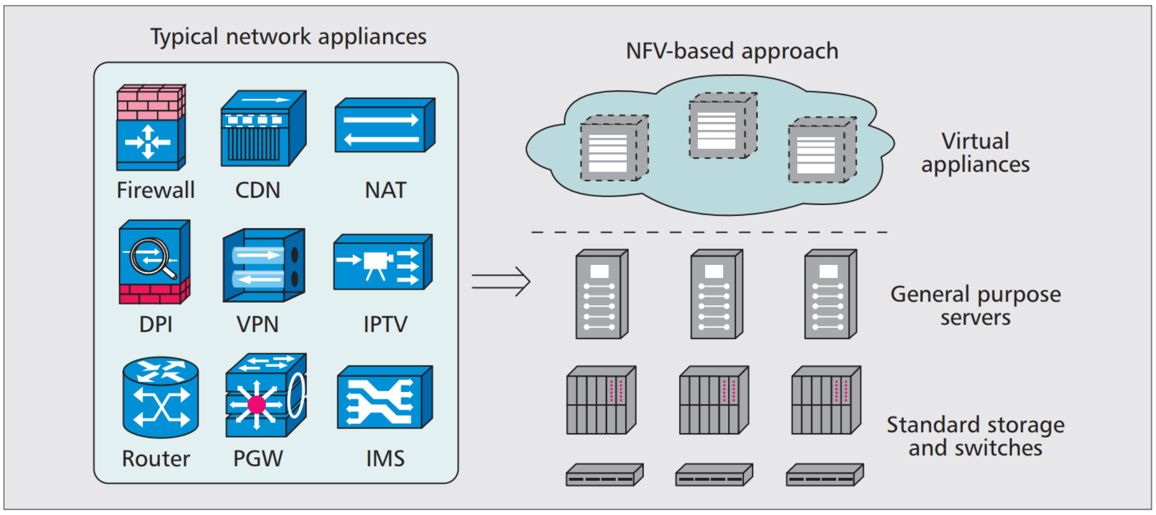


Figure 1.2: SDN architecture and its fundamental abstractions as proposed in [2]

Fig. 1.1 shows the difference between the traditional network and the SDN network. The SDN architecture separates the forwarding hardware plane from the control logic plane and enables merging the policy management, middle-boxes, and new functionalities. The dashed lines in the SDN network define the control-plane links and the solid lines denotes the data-plane links.

NFV is a new framework which has been proposed to reduce the cost of deployment and operation of large networks by applying a flexible allocation for network resources, in addition to integration between new network services and heterogeneous network architecture. Traditionally, the network service (NS) consists of different sets of network functions which are processed and implemented on hardware middleboxes (e.g., firewall, load balancers, network address translators). It needs the data flow/traffic to traverse in a specific order into a fixed set of middleboxes, which are causes more processing to serve/process all functions [5]. NFV decouples the physical network equipment from the functions or services which run on them, e.g. the incoming service can be decomposed into a different set of virtual network functions (VNFs), and each function can be processed one after another in software which implements on top of physical nodes placed in data centres.

Fig. 1.2 illustrates the innovation of the NFV approach by separating the software-based from allocated hardware-based appliances to serve the network services such as routers, deep packet inspection (DPI), firewalls, network address translation (NAT), Virtual private networks (VPNs), packet data network gateways (PDN-GWs or PGWs), IP multimedia subsystems (IMSs), and IPTV. NFV runs the network functions on different hardware (i.e., switches, standard servers, and storage) using software virtualization techniques, as shown in Fig. 1.2.

This principal flexibility of NFV not only decouples the physical network equipment from the network services, but also allows operators to process and implement services quickly and transfer them around as virtual machines to respond to the network needs [5]. Consequently, instead of installing expensive hardware middleboxes, the processing for VNFs into software will provide more available network services for customers to use, moreover, it will reduce the total cost of telecom operators.

1.3 Problem statement

To bring these software-oriented network functions to fruition, many research questions have to be addressed: VNF placement, service chaining, VNF management and orchestration, VNF scheduling for low latency and efficient virtual network resource allocation with NFV infrastructure among others. One of the main challenges that faces the NFV deployment is to optimize the resource allocation of demanded network services in the NFV environment. This problem has been named the NFV Resource Allocation (NFV-RA) problem.

According to [6], every network service request has a different capacity from the others, during the transmission for the flow of the VNFs chains of these requests between the multiple VMs/servers, a transmission delay may occur because of the limited band-

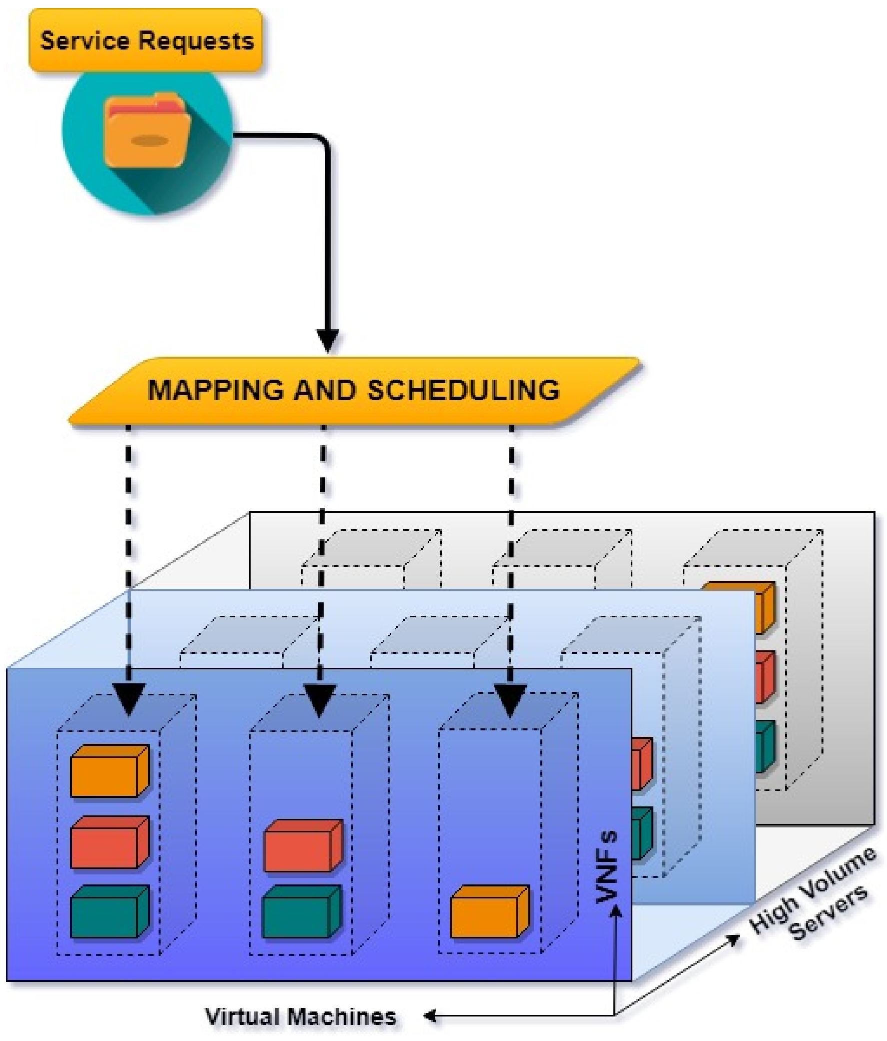


Figure 1.3: Mapping and scheduling for a set of service requests in the NFV environment

width for the virtual links. Since the implementation of the VNFs chains could take more time to be processed on different VMs/servers, a processing delay may occur on these VMs/servers. Both processing and transmission delays will affect the overall processing time which definitely will reduce the overall network performance and increase the processing cost for the service providers. Thus, an efficient resource allocation method is needed to schedule the incoming service requests and to place the VNFs chains in available VMs, so that the service requests can be served effectively.

This thesis presents a new formulation for the NFV-RA problem by introducing different optimization models to cover different network scenarios of the NFV-RA problem. In this study, two different problems are considered, namely, the incoming requests scheduling and network resource allocation for the VNFs service request chains. The first problem seeks to find the best schedule for the arriving service requests to be processed into VMs/servers which will improve the overall network performance by minimizing the total execution time of the network services. The second problem seeks to find the best allocation for VNFs chains to be traversed and processed between the available VMs/servers to improve the overall network performance by minimizing the total execution time of the network services.

It is assumed that every VM supports particular VNFs on top of it which can be shared by different service requests. Besides, the study presents two new multi-objective optimization algorithms to find the near-to-optimal solution for NFV-RA problem.

1.4 Research Gap

In order to achieve a high utilization for the resources and small latency on VMs in the NFV environment, different tasks are required.

1. Every VM supports a specific number of VNF which can be shared by multiple incoming requests. Propose an effective method to find a best scheduling of the arrival requests on available VMs which will lead to reduced congestion, queuing, job rejection rate, latency and achieve high resource utilization.

2. Since there is a resources capacities difference between the computing nodes and VNFs request demand, propose a mathematical model to find the near-optimal solution which can reduce the execution time and cost and simultaneously increase the overall resource utilization of the computing nodes.

3. Propose an efficient model and algorithm to find the optimal way to transfer the functions from one VM to another and also the optimal scheduling for these requests to process in order such that different objectives should be met, for instance, transmission delay, acceptable flow rate, processing delay, link cost, maximize the profit.

4. Propose an efficient algorithm to determine which VM should deploy the VNFs of each service request.

1.5 Research Aims and Objectives

1.5.1 Aims

1. Survey and analysis of the virtual resource allocation, characteristics of different NFV network structures, mapping, and scheduling algorithms.

2. Research on VNFs incoming service request requirements in different NFV network structures including all VNF resource allocation constraints and the prediction of resource workload and service workload.

3. Definition of network attributes in the NFV network and set up different mathematical models.

4. Design resource allocation, mapping, and scheduling algorithms to solve the VNFs requests, which meet the following requirements: utilizes existing virtual and physical resources effectively, fulfills the VNFs transmission and processing requirements and minimizes the resource fragments.

5. Different objectives are considered to choose the best mapping and scheduling including transmission and processing delay, load balance, cost, and resource utilization.

1.5.2 Objectives

1. A literature review of current NFV resource allocation problem including mapping and scheduling for the VNFs service requests.

2. Definition and specification of the VNF network architecture and different network scenarios.

3. Specification and design of a mathematical formulation to optimize the virtual resource allocation by finding the best scheduling and mapping for VNFs of the arrival service requests.

4. Specification and design for different sets of objectives such as maximizing the number of accepted service requests, optimizing link utilization, minimizing the number of bottleneck links, the relative processing time, minimizing the overall processing time of service requests, minimizing the processing capacity for every VM and minimizing the processing delay at every VM.

5. Design of a model that meets the previous objectives and should consider different network constraints including the finite capacity of the links between VMs, VNF's order, VM capacity, expiry time, link traffic as well as the communication and processing delays.

6. Specification, design, and implementation of a multi-objective optimization algorithm, considering the expiry time for the service requests to be processed, supporting both scenarios of the uniform and non-uniform arrival requests and enabling several tenants and multi-services to access the virtual network resources.

7. Develop efficient mapping and scheduling algorithms to optimize the virtual resource allocation. Smart scheduling method considers the best allocation for the incoming VNFs of the service requests to be carried out on specific VMs taking into

consideration a balance between multiple objectives (such as time, performance, cost) in NFV environment.

8. Apply the algorithms into different network scenarios and several instances to find the best algorithm achieving the near-to-optimal solution for the proposed algorithms in every network scenario.

9. Test the model and compare the results with other models proposed previously to verify the performance of the resource allocation problem.

1.6 Research significance

This study contributes to the enhancement of NFV network quality by optimizing the virtual resource allocation. The research encourages the smooth migration of VNFs to be processed through different VMs in the NFV environment. Moreover, the model accelerates the placement and scheduling for the VNFs of the arrival service requests to be run on VMs which increase innovation as well as reduce the processing time, complexity, and cost for VNF providers. The main significance of this study is that the outcome has the following properties:

- Better availability of virtual resources to serve the new arrival requests

- Better distribution for the VNFs of the incoming service requests.

- Reduce the cost of processing the VNFs on VMs

- Reduce VM instantiation and Provisioning Latency, and provide a balance between the different VMs to process the VNFs according to their availability.

1.7 Thesis Contributions

1. Finding an optimal mapping and scheduling for a set of incoming service requests to process in order through different VMs such that three different conflicting objectives are considered. In Particular, maximizing the total number of incoming service requests that can be assigned to VMs, optimizing link utilization and minimizing the processing time while taking into consideration forwarding, assignment and traffic, and link capacity constraints.

2. Formulating a new mathematical model for three mentioned objectives and constraints.

3. Applying scheduling and mapping multi-objective evolutionary algorithm based on decomposition (SM-MOEA/D) algorithm and introduces a penalty function to discard any infeasible solutions of the problem. The main advantage of using this algorithm is to have good scalability and computational efficiency such that a set of conflicting objectives and constraints are met using the updated information in the SDN controller as input parameter for the algorithm.

4. Applying (SM-NSGA-II) algorithm to find the near-to-optimal assignments for VNF service requests to implement in suitable VMs based on the output of the set of pareto solutions.

5. Finding the optimal mapping and scheduling for a set of non-uniform arriving service requests to be processed in suitable VMs. The processing deadline for every incoming request has to be considered.

6. Formulating a mathematical model for the VNF scheduling and mapping problem, that considers the network constraints and the following three objectives: maximiz-

ing the number of accepted service requests, minimizing the number of bottleneck links, and minimizing the overall processing time.

7. Developing algorithms based on genetic algorithms, namely multi-objective evolutionary algorithm based on decomposition (MOEA/D) [7] and Non-dominated Sorting Genetic Algorithm II (NSGA-II) [8]. The algorithms developed can optimize different combinations of the provided objectives.

8. Finding an optimal mapping and scheduling for a set of incoming service requests to process in order through different VMs such that four different conflicting objectives are considered. In Particular, maximizing the acceptance rate, optimizing link utilization and minimizing the overall processing, minimizing the relative processing time while taking into consideration VNF order link, traffic, link capacity and expiry time constraints.

9. Formulating a new mathematical model for the four mentioned objectives and constraints.

10. Applying SM-MOEA/D algorithm and introducing a penalty function to discard any infeasible solutions of the problem. The main advantage of using this algorithm is to have good scalability and computational efficiency such that a set of conflicting objectives and constraints are met using the updated information in the SDN controller as input parameter for the algorithm.

11. Applying scheduling and mapping Non-dominated Sorting Genetic Algorithm II (SM-NSGA-II) to find the near-to-optimal assignments for VNF service requests to implement in suitable VMs based on the output of the set of pareto solutions.

12. Formulating the mapping and scheduling process of a set of arriving service requests at time t among different VMs to be processed in the cloud.

13. Finding the solution that can optimize the following objectives simultaneously: minimizing the transmission delays occurring in every link, minimizing the processing capacity for every VM and minimizing the processing delay at every VM.

14. Solving the resulting problem by proposing an evolutionary algorithm, the lowest delay multi-objective evolutionary algorithm based on decomposition algorithm (LDMOAD/DE).

15. In order to maximize the total income savings for the installed machines in data centers and maximize the accepted traffic flow simultaneously, we formulate the mathematical model considering all these objectives and the network constraints.

16. A heuristic algorithm called Resource Utilization Multi-Objective Evolutionary Algorithm based on Decomposition (RU-MOEA/D) is used to solve the proposed problem and achieve near-optimal placement and cost effectiveness for incoming VNF requests.

17. Extensive simulations for different network sizes are executed to evaluate the performance of the RU-MOEA/D algorithm. The experimental results show that RU-MOEA/D achieves better results than the GA-NBA algorithm in the objective values with execution time less than the GA- NBA algorithm.

1.8 Thesis Structure

- *Chapter 2* presents the background and related work relevant to the NFA-RA research problem. Particularly, the chapter discusses the various categories of the NFV-RA problem such as VNF placement, VNF scheduling, and VNF routing problems. The chapter also elaborates on the main objectives, solution (method) and limitations of each category.

- *Chapter 3* explains in detail the research method which is used in this thesis to solve the proposed VNF-RA problem. There are two new algorithms based on a genetic algorithm which are applied in this study to find the near-to-optimal solution for the proposed VNF-RA problem.

- *Chapter 4* proposes a network scenario that finds the optimal mapping and scheduling simultaneously for a set of incoming service requests to process in order through different VMs such that different objectives are considered.

- *Chapter 5* focuses on the network scenario that considers non uniform sets of service requests arriving to the cloud which need to be processed in order by the installed VMs in the NFV environment.

- *Chapter 6* presents the network scenario that considers the expiry time for the incoming service requests to be processed in the VMs. Moreover, the optimization model in this chapter is applied to solve both cases of uniform and non-uniform arrival of the incoming service requests to be processed in order by the installed VMs in the NFV environment.

- *Chapter 7* presents two different algorithms, a mapping algorithm and a scheduling algorithm, to find the near-to-optimal solution for the proposed network scenarios.

- *Chapter 8* presents different network scenarios and the algorithm design that is tested to maximize the network cost and the admitted traffic. The experiment also tests the GA-NBA algorithm using a multi objective optimization algorithm.

- *Chapter 9* summarises the main contributions of this thesis. Moreover, it provides a window for future possible works in improving and expanding on the presented work of the NFV-RA problem.

Chapter 2

Literature Review

2.1 Introduction

NFV concentrates on optimizing the network services it; changes the design of networks transportation by separating software and hardware through leveraging virtualization technology. The main benefit of NFV is that it increases flexibility, reduces complexity and speeds up service deployment.

2.1.1 The Relationship between SDN and NFV

SDN and NFV have been proposed to combine relevant network functions, assign target performance parameters, and map them onto infrastructure resources[9]. SDN serve NFV by implementing the control plane to run NFs on programmable hardware and to provide connectivity between these NFs. Accordingly, SDN can support NFV to enhance the network performance, facilitate its implementation and simplify the deployment compatibilities [5], [10]. NFV serve SDN by virtualizing the SDN controller to be run

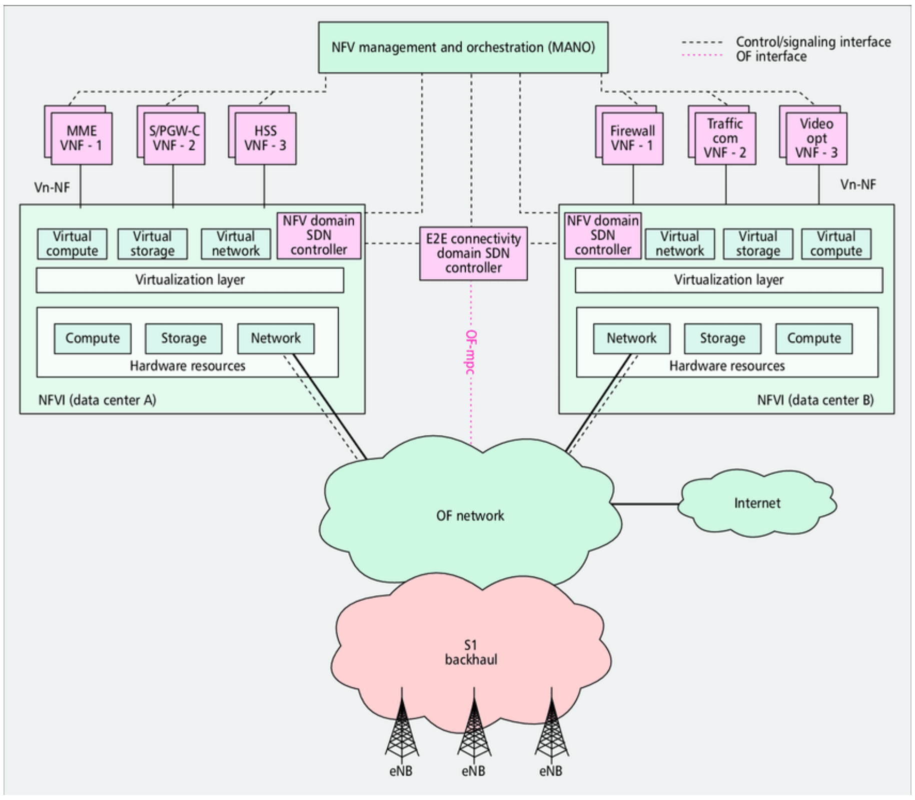


Figure 2.1: SDN and NFV based mobile packet core (MPC) architecture as proposed in [3]

on the cloud and finding the best placement for the controller according to the network needs.

Implementing NFV with SDN together have been proposed several times to utilize the network. For instance, Pate and Han in [5, 11] enable the data packets to be forwarded by an optimized data plane, while the control plane function running on a virtual machine on a rack mount server is used to control distributed forwarding virtual functions.

Authors in [3] proposed a centralized Evolved Packet Core (EPC) control plane (i.e., combined gateway handler) to provide programmability, flexibility, and network resources optimization. As shown in Fig. 2.1, the NFV is responsible for providing the connectivity locally for the storage and computing resources to the data center. Those resources will be linked as endpoints for the transport capabilities in the open flow network.

However, there are some differences between the two concepts SDN and NFV; SDN aims to separate network forwarding functions from the network control functions, while NFV aims to decouple NFs away from dedicated hardware and allows the NFs to be served on servers in the cloud data centers [1]. Moreover, the centralization of the functions can produce scalability issues through two possibilities [3], namely, the controller is run in the clustered medium using a shared database or the control functions (i.e., installation rules of the flow) are decentralized between the switch and the controller. The scalability problem in the SDN network is an important field of research, research; however, this issue can mostly be addressed without losing the benefits of SDN [12].

2.1.1.1 SDN Architecture

The network architecture of the SDN consists of four components [2] as shown in Fig. 2.2

1. The separation between the control plane and forwarding hardware plane (data plane). The control plans is removed from the network devices to be driven by the controller (called, SDN controller) and the network devices turn into simple packet forwarding elements (forwarding hardware plane).

2. Forwarding decisions are flow-based for a set of packets for which field values define a set of instructions. According to the references [13, 14], all packets of the flow get identical policies at the forwarding devices.

3. Control logic is transferred to an external element called the SDN controller.

4. The network is programmed by software applications running on the controller to deal with the underlying data plane devices.

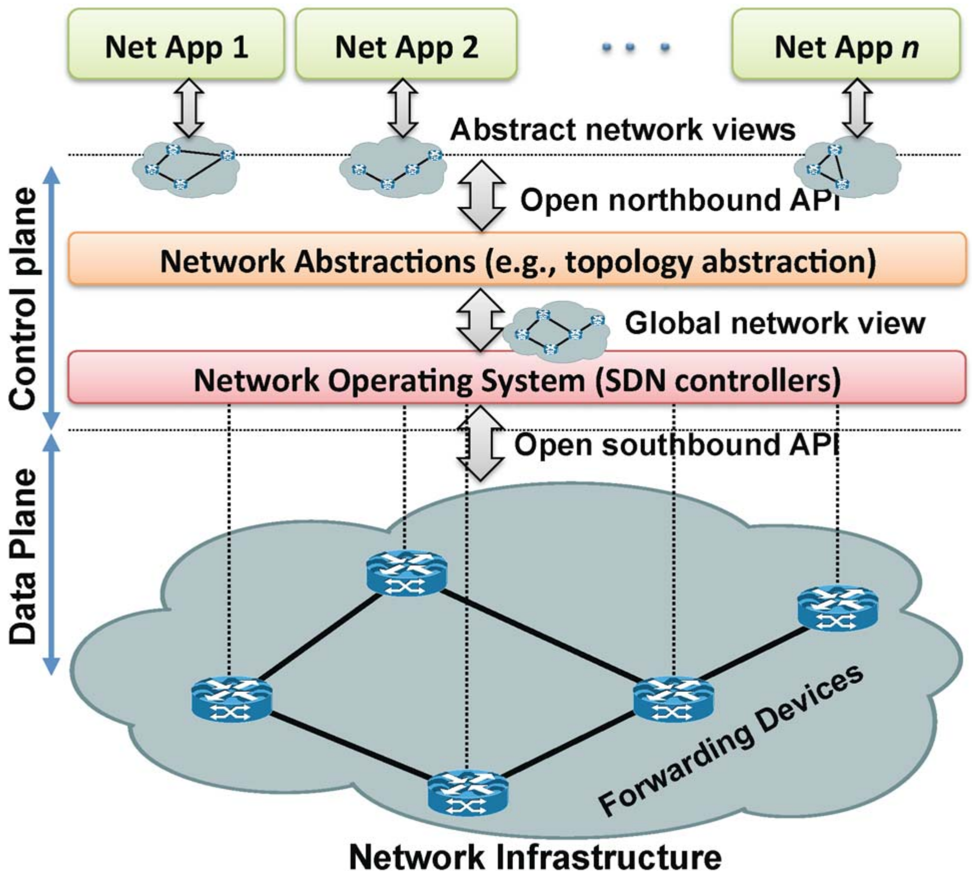


Figure 2.2: SDN architecture and its fundamental abstractions as proposed in [2]

2.1.1.2 SDN Terminology

1. Forwarding Devices (FD): these are software or hardware-based data plane devices that implement a set of primary operations. These devices have sets of specific instruction (e.g., flow rules) used to make a decision for the incoming packets (e.g., forward to the controller, forward to specific ports or drop).

2. Data Plane (DP): there are different ways to connect between the forwarding devices such as wired cables or wireless radio channels; this connection represents the data plane.

3. Southbound Interface (SI): responsible for setting the instruction of the forwarding devices, and for setting the communication protocol between forwarding devices

and control plane entities.

4. Control Plane (CP): this is responsible for programming the forwarding devices using a well-defined SI. Furthermore, it is responsible for setting the control logic in the controllers and applications.

5. Northbound Interface (NI): this is used to abstract the low-level instruction collections used by southbound interfaces to program forwarding devices.

6. Management Plane (MP): is a set of applications (i.e., firewalls, routing, monitoring, load balancers) that influence the functions presented by the NI to execute the network control and operation logic.

2.1.1.3 NFV ARCHITECTURE

The NFV architecture consist of three elements as proposed in [15–17]: Virtual Network Functions (VNFs), Network Function Virtualization Infrastructure (NFVI) and NFV Management and Orchestration (NFV MANO) [16] as shown in Fig. 2.3.

1. Virtual Network Functions (VNFs): the virtualization layer separates the physical resources and the virtualized infrastructure. It ensures the independence of the underlying physical platforms and the NFV life cycle by providing standard interfaces. These functions are provided in the VMs and their hypervisors.

   The orchestrator of the virtual infrastructure is responsible to virtualize and manage the interaction between storage, computing, and network resources with VNFs. It deploys the VM on a suitable hypervisor and manages the network connectivity. The orchestrator is also responsible for analyzing the performance issues and collecting any information about the infrastructure flaw for capacity planning and optimization.

The virtualization layer separates the physical resources and the virtualized infrastructure. It ensures the independence of the underlying physical platforms and the NFV life cycle by providing standard interfaces.

VNFs can be defined as an implementation of NFs (for example, firewalls, DHCP servers) which is processed on virtual resources (e.g., VMs). Every VNF can consist of single or multiple components which can be processed via multiple VMs; however, each VM serves (hosts) only one component of the VNF [15].

A service consists of one or more NFs and can be served by a telecommunications service provider (TSP). In the NFV concept, the NFs are virtualized and processed on different virtual resources such as VMs. The ordering, type, and the number of VNFs are determined by the behavioral specification and service functional. Particularly, the behavior of the service depends on the constituent VNFs.

2. NFV MANO: the orchestrator is in charge of the orchestration and management of the virtualized hardware infrastructure and software resources to perform the network services. The VNF manager is responsible for the scaling, instantiation, update, and termination of the events during the life cycle of a VNF.

   In the data center networking, the hardware resources are almost equivalent which makes their coordination easier but the cost and value of these resources may differ according to the customer's premises and network points of presence. The NFV management system is totally different from the scenario proposed in the data center networking system.

   NFV MANO provides different functionality to support all requirements of the VNFs and all related operations (i.e., the configuration of the VNFs to the VNF infrastructure layer [4, 16]. It also provides management and orchestration for the software and/or physical resources to support the VNFs and NFVI. Moreover, it provides a database which can be used to store the information of the system model, resources, services, function properties and deployments. NFV MANO responsible for all management virtualization tasks for the NFV framework. This framework provides an interface to communicate the coordination of the traditional network management systems and all different NFV MANO components.

3. NFVI is the combination between the software and the hardware resources which form a suitable environment for the VNF to be deployed. The hardware resources contain storage, computing hardware, commercial-off-the-shelf (COTS) and the network resources which is responsible for the connectivity and processing of the VNFs.

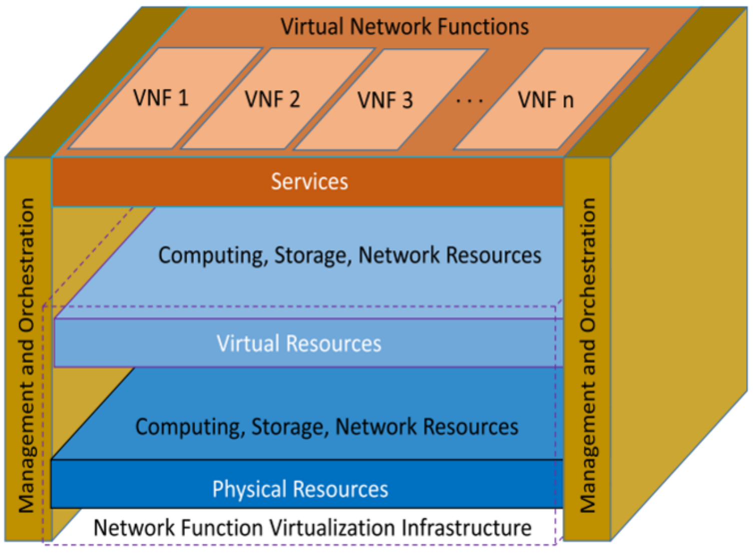
   In a data center, the storage, computing, and connectivity can be performed as one or more VMs, while the virtual networks consist of nodes and connected with virtual links. A virtual node can be defined as a software component has routing or hosting functionality for the VMs (i.e., the operating system which should be processed in a VM). A virtual link is responsible for the interconnection between the virtual nodes, it appears as a direct physical link with dynamically changing properties [18].

2.1.2 Resource Optimization Problem

Most previous attempts to tackle this problem divided the NFV-RA problem into two main sub-problems: network function virtual-resource allocation (NFV-RA) and virtual network embedding (VNE) [4].

- NFA-RA and VNE

The NFV-RA problem aims to find the optimal scheduling and mapping for the VNF to run in the installed VMs. The VNE problem [19, 20] aims to find the best allocation for virtual resources into physical network infrastructure (in both links and nodes) [21]. It


Virtual Network Functions
VNF 1
VNF 2
VNF 3
· · ·
VNF n
Services
Computing, Storage, Network Resources
Virtual Resources
Computing, Storage, Network Resources
Physical Resources
Network Function Virtualization Infrastructure
Management and Orchestration
Management and Orchestration

Network function virtualization architecture.

Figure 2.3: SDN architecture and its fundamental abstractions as proposed in [2]

aims to find an efficient mapping for the virtual network requests on a shared substrate network.

The VNE problem can occur online or offline. In the online problem, all the virtual network requests arrive at the network dynamically and stay for an arbitrary duration, while the offline problem, the requests are scheduled in advance [22–24]. VNE, as proposed in [25], is an NP-hard problem and different meta-heuristic or heuristic algorithms are proposed to solve this problem. Different optimizations objectives to solve the embedding problem have been proposed such as link bandwidth, QoS, energy efficiency [26, 27], economical profit, security [28] and embedding cost.

The two problems (NFV-RAs and VNE) can occur in the same domain. However, there are some differences between these problems as follows:

- The NFV-RAs input request consists of a set of VNFs with resource demands and precedence constraints which can be represented by some Virtual Network Function Forwarding Graphs (VNF-FGs). However, in the VNE problem, the network topologies are static and the nodes are organized in a predetermined and fixed order.

- The resource bandwidth demands for the NFV-RAs problem vary according to the traffic loads assigned to the VNFs, or/and the VNF instances order, while the resource demands in the VNE problem are mostly fixed. In addition, VNEs inputs are scheduled previously according to the order of incoming requests as virtual network topologies are static.

Four main key problems should be addressed carefully to achieve the best efficiency for the NFV implementation: Virtual Network Functions Scheduling (VNFs-SCH), VNF Forwarding Graph Embedding (VNF-FGE), Virtual Network Functions placement (VNFs-PLA) and Service Function Chaining (SFC) [29].

There are two main solutions to solve these mentioned problems, these are: exact and heuristic solutions.

- Exact algorithms are algorithms that always propose the optimal solution for the optimization problem. The exact algorithm is used to solve the small instant optimization problem, and it cannot run in worst-case polynomial time if the problem is NP-hard.

- Heuristic algorithms are a technique designed to solve a problem faster than the classic methods when the classic methods are very slow. Moreover, it is proposed to find the approximate solution (global optimum) when the classic methods cannot find the exact solution for the optimization problem. Since NFV-RA deals with online environments, it is important to minimize the execution time for the service requests to avoid the delay which may occur to process the service requests. Accordingly, heuristic solutions have been proposed to solve NFV-RA.

2.2 VNF-FGE

A lot of works has been proposed to solve the VNF forwarding graph embedding problem such as [30–35].

Before we talk about VNF-FGE problem, we should explain the VNF-FG process which is considered the input for the VNF-FGE. The VNF-FG is the VNF chains formulating a graph of the end-to-end network service process, in other words, it a set of VNFs that process the NS in order to meet the attributes services (i.e., performance, security, reliability, and availability) [15]. The final graph of the VNF-FG process will be given as input for the embedding process of the VNF-FGE stage.

VNF-FGE aims to find the best placement for the VNFs in substrate resource considering the set of network service requests. In addition, a set of objectives and the network constraints should be achieved to optimize the network resources. In particular, minimizing the energy consumption, installation cost and maximizing the usage of available network resources.

VNF-FGE can be named as middlebox/network function placement. The main challenge of this stage is that each VNF has a specific type (such as storage, networking, computing) and this type has to be placed into the physical node that matches this type.

Fig. 2.2 proposed in [2] explained in detail the deployment of the VNF-FGE stage. As shown in the figure, the orchestrator is responsible for the management process between the virtualized hardware infrastructure and software resources to satisfy the networking services. Furthermore, the virtualization layer is separated from the physical resources and runs the VNFs on it using standardized interfaces [5]. The NFV physical layer supports network, computing and storage that provide connectivity, processing and storage for VNFs in the virtualization layer. The VNFs of the same functionality type (computing, networking, storage) can be allocated to run on one or more VMs which are located in network nodes and data centres [36].

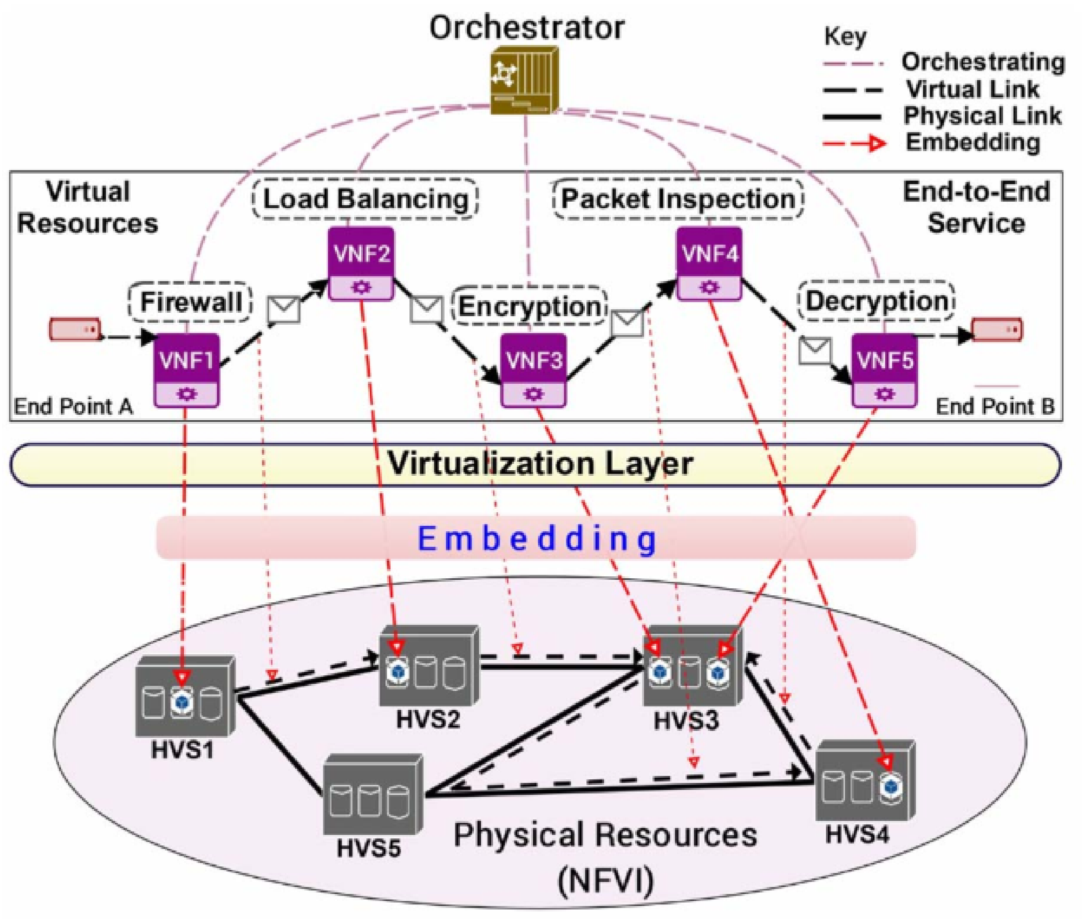


Figure 2.4: VNFs forwarding-graph embedding (VNF-FGE) proposed in [4]

There are three different VNFs deployment architectures as mentioned in [4]. First, the VNFs are deployed on Container Virtualization [4]; this container can run several VNFs and different applications in the cloud. Second, VMs are assigned to the hypervisors which manage the network connectivity [16, 37]. With the third architecture, the VNFs have access to the servers for physical resources without using the hypervisor [38].

Fig. 2.4 shows an example of the embedding process for the VNFs. First, the orchestrator is responsible to run the VNF-FGE algorithm which makes embedding decisions, according to the optimization objective (s). Second, VNF 1 is embedded onto HVS1, VNF 2 is mapped onto HVS 2, VNF3 and VNF 4 are allocated onto HVS 3, and finally VNF 5 is embedded onto HVS 4. Third, the virtual links which connect the VNFs on VMs/servers are mapped using specific algorithms, the path consist of one or more physical links. For example, the virtual link connecting the VNF 3 with VNF 4 is mapped into the path which consists of two physical links HVS 5 - HVS 4 and HVS 3 - HVS 5. The VNF-FGE problem can be more complex if the problem has to be solved dynamically, in that case, the orchestrator has to keep tracking the placement of the running VNFs. In other words, the orchestrator is responsible to rearrange VNFs of different services and move the VNF from HVS to another to minimize the physical resources allocation.

Many of existing works assume that a VNF-FGE problem is closely related to the VNE problem [20, 36, 39–41]. The VNF-FGE is a complex problem and has proved previously to be an NP-hard problem in [25].

2.2.1 VNF-FGE Optimization strategies

Different heuristic and exact algorithms have been applied to solve the VNF-FGE problem; in the following subsections, we will explain in detail the related works which have been using those two algorithms to solve this problem.

2.2.2 Exact

The exact solutions have been used in different situations to solve the VNF-FGE problem. In particular, branch and bound or branch and price have solved the optimization problem in a small number of instances in reasonable time [42]. Also, Integer Linear Programming (ILP) solutions have been used in several practical situations for the formulation of the VNF-FGE problem using software tools available as proprietary (e.g. CPLEX [43]) or as open-source (e.g. GLPK [44]).

Moens et al. [45] formulated a mathematical model to minimize the number of used HVSs in NFV environments, then provided a numerical evaluation for the proposed model. In [39], authors formulated a model to find the best placement for the VNF service chains using a mixed ILP method. The model aims to minimize network latency and traffic engineering in the NFV environment. Martini et al. [6] considers the problem of VNFs deployment to the network nodes and computing the best path for these VNFs to traverse from one node to another. The main objective of this paper is to minimize the overall network latency.

Bauschert et al. [46] formulated a novel mathematical model to solve the problem of finding the best embedding for the virtual mobile core network considering latency constraints.

Ref [36] formulated a new embedding model based on ILP formulation considering the inter-DC network domains and DC load balancing constraints. Authors of [47], provided a design for the VNFs placement problem and evaluated the performance of the proposed algorithm in terms of its ability to support end-to-end requests considering the limited physical resources.

2.2.3 Heuristic

An example of using a heuristic method to solve VNEFGE by minimizing the OPEX is introduced in [30]. The authors formulated a mathematical model to solve VNF-FGE problem considering the minimization of TSP's OPEX and provided rounding-based heuristics algorithm to solve it.

Moreover, authors of [48] proposed a recursive greedy algorithm to solve the VNFFGE problem in WLANs NFV environments; in particular, the algorithm finds the best mapping for the VNFs to network (physical) nodes. Ref [49] proposed a heuristic algorithm to find the best deployment for the mobile core gateways considering different network gateways scenarios and delay constraints.

Authors of [33] formulated a mathematical model to find the optimal placement for the VNFs in packet data centers. Then, they proposed an efficient heuristic algorithm to minimize the overall optic-electronic-optic conversion considering the system constraints. Also, Ref [50] proposed a mathematical model to solve VNF-FG problem which minimizes the execution time for deploying the network services objective. Furthermore, Ghaznavi et al. [51] R1 presented a model to find the best deployment for the elastic VNFs problem; the model aims to minimize the overall cost of providing VNF service.

Bruschi et al. [52] proposed a new energy-aware game theory solution to optimize the VNFs resources in NFV environments. Authors of [34] provided a mathematical formulation to find the best placement for the VNF service requests and the optimal routing for the flows to traverse from one VM to another. The authors proposed a heuristic algorithm to solve the problem efficiently without imposing a big penalty. Moreover, Ref [53] proposed a heuristic algorithm to find the best placement for the VNFs to process on suitable VMs depending on the application-level constraints.

Bagaa et al. [54] provided a model to find the best deployment for the mobile network functions over a federated cloud. Moreover, three heuristics are proposed to solve this problem. Then, Nemeth et al. [55] designed a novel model to solve the carrier-grade networks and evaluated the proposed algorithm to solve this problem. And, authors in [56] formulated a mathematical model to solve the virtual deep packet inspection placement problem taking into account minimizing the network setup cost. In addition, the authors proposed an ILP to solve the multi-commodity flow problem. Lastly, authors of [57] proposed a solution for the VNF-FGE problem in the multi-domain networks by applying the vertex-centric based distributed approach to it.

2.2.4 Exact and Heuristic

Authors in [58] firstly, presented an ILP to minimize the OPEX created by service provider deployment while considering the service delay bounds, then proposed a heuristic algorithm to find the near-to-optimal performance solution while considering the reduction of the execution time. Then [59], the authors formulated a mathematical model to solve the network functions placement problem and presented two different algorithms to solve the problem. In particular, they proposed an ILP model for the small network scenario which aims to minimize the number of VNF instances to be deployed in the cloud and heuristic algorithm to solve the same problem in large size network.

Elias et al. [60] formulated a non-linear optimization model to solve the composition of the network functions problem; in particular, the model aims to detect and minimize the congestion occurring in the physical resources. In addition, Ref [61] proposed a binary integer linear program to find the optimal placement for the radio access network problem in small network instances. Moreover, a greedy approximation algorithm is provided to solve the problem in large network instances. Also, Bellavista et al. [62] presented the challenges of the technical issues facing the optimal VDCs placement problem considering different physical and virtual resources constraints.

Lin et al. [63] introduced two different solutions to minimize the cost of deploying a VNF service request and finding the best route for the VNF traffic flow, namely, a heuristic algorithm based on game theory and mixed-integer linear program. Furthermore, Ref [64] proposed a model to find the optimal deployment for the VNFs on available radio resources using a greedy heuristic algorithm. Lastely, Ref [65] introduced a mathematical model which aims to minimize the total cost of the bandwidth and host set up. A mixed integer programming is proposed to solve the problem in the small network instance and a heuristic algorithm is proposed to solve the problem in large network instances.

2.3 Service Function Chaining (SFC)

SFC is end-to-end delivery of services that requires several service functions to process. SFC is defined as a chain-ordered set of service functions (SFs) that provides network support for the VNFs through the processing of the chains order, service operation, and delivery.

Service chaining is being used with SDN and NFV in several cases, including carrier networks, virtual customer edge and data centers (chaining together virtual or physical network functions). The main advantage of using the combination between SFC, NFV and SDN is to automate virtual network set up to control traffic flows for connected services. For instance, an SDN controller can control the incoming chain of services and distribute them to various traffic flows depending on the type of traffic, source or destination. Moreover, this combination improves the application performance and reduces the usage of network resources by using SDN analytical tools to show the network resources availability.

SDN and NFV provide efficiencies and flexibility to the life-cycle of service function chains. This can happen by composing the chains of the VNFs dynamically and placing them to physical network nodes such as a set of different operator's objectives should be achieved [16]. Moreover, SDN provides monitoring and controlling for the topology service chains using the data plane and transferring the traffic flow across Service Functions (SFs) using the data plane. SF functions provides special treatment for the received packets [66].

This combination is being developed in various industry projects such as The Internet Engineering Task Force (IETF) and The European Telecommunications Standards Institute (ETSI). IETF [29] is proposing a SFC architecture to provide the best traffic route between service functions using network flow classification. ETSI [15] has provided a service chain architecture that finds the best route for traffic between a network service header and virtual network functions (VNFs) using network forwarding graphs.

Traditionally, the network flows pass through different network functions, these functions are determined previously. As a result, the network flow has to transfer between these NFs in a determined order to be processed. This problem is known as network service chaining or network function chaining [17, 67]; complete frameworks of the network function chaining is proposed in [68].

2.3.1 Network Service Chaining

Different ways have been applied to modify the way the network flows are traversed. Firewalls check the data then may drop particular packets; as a result, the final resulting flows are less than the incoming flows. DPI checks the type of packet for the incoming flows, then it can split the flows into different branches according to their type. A video optimizer may change the video encoding scenario to increase the flows data rate.

In some cases, some of the functions in the chaining have a dependency on each other, which means traversing these chains through different NFs has to be in a specific order. For example, if the packets of the incoming flows should pass through WAN optimizer first then the IDS, in that case, the packet inspection should be processed first before the WAN optimizer encrypts the contents.

However, in some other cases, these functions do not have a dependency on each other, therefore, it is possible to execute them all together. There are several factors which have to be considered in that scenario; one of them is how each function in the chain can modify the data rate of the flows as different chaining options have a different effect on the network performance, latency, or on the transmission delays on network links.

Authors of [69, 70] proposed a mathematical model to solve the Location-routing problems for chained NFs. In particular, it aims to find the optimal placement of the VNFs component while minimizing the cost of the links and nodes in the network. These problems consider that each path has a start and endpoint and did not consider the scenario if there are several routes between the NFs to traverse the chains. Ref [20] provided a solution for embedding the chains into a substrate network problem. The authors consider nodes and links of the virtual graphs are independent and they did not consider that the tenants can share the NFs to process the chains.

Authors of [71] presented a mathematical model to express the functionality and attitude of a specific number of NFs, the model does not consider the computational resources and their influence on the network traffic. Authors of [72] provided a model to consider the network-aware orchestration layer for NFs. The model does not consider the resource requirements of the functions while the traffic is processed. The orchestrator plays an important role in finding and reserving adequate resources [73].

2.3.2 Exact

Ref [74] formulated a mathematical model to find the optimal placement for the VNFs taking into account the traffic flows and service chains constraints. Authors of [75] proposed an online algorithm called ACE (Admission control and Chain Embedding) to find the best embedding for the VNF service chain and the traffic flow admission control.

2.3.3 Heuristic

A research by Beck and Botero [76] proposed a heuristic algorithm to find the best deploying for VNF chains into the substrate network. Then, the authors introduced a recursive heuristic algorithm which initially composes the VNFs in the service chains then embeds them in the SN. At each step, the algorithm tries to find the best mapping for the VNFs in the service chains which achieve the best value in a feasible solution.

Reference [77] formulated the problem of deploying the incoming VNF function chains. The authors proposed a placement and routing scenario for VNF instance to run in multiple machines and aimed to minimize the network utilization using a mixed integer programming (MIP) for a small network scale and heuristic called Kariz for the large scale.

2.3.4 Exact and Heuristic

A model that specifies the placement of VNFs chaining requests is presented by [78]. First, it formalizes all incoming request chains into different sets of NFs together using a context-free language within all elements and rules of language. Second, the model aims to minimize the data rate for the chains after sorting the VNFs in ascending order based on the ratio of the incoming and outgoing data rate. Thirdly, it finds the best deployment for the output chains from the previous formalization in the network. The authors proposed a Pareto set analysis and implemented a mixed integer quadratically constrained program (MIQCP) for two different chaining sets to solve this optimization problem and find the actual placement for NFVs.

The authors of [73] proposed two new algorithms based on ILP and heuristic methods to find the best mapping for the service function chains into the network infrastructure taking into consideration that the decomposition of the network services is permitted in their model.

2.4 VNFs Placement (VNFs-PLA)

The problem VNFs-PLA is proposed to exactly find the optimal placement for VNFs in the NFV network infrastructure, such that a set of objectives is achieved (e.g., minimizing the link latency of the chosen paths, minimizing network power consumption and the number of servers used by the network system).

For instance, Ref [21] proposed a new algorithm to find the near to optimal solution for the VNFs placement problem, but the model does not consider the processing order for the VNFs.

Earlier studies focused on the **VNFs-PLA** problem, path selection and the bandwidth allocation for every selected path to transfer these VNFs into the physical network (considering traffic load constraints).

2.4.1 Exact

In [30], the authors proposed a model to determine optimal deployment for VNFs through the physical network. Authors introduced two models: capacity and incapacity model. A limited number of users can be served by a particular function which is located at any node in the capacity model. In contrast, in the incapacity model, the number of clients that can be served by a function is unlimited. Their study provided a mathematical model to enhance the network performance, then the authors proposed a theoretical analysis of the NFV placement problem. Furthermore, their study proposed a combination of two well-known NP-hard optimization problems: the facility location problem and the generalized assignment problem (GAP) to solve the VNFs deployment problem. Authors in [45] had an approach for the virtual network function placement problem (VNFs-PLA) in NFV environments. In particular, they focused on finding the best placement for the VMs and services in the request service chains, but they did not take into consideration the underlying network. The paper used integer linear programming (ILP) to implement the proposed model and to solve the problem. The proposed model applied in either real NFV network or in hybrid network, which means many services can be provided by virtualized network instances only or hybrid with physical hardware.

Clayman et al. [79] introduced a model to find the automatic deployment for the virtual nodes on the VNF environment and for the NSs in these virtual nodes depending on the orchestrator structure. The work elaborates details of managing effective communication between the main four layers (the Application Layer, the Orchestration Layer, the Abstraction Layer, the Infrastructure Layer) which is composed of a combination of SDN & NFV architecture.

The application layer contains the management application (software) which may be simple or complex and responsible for interaction with the Orchestration Layer. The Orchestration Layer consists of three components; the Monitoring Manager, the Placement Engine, and the Service Orchestrator. Firstly, the Monitoring Manager is a critical element responsible for monitoring and controlling functions to achieve a full control loop. Secondly, the Placement Engine which is responsible for finding the best placement of virtual router considering the usage of these virtual routers and the first topology, moreover, the authors paper proposed three algorithms (Least Used Host, Least Busy Host and N at a time in a Host) to determine the best placement of these routers into host. Then the service orchestrator is an application which is running on virtual routers and responsible for the automatic deployment of services or functions into these virtual routers. The Abstraction Layer is in charge of the connection between the Orchestration and Infrastructure layer as it contains the Local controller which is responsible for starting and stopping of virtual routers and the connection between routers. The Infrastructure Layer: this layer consists of physical infrastructure, which is represented by data centres which running VMs on them and virtual infrastructure which is represented by virtual resources (V.Routers, V.Switches) and how these resources connect together through this layer.

2.4.2 Heuristic

In [37], a mathematical formulation and algorithm for the VNF Placement and Routing (VNF-PR) problem were proposed. The main two goals of (VNF-PR) are to find the optimal demand bandwidth assigned to every VNF chains and find the optimal deployment of VNF nodes through NFVI clusters. In particular, the work focused on minimizing the allocated resources and maximum link utilization, taking into consideration the network constraints and forwarding latency regimes. The optimization objectives covered both NFVI- level and network level performance metrics. Finally, the problem is formulated as a multi-objective math-heuristic approach on realistic settings which allows the provider to run experiments for significant instances of VNF-PR within an excellent performance in a reasonable execution time.

Authors of [37, 78] proposed mathematical models to minimize the network operational costs by finding the best placement for VNFs and assignment flow to the VNFs while meeting the traffic policy constraints. The paper proposed heuristic algorithms to solve these problems.

2.4.3 Exact and Heuristic

Reference [80] proposed the joint VNF placement and path selection (JVP) problem, which consists of two significant problems: the sufficient routing path and the best placement of (VNF-SC) into different VMs. The main objective of this problem is to achieve the optimal utilization of the limited resources in servers. The authors proposed VNFs placement algorithm to determine the best placement for VNFs service chains to run in suitable VMs according to path capacity.

Reference [81] provided a solution to solve VNF service chains mapping problem through several VNF instances. The paper aims to minimize the total resource consumption in wide-area network (WAN). The solution consists of three different algorithms: a model called an integer linear program (ILP), column generation-based ILP, and a scheme called two-phase column-generation-based model (2PhMod). These solutions are applied to solve the main objective (minimize the network bandwidth consumed) and several routing constraints are considered in every scenario.

2.5 VNFs-SCH

Given a set of VMs, each VMs supports a limited number of functions on it. The incoming service requests need to be processed in a certain order in these network functions, the scheduling problem seeks to find the minimum execution time slot for each VNF along the service chain to be executed in the corresponding server/VM to improve all network performance.

Several mathematical models and heuristic solutions were proposed to solve the **VNFs-SCH** problem such as the following articles.

Authors in [82] formulated the scheduling problem of the incoming service chain as a flexible job-shop problem. Therefore, there is no exactly polynomial-time solution existing in their model; the model considers only processing delays for the VNFs as the main objective.

Ref [83] provided a novel solution to solve the problem of scheduling the VNFs to process in service chains. The algorithm is called low complexity multi-resource packet scheduling algorithm; it provides a full analysis for the queue characteristics of the VNFs depending on their framework but the algorithm does not consider all the features of the VNFs chaining.

2.5.1 Exact

Authors of [84] formulated a joint service-function deployment and traffic scheduling (SUPER) problem and defined it as an MLIP. The paper aims to maximize the total cost profit for NFV providers by maximizing the total number of serving users flows in the cloud. They then proposed an approximation algorithm based on the Markov approximation technique to solve the proposed algorithm.

2.5.2 Heuristic

In [85], the authors presented an algorithm to solve the NFVI resources optimization problem. NFVI Resource Filter is responsible for determining sufficient NFVI resources requirements. Then, the NFVI Resource Scheduler finds the best schedule for resources and stakeholder policies together. Furthermore, authors in [86] defined VNF chains placement and scheduling problem; the authors provided two algorithms priority-driven weighted algorithm (BFDSU) and Reverse Complete Karmarkar-Karp (RCKK) algorithm to increase the network resource utilization and minimize the network latency respectively.

In [87], the authors have suggested the VNF scheduling problem and proposed a resource optimization algorithm to solve this issue. In particular, the article presented in detail the VNF transmission and processing delays and formulated a mathematical model for the problem. Then, the authors provided three different heuristic methods using a genetic algorithm (GA): GA-without bandwidth allocation (GA-NBA), GA-bandwidth allocation (GA-BA) and GA-bitrate variation (GA-BRV) to improve the complexity of the problem in larger instances. The experimental results of this paper showed that the GA heuristic method provides a good scheduling performance compared with a mixed integer linear program (MILP). Moreover, the results showed that the Ga heuristic methods mentioned previously have shorter schedule times in comparison with the simple greedy best availability scheduling method (GBA) proposed in [82].

The work in [88] formulated the problem of finding the optimal dynamic service function (SF) placement and the routing for the flow in an SFC network. Two different objectives are proposed in this paper: minimizing the energy cost and maximizing acceptable flow rate. The solution first converted the multi-objective optimization problem into a single objective MILP problem then proposed a polynomial time algorithm based on rounding and linear relaxation to find a near to optimal solution for the MILP problem.

2.5.3 Exact and Heuristic

Authors in [82] merge two scheduling and mapping of VNFs chains problems together. The was paper concerned with three different objectives: minimizing flow time (the time of the arrived head service chain to the time of completed serve tail of the last service chain), revenue (the total sum of utilized physical network resources) and cost of physical resources (time gaps of available (non-used) left functions). Authors proposed three greedy algorithms (Greedy Fast Processing (GFP), Greedy Best Availability (GBA) and Greedy Least Loaded (GLL)) and tabu search-based heuristic to solve the previous objectives. The results present that the tabu search-based algorithm has a better performance than all proposed greedy algorithms.

Reference [89] provided mixed-integer linear programming, and a heuristic algorithm called (JoraNFV) to find the optimal solution for NFV-RA problem. This study aims to minimize the total service performance and network cost. Firstly, the authors provided an one-hope algorithm to solve the traffic scheduling problem, then they proposed a multi-path greedy algorithm to solve VNF-CC and VNF-FGE phases.

A new model to solve the VNF scheduling and placement mechanism in the radio access network (RAN) is formulated in [53]. Authors in this paper proposed an ILP to solve the proposed problems in small networks and a heuristic algorithm called Wireless Network Embedding (WiNE) to solve the problem in the instance of a larger network. The results reported the updated proof of concept implementation for the solution of the proposed problem in the NFV environment.

2.6 Conclusion

This chapter studied the literature of NFV-RA problem achieving the best efficiency for the NFV implementation, in particular, VNFs-SCH, VNF-FGE, VNFs-PLA and SFC. These approaches are different from our model, as they aim either to find the best placement only for a set of VNF to serve in suitable VMs or to find the best schedule for the arriving service requests into an NFV environment considering different objectives individually instead of solving mapping and scheduling problems together and optimizing simultaneously all proposed objectives using evolutionary multi-objective optimization. our study seeks to solve this problem considering the trade-off between various objectives including the final acceptance ratio of service request numbers, provide sufficient bandwidth at each VM to process VNF chains and VM cost for multiple incoming service request chains by determining the best VNF mapping and scheduling for the arriving service requests and VNF chains.

Table 2.1: Summary of VNFs-PlA and VNF_SCH

Reference	Category	Objectives (s)	Constraints	Algorithm	Algorithm Type
[79]	VNFs-PLA	———	———	1.Least Used placement 2. Placement N of virtual routers in a Host at time T. 3.Least Busy Host	Exact
[30]	VNFs-PLA	1. minimize the total cost for the system (connection and setup cost). ——— 2.Minimize the total cost of the network system.	1.The NFV demand 2.The setup cost 3.The NFV size 4. The states ——— 1.the job demand 2.The fractional sizes	1. The GAP Rounding Algorithm. ——— 2.The Uncapacitated and Capacitated NFV-Location Algorithm.	Exact

[78]	VNFs-PLA-SFC	1.Maximizing the remaining data rate on network links. 2.Minimizing the used nodes number in the network. 3.Minimize created paths latency.	1) Network Function Placement. 2) Path Creation 3) Metrics Calculation	Mixed Integer Quadratically Constrained Program (MIQCP).	Exact and Heuristic
[80]	NF-PLA	maximize the total size of admitted demands capacity	1.Link and VM capacity 2.Path and Chaining 3.VNF placement	Dynamic programming for solving Joint VNF Placement and Path Selection (JVP)	Exact and Heuristic
[45]	VNFs-PLA	minimize the number of used servers.	———————	The Integer Linear Programming (ILP)	Exact

[37]	VNFs-PLA	1.minimize number of allocated cores (CPU) used to run VNFs functions. 2.Minimize the maximum network link utilization.	1.NFVI cluster capacity 2.VNF flow and forwarding latency 3.VNF node sharing constraints.	Mixed integer linear programming formulation (MILP)	Heuristic
[53]	VNFs-SCH	————-	————	1. An integer linear programming (ILP) for small networks. 2. A heuristic algorithm wireless network embedding (WiNE) for the larger placements in NFV environment.	Exact and Heuristic

[81]	VNFs-PLA-SFC	1. Minimize bandwidth consumed	Several routing constraints	1. an integer linear program (ILP) 2. column generation-based ILP 3. scheme called two-phase column-generation-based model (2PhMod)	Exact and Heuristic
[85]	VNFs-SCH	1. Minimizing the load capacity 2.Minimizing the intra -data centre traffic	————	modified Multi-objective Genetic Algorithm (MOGA)	Heuristic

[87]	VNFs-SCH	1. Minimize Processing delay	1. transmission delay 2. Link and VM bandwidth	1.genetic algorithm-without bandwidth allocation (GA-NBA) 2. genetic algorithm bandwidth allocation (GA-BA) 3. genetic algorithm bitrate variation (GA-BRV)	Heuristic
[89]	VNFs-SCH	1. Minimize the total service performance 2. Minimize network cost	1. Resource consumption of VNFs 2. Traffic processing capacity	1. one-hope algorithm 2.multi-path greedy	Exact and Heuristic

[82]	VNFs-SCH & VNFs-PLA	1.Minimizing flow time 2.Maximizing Revenue 3.Minimiz-ing cost	————	1.Greedy Fast Processing 2.Greedy Best Availability 3.Greedy Least Loaded 4.Tabu Search	Exact and Heuristic
[86]	VNFs-SCH	1.Maximize the network resource utilization of each computing node. 2. Minimize the average response latency	———-	1. priority-driven weighted algorithm BFDSU 2. Reverse Complete Karmarkar-Karp (RCKK)	Heuristic

[84]	VNFs-SCH	1. Maximize the total cost profit for NFV providers by maximizing the total number of serving users flows in the cloud.	Physical resources constraints	1. MLIP 2. approximation algorithm based on the Markov approximation technique	Exact
[88]	VNFs-SCH	1. Maximizing the energy cost 2. Maximizing acceptable flow rate.	1. SC Flow Conservation Constraints 2. Capacity Constraints 3. SC Order Constraints 4.Decision Variable Constraints	1. Flow compensatory rounding-based placement (FCRP)	Heuristic

Chapter 3

Multi Objective Genetic algorithms

3.1 Introduction

Optimization is a process to find the best (the most effective) solution for a set of specific objectives (parameters) without violating some specific constraints. Most previous attempts to tackle this problem divided the decision-making process into six major phases or steps.

- Identify the problem.
- Define the problem.
- Formulate a mathematical model for the problem.
- Obtain a solution for the proposed model.
- Test the proposed model, evaluate the solution, and analysis.
- Implement the solution.

The general optimization problems can be put in categories as follows: firstly, the number of objective functions in which the problem can be a function of a single variable or multiple objectives (called multi-objective optimization), secondly, the optimization objective type (minimization or maximization), thirdly, whether the optimization problem has particular constraints to be considered or not, and finally whether the variable is an integer (discrete), continuous or mixed. For the multi-objective optimization problem, the primary rule is to provide the decision-maker to identify a suitable trade-off between the objectives.

3.2 Genetic algorithms (GAs)

GAs simulates the survival of the fittest with individuals over a successive generations for solving a problem. Each generation consists of a population which has character strings similar to the chromosomes in DNA. Each individual is described as a point in a search space and a potential solution. The individuals in each population are created to pass through the procedure of evolution.

GAs rely on the similarity with the genetic structure and behavior of chromosomes within a population of individuals using the basic idea that individuals in a population compete for resources and mates. Those individuals who win the most in each competition will create more offspring than those individuals with poor achievements. Genes from good individuals are distributed across the population; the two parents may create an offspring that is better than either parent. Thus, every consecutive generation will become more suitable for the environment. GAs update the population of individuals regularly, at each iteration, individuals are calculated using the values of the fitness function. A new generation of the population is formulated by choosing the fitter individuals from the current generation. Some of these individuals are forwarded to the next generation without any modification; while others have to apply the genetic operators such as crossover and mutation to generate a new offspring.

There are six processes to apply the genetic algorithm: generate the initial population, evaluate the fitness value of each individual in the population, rank the individuals based on their fitness, select the best individuals to be submitted to the next generation according to their fitness value, use a genetic operations, such as crossover and mutation to generate a new population and finally, repeat the process from step 2 until the problem 's objectives are satisfied. GAs are a stochastic search process for solving both constraints and non-constraints optimization problems that utilize ideas from natural evolution. A genetic algorithm can present more significant profits than any other typical search optimization technique in searching for the n-dimensional surface, multi-modal state-space, or large state-space.

3.2.1 Multi-objective Optimization

The main feature of evolutionary methods is to use the suitable solution of a population developed in each generation to fit the multi-objective optimization problems. In every generation, a set of the non-dominated solutions are created, one of the main goals for Multi-objective Optimization Problem (MOP) solvers is to find a set of non-dominated solutions with the minimum distance to Pareto front. There are several advantages of evolutionary algorithms as proposed in [90] such as easy implementation, less vulnerability to shape and continuity of Pareto-front, robustness and the capability to be executed in parallel.

There are three main approaches that can be applied to solve an optimization problem by using an evolutionary technique: pareto dominance-based algorithms, indicator-based algorithms, and decomposition-based algorithms. Firstly, pareto dominance-based algorithms [8, 91] have an excellent performance when applied to solve multi-objective optimization problems (MOPs). However, the performance of the pareto dominance is determinate when it is applied to many-objective problems (more than four objectives), as reported in [92, 93]. A clear example for pareto dominance method is Non-dominated Sorting Genetic Algorithm II (NSGA-II).

Secondly, indicator-based approaches [94, 95] employ indicator functions to direct the search; these functions determine the quality of the approximating set. The hypervolume indicator is frequently applied. For instance, the authors in [96] showed that the indicator-based techniques scale comparatively well; however when the optimization problem has up to seven objectives, this occurs only if the indicator can be evaluated.

Finally, decomposition approaches can solve both a single objective optimization problem and multiple singles. The two frequently used weight-based decomposition techniques are the weighted Tchebycheff approach [42] and the weighted sum approach [97] other scalar techniques like Weighted L_p , augmented Tchebycheff, and modified Tchebycheff [97–99] which can be applied as a decomposition technique. In the decomposing approach, the selection pressure is turned into weight vector diversity; the main requirement for this stage is to create sufficient weight vectors for the fitness functions. A number of sub-problems is needed to approximate the Pareto front which also grows with the number of objectives exponentially. As an example [7] and [100] adress a general class of continuous multi-objective optimization test instances. These test instances can be used to study the capability of MOEAs and to provide a new version of MOEA/D derived from differential evolution (DE) MOEA/D-DE.

In the following, we will explain one example from each approach.

3.2.1.1 NSGA II

Deb et al. [8] proposed the NSGA-II algorithm which is the most famous algorithm using EMO procedures. It is a genetic algorithm which aims to find several Pareto-optimal solutions in a multi-objective optimization problem.

At each generation, there is an offspring (Qt), which is generated by the parent (PT) population, then the algorithm combines these two populations together and to form a new population (Rt) with size $2N$ and calculates the fitness evaluation method which depends on Pareto dominance value. The new population will be filled with points of different non-domination fronts. The population with the best filling starts rank (Rank 1) of the non-domination front are temporarily removed from the present population; then the algorithm starts to fill the next rank (Rank 2) of all non-dominated individuals in the remaining population, and so on. Since the total size of the population, Rt is 2N, then not all fronts can be fit in the available N slots of the new population, so the algorithm fills it with the best population ranks until the size of the next generation is N. In the case of some individuals having the same rank, the algorithm will apply the second principle method called crowding distance provided in detail in [52]. Briefly, if individuals have the same rank, individuals in less crowded areas of the objective space are better than individuals in more crowded areas of the same objective space. Then, the algorithm applies the genetic operator's selection, crossover, and mutation to fill an archive of individuals.

3.2.1.2 Hyper Volume

The Hyper-volume indicator has been introduced in [96]; it is a measuring method used in the multi-objective optimization problem to evaluate the performance of search algorithms. The indicator-based techniques scale comparatively well when the number of objectives is more than one.

3.2.1.3 MOEA/D

We use a decomposition approach to solve our proposer problems to solve single and multi-objective optimization problems. It decomposes the problem and solves them as multiple single objective problems. Authors in [7], developed a multi-objective evolutionary algorithm based on decomposition (MOEA/D) technique; MOEA/D uses the Tchebycheff decomposition approach [97] to convert the problem from approximating the Pareto front (PF) into a different number of scalar optimization problems. Each sub-problem uses the information which comes from the neighboring sub-problems to solve the problem by developing a population of solutions. At each generation, the population is composed of the best solutions of each sub-problem. The best solution can be calculated by finding the shortest distances between the neighborhood and their aggregation coefficient vectors.

At each generation t, the MOEA/D maintains the following variables:

- A population $\{X^1,...,X^N\}$ has a size N, where $\{X^i$ is the current solution for the i_{th} sub-problem.
- Calculate the fitness value for each population $\{FV^1,...,FV^N\}$ which is corresponding to a specific sub-problem, for instance, $\{FV^i = F(x^1)\}$ for each $i = 1,2,...,N$.
- The reference point Y^*=$(y_1^*, y_2^*,...,y_M^*)$, where y_i^* is the best value for objective function f_i.
- An external population (EP), which is used to store up non-dominated solutions found through the search.

MOEA/D [7] uses a predefined set of weight vectors to control several sets of trade-off solutions. MOEA/D has been successfully applied to solve unconstrained multi-objective evolutionary algorithm MOEA. However, MOEA/D requires two parameters: penalty parameter and a niching parameter determining the extent of the neighborhood that must be set right. Authors of MOEA/D have not recommended any effective procedure for handling constraints using MOEA/D; thereafter, M. A. Jan and Q. Zhang [101] regulate the replacement, as well as update the scheme of MOEA/D-DE [100] for transacting with constraints in multi-objective optimization problems. The modified scheme CMOEA/D-DE-ATP presented a penalty function to reject infeasible solutions.

The penalty function sets a threshold to control the amount of penalty which provides the infeasible solutions relying on an adaptive threshold value which is provided previously in the updated scheme of MOEA/D-DE [100].

3.2.1.4 Constraint Handling

Several constraints handling techniques have been proposed previously in the literature, Michalewicz and Schoenauer [102] divided the methods for handling constraints using Evolutionary Algorithms into four categories: separate feasible and infeasible solutions, preserving feasibility of solutions [103], penalty functions and hybrid methods. Based on the number of feasible solutions for the current solutions, the search process of a constrained problem can be categorized into three phases [82] taking into consideration the combined parent-offspring population:

1. No feasible solution
2. At least one feasible solution
3. Combined offspring-parent population has more feasible solutions than the size of the next-generation parent population.

The main difference between constraints-handling techniques is how to deal with the infeasible individuals throughout the three search phases.

3.3 Genetic Operators

3.3.1 Initial Population

The initial population is the first task for all Evolutionary Algorithms. The search techniques initiate different solutions (initial population) and try to improve them in the direction of some optimal objective values (solutions). The searching process stops when the predefined criteria are satisfied. If the prior information for the solution is absent, the algorithm usually initiates the population randomly. If the problem is a single objective optimization problem, the best solution will be the solution with the highest fitness value in the population. On the other hand, if the problem is a multi-objective optimization problem, the algorithm will check for the optimal solution for all fitness values in the multi-objective domain.

3.3.2 Selection Operator

The selection operator is the most important point in terms of MOPs, it will mainly select individuals with higher fitness values from the order list of survival. The upper part of the list will be used when good chromosomes are needed, while the lower part represents bad chromosomes [104]. This is one of the reasons for the selection operation to be sometimes identified as the reproduction operator.

Many methods for selection operator in GA have evolved such as Roulette wheel selection, in which the parents are chosen based on their fitness values. Alternatively, Rank Selection, firstly, ranks the population, then ranks each chromosome which receives fitness from this ranking. The best will have fitness equal to N (N denotes the number of chromosomes in the population). The worst fitness will have fitness equal to 1, second the worst fitness equal to 2 and so on. The main idea is to arrange the chromosomes in decreasing order according to their fitness values. Finally, the simplest tournament selection is to choose two random individuals from the population and to stage a tournament to decide which one is selected.

3.3.3 Crossover

After two parents have been chosen through the selection method, crossover occurs. Crossover is the genetic operator that combines the two parents (chromosomes) to formulate a new offspring chromosome; this new chromosome may be better than both parents (obtains the best characteristics from both parents).

Several approaches of the crossover operator have been introduced previously in [105]. Firstly, the single-point crossover is the most frequent crossover technique used between different crossover techniques. A single point crossover is randomly point chosen [106, 107]. Second, the two-point crossover is two points are chosen on the parent organism strings. This type of crossover consists of three types as follows: the binary string which is derived from one parent, the portion between the first to the second point is derived from the second parent and the remaining portion is derived from the first parent. Third, the uniform crossover is a crossover operator that allows the bits of chromosomes to be mixed randomly from the first and second parent. Finally, the Arithmetic Crossover (AC), generates children that have a weighted arithmetic mean which comes from the two parents. The children are feasible and satisfy linear constraints and bounds.

3.3.4 Mutation

The main goal of the mutation operator is to maintain the diversity of the population and increase the opportunity of not losing any potential solution in the global optimal [108], while the crossover operator is a method of quick exploration in the search space. The mutation is a more common name for the asexual genetic operator; it is executed as a bit flip to maintain the diversity in the population and inhibit premature convergence.

3.4 Conclusion

Genetic algorithms (GAs) rely on similarity with the genetic structure and behavior of chromosomes within a population of individuals using the following basic idea: individuals in a population compete for resources and mates. Those individuals who win the most in each competition will create more offspring than those individuals with poor achievements.

Here, to solve the scheduling and mapping problem, we apply two different approaches of GA, namely Pareto dominance and decomposition approaches.

Pareto dominance approach promotes the non-dominated solutions and selects based on the preference of the decision maker, while the decomposition approach decomposes a multi-objective optimization (MOP) into a set of sub-problems and solves them individually. The main reason for choosing these approaches is to provide good scalability and computational efficiency under the consideration of a different number of objectives and constraints.

Hence, we selected the well-known algorithm for each approach, i.e., CMOEA/D-DE-ATP and NSGA II to solve our proposed problem as single and multi-objective and achieve a high performance. The algorithm contains all the standard techniques of the genetic algorithm: generating an initial population, representation, selection, crossover, and mutation. Authors in [109] used a GA and proved that the GA heuristic method has a good scheduling performance in running time compared with Mixed Integer Linear Program (MILP), and simple Greedy Best Availability Scheduling Method (GBA) proposed in [82].

CHAPTER 4

Resource Allocation and Optimization Scheduling for Virtual Network Function (VNF) Requests

4.1 Introduction

SDN and NFV have recently integrated together to share the same feature of promoting creativity, competitiveness, openness and innovation [110], [111]. This chapter focuses on one of the most important problems related to the NFV problem which is the NFV-RA problem as we explained previously in chapter 2. Nowadays, by using the virtualization term, the physical servers can be virtualized as one or more VMs or virtual nodes, these VMs are connected together using virtual links. Given a network structure with a different number of nodes (for instance, 5 nodes as shown in Fig. 4.1), with limited network resources for every VM (computation capacity, CPU type, performance, etc.) and different number of virtual links to connect between these VMs with limited capacity for every link, an unscheduled bottleneck can happen in some of these links and a processing delay can be also happen in every VM.

In this chapter, we assume that the cloud provider supports several cloud tenants and each of these tenants use one or more resources in the cloud. The main goal for the cloud provider is to develop a new strategy, which can serve as many service requests as possible with minimal cost while maximizing the total unused capacity of each VM. Our proposed model addresses the scenarios considered in [82, 109]. The VNFs of each incoming service request should be processed in the order given in the service request. Communication between different VMs handling different VNFs of the same service request should be considered in the model too. The main challenge is to assign the VNFs of the incoming service requests to the VMs in a manner that can maximize the total number of incoming service requests that can be assigned to VMs, optimize link utilization and minimize the processing time while taking into consideration forwarding, assignment and traffic, and link capacity constraints.

As a result of the complexity of the problem, we develop two different algorithms based on genetic algorithm to solve the problem efficiently, namely, SM-MOEA/D and SM-NSGA-II. The effectiveness of both algorithms is shown through the numerical simulation, the simulation finds the near-to-the optimal solution for different applied scenarios within a reasonable computational time.

This chapter is organized as follows: section 4.2, defines a formal mathematical model and an example of the problem. Section 4.3, explains in detail SM-MOEA/D and SM-NSGA-II algorithms which applied to solve the proposed problem. Section 4.4 shows our experimental results on three different scenarios which are presented and discussed in details. Finally, Section 4.5 concludes the proposed problem.

Table 4.1: MATHEMATICAL NOTATIONS

Variable	Description
Network Inputs	
$\mathscr{V}$	A set of VMs nodes installed in the cloud.
$\mathscr{E}$	A set of VLs between VMs.
$\mathscr{S}$	A set of incoming service requests.
F	A set of unique VNFs supported in the cloud.
Service Inputs	
VF_k	The VNFs supported by the k-th VM.
$d_{i,j}$	The bandwidth demand of i-th service request after processing its j-th VNF.
$R^{k,l}_{i,j}(t)$	The traffic flow of the j-th VNF of the i-th service request passing through the virtual link (k,l) at t-th time interval .
f_{ij}	Define the j-th VNF of the service request s_i.
w_i	The weight of each service request s_i which reflects its priority.
$lc_{k,l}$	A transmission capacity of the virtual link (k,l).
$A^k_{i,j}(t)$	$\mathrm{x}^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1))$.
Binary Variables	
$x^k_{i,j}(t)$	whether the j-th VNF of the service request s_i has been assigned to the k-th VM at t-th time interval or not.
$y^{k,l}_{i,j}(t)$	If the current bandwidth demand (d_{ij}) is higher than the available transmission capacity of the link (i.e., $d_{i,j} > lc_{k,l} - \sum_{i'=1}^{\lvert S\rvert}\sum_{j'=1}^{\lvert s_i\rvert} R^{k,l}_{i',j'}(t)$) or not.
$\sigma(R^{k,l}_{i,j}(t))$	If the traffic flow $R^{k,l}_{i,j}(t) > 0$ or not.

4.2 VNF Mapping and Scheduling Problem Formulation

In this section, we formulate the model for the network and the scenario addressed in this chapter.

We consider the network structure presented in Fig. 4.1, and model it as an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ where $\mathscr{V} = \{k|k = 1,...,|\mathscr{V}|\}$ is the set of VM nodes and $\mathscr{E} = \{(k,l)|k,l = 1,...,|\mathscr{V}|, k \neq l\}$ is the set of virtual links (VLs) between VMs. $F = \{f_1^u, f_2^u, ..., f_{|F|}^u\}$ is the set of unique VNFs supported in the cloud. The VNFs supported by the k-th VM is denoted by VF_k for $k = 1,...,|\mathscr{V}|$. For instance, if VM_3 runs $\{f_2^u, f_5^u\}$ as shown in Fig. 4.1, that means only VNFs f_2^u and f_5^u of the incoming request can be executed in VM_3. Let $lc_{k,l}$ for $k \neq l = 1,...,|\mathscr{V}|$ be the transmission capacity of the virtual link (k,l). In this chapter, we assume that there is a virtual link between every two VMs. We define the incoming sequence of service requests as $\mathscr{S} = [s_1,...,s_{|\mathscr{S}|}]$ and each request $s_i \in \mathscr{S}$ for $i = 1,...,|\mathscr{S}|$ has a sequence with different order of VNFs. Let $f_{i,j}$ for $i = 1,...,|\mathscr{S}|$ and $j = 1,...,|s_i|$ be the j-th VNF of the i-th incoming service request (s_i). We assume that every request s_i for $i = 1,2,...,|\mathscr{S}|$ has different number of VNFs. The VNF f_{ij} of the incoming request should be processed in the given order through the selected VMs.

We define $d_{i,j}$ for $i = 1,2,..,|\mathscr{S}|$, $j = 1,2,...,|s_i| - 1$ as the bandwidth demand of the i-th service request s_i after processing its j-th VNF, and $R_{i,j}^{k,l}(t)$ for $i = 1,2,..,|\mathscr{S}|$, $j = 1,2,...,|s_i| - 1$ and $(k,l) \in \mathscr{E}$ as the traffic flow of the i-th service request after processing its j-th VNF, which is passing through the virtual link (k,l) at t-th time interval.

In next three subsections, we address three main objectives considered for mapping and scheduling the service requests.

4.2.1 Maximizing The Acceptance Rate for The Total Number of Incoming Service Requests

In this subsection, we explain in detail the formulation for the first objective which is focused on maximizing the acceptable number of incoming requests, while ensuring particular network constraints are met. This objective function can be defined as below,

$$\max_{x} Z_1(x) = \sum_{i=1}^{|S|} \sum_{k=1}^{|V|} w_i \cdot x_{i,1}^{k}(1) \tag{4.1}$$

where w_i is the weight of each service request s_i, which reflects its priority. This weight is normalized to 1. $x_{ij}^{k}(t)$ is a binary variable denoting whether if the j-th VNF of the service request s_i has been assigned to the k-th VM at t-th time interval. In other words,

$$x_{i,j}^{k}(t) = \begin{cases} 1 & \text{If the } j\text{-th VNF of the service} \\ & \text{request } s_i \text{ has been assigned to} \\ & \text{the } k\text{-th VM at } t\text{-th time interval} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

The cost function $Z_1(x)$ simply counts the number of service request where their first VNF ($j = 1$) has been assigned to any of the VMs at the first time step ($t = 1$). Here it has been assumed that no new service requests will be accepted after time $t = 0$, until all the current service requests are processed.

4.2.2 Optimizing Link Utilization

The available transmission capacity of link (k,l) is the difference between its transmission capacity ($lc_{k,l}$) and its current traffic $\sum_{i'=1}^{|S|}\sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t)$. When the current bandwidth demand (d_{ij}) is higher than the available transmission capacity of the link, i.e., $d_{i,j} > lc_{k,l} - \sum_{i'=1}^{|S|}\sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t)$, the link (k,l) becomes an unscheduled bottleneck. The aim of the objective defined in this subsection is to minimize the number of unscheduled bottlenecks. In doing so, we defined the following binary variable,

$$y_{i,j}^{k,l}(t) = \begin{cases} 1 & \text{if} \quad d_{i,j} > lc_{k,l} - \sum_{i'=1}^{|S|} \sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t), \\ 0 & \text{Otherwise.} \end{cases} \tag{4.3}$$

where k represent the current VM where the j-th VNF is finished and l represent the next VM where the $j+1$-th VNF will be processed. Formally, this objective function is defined as below,

$$\begin{aligned} &\min_x Z_2(x) = \\ &\sum_{t=1}^{T} \sum_{i=1}^{|S|} \sum_{j=1}^{|s_i|-1} \sum_{k=1}^{|\mathcal{V}|} \sum_{l=1}^{|\mathcal{V}|} x_{i,j}^k(t) \cdot y_{i,j}^{k,l}(t) \cdot x_{i,j+1}^l(t+1) \\ &\qquad + x_{i,j+1}^l(t) \cdot \sigma(R_{i,j}^{k,l}(t)) \cdot x_{i,j+1}^l(t+1) \cdot y_{i,j}^{k,l}(t), \end{aligned} \tag{4.4}$$

where

$$\sigma(R_{i,j}^{k,l}(t)) = \begin{cases} 1 & \text{if} \quad R_{i,j}^{k,l}(t) > 0, \\ 0 & \text{Otherwise,} \end{cases} \tag{4.5}$$

and T is the maximum number of time steps considered for processing the incoming service requests. In the cost function defined in (4.4), the first term $(x_{i,j}^k(t) \cdot y_{i,j}^{k,l}(t) \cdot x_{i,j+1}^l(t+1))$ corresponds to the case where the processing of the j-th VNF of s_i has been finished in the k-th VM and next VNF of s_i (i.e., $j+1$-th VNF) has to be processed in the l-th VM. Thus, at time interval t, a transmission will start from k-th VM to l-th VM. If this transmission is not finished in the t-th time interval, the second term in the cost function (4.4) (i.e., $x_{i,j}^l(t) \cdot \sigma(R_{i,j}^{k,l}(t)) \cdot x_{i,j}^l(t+1) \cdot y_{i,j}^{k,l}(t)$) will be triggered if the link (k,l) becomes an unscheduled bottleneck. The function $\sigma(R_{i,j}^{k,l}(t))$ is for verifying if there is any traffic in the link (k,l) at time interval t, due to the eventual processing of j-th VNF of s_i in the l-th VM.

In Table 4.2, an example of mapping and scheduling has been provided. Also, in Fig. 4.5, an example is provided which shows that during the transmission between two VMs, in the first time-step, the first term of the cost function $Z_2(x)$ determines if the link has become an unscheduled bottleneck, while in the following time-steps, the second term of the cost function $Z_2(x)$ determines if the link has become an unscheduled bottleneck.

4.2.3 Minimizing Processing Time

The processing time of a service request is the number of time steps from the time that the service request was accepted until its processing is finished. From each service request's point of view, it is important to minimize its processing time. This can be reflected in the following cost function,

$$\min_x Z_3(x) = \sum_{i=1}^{|S|} \sum_{j=1}^{|s_i|} \sum_{k=1}^{|\mathcal{V}|} \sum_{t=1}^{T} x_{i,j}^k(t). \tag{4.6}$$

Here T is the maximum number of time steps considered for processing the incoming service requests. When processing of service request s_i is finished, all future values of $x_{i,j}^k(t)$ for $j = 1, ..., |s_i|$, $k = 1, ..., |\mathcal{V}|$ is zero. Therefore, for the accepted service requests, their total processing time is calculated in the cost function Z_3.

Comparing the cost functions $Z_1(x)$ and $Z_3(x)$, it is obvious that $Z_1(x)$ prolongs the processing time (to maximize the number of accepted service requests), while $Z_3(x)$ minimizes the processing time.

4.2.4 Constraints

Here, we describe the constraints that have been considered and imposed to the optimization problem.

1. **Forwarding Constraint:**

   The first constraint ensures that the traffic for the $(j+1)$-th VNF of service request s_i start forwarding only if the previous j-th VNF of the same service request s_i complete the processing in the current VM. In other words, at time step $t+1$, the

$(j+1)$-th VNF of service request s_i can be assigned to the l-th VM, if the j-th VNF of service request s_i is finished in the k-th VM. This means that $x^l_{i,j+1}(t+1)$ can be equal to one if $x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1)) = 1$ or $x^l_{i,j+1}(t) = 1$. The term $x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1))$ is equal to one only in the time step where the processing of the j-th VNF of service request s_i is finished in the k-th VM. In other words, $x^l_{i,j+1}(t+1)$ is equal to zero for all values of l, if $x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1)) = 0$ and $x^l_{i,j+1}(t) = 0$, i.e.,

$$\begin{aligned} x^l_{i,j+1}(t+1) = 0 \quad &\text{if} \\ &x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1)) = 0 \text{ or } x^l_{i,j+1}(t) = 0, \end{aligned} \tag{4.7}$$

for $i = 1,...,|\mathscr{S}|$, $j = 1,...,|s_i|-1$, $k,l = 1,...,|\mathscr{V}|$.

2. **Assignment & Traffic Constraint:** The link (k,l) can not have any traffic due to the $(j+1)$-th VNF of service request s_i if the $(j+1)$-th VNF of s_i has not been assigned to the l-th VM. This constraint dictates the relation between $R^{k,l}_{i,j}(t)$ and $x^l_{i,j+1}(t)$, where it can be written as

$$R^{k,l}_{i,j}(t) = 0 \text{ if } x^l_{i,j+1}(t) = 0, \tag{4.8}$$

for $i = 1,...,|\mathscr{S}|$, $j = 1,...,|s_i|-1$ and $(k,l) \in \mathscr{E}$.

3. **Link Capacity Constraint:** The total traffic flow passing through every link (k,l) (i.e., $\sum_{i=1}^{|\mathscr{S}|}\sum_{j=1}^{|s_i|} R^{k,l}_{i,j}(t)$) should not exceed its capacity ($lc_{k,l}$). This can be shown as the following constraint.

$$\sum_{i=1}^{|\mathscr{S}|}\sum_{j=1}^{|s_i|} R^{k,l}_{i,j}(t) \le lc_{k,l}, \quad \forall (k,l) \in \mathscr{E}. \tag{4.9}$$

4.2.5 Example of Mapping and Scheduling for the total incoming number of service requests

Consider the network shown in Fig. 4.1 with 4 VMs, and incoming service requests $s_1,...,s_5$ at time step $t = 0$. The VNFs of service requests are $s_1 = \{f^u_5, f^u_2, f^u_1\}$, $s_2 =$

$\{f_3^u, f_2^u, f_4^u, f_5^u, f_1^u, f_2^u, f_5^u\}$, $s_3 = \{f_2^u, f_4^u, f_1^u, f_5^u\}$, $s_4 = \{f_1^u, f_2^u, f_4^u, f_3^u, f_5^u\}$ and $s_5 = \{f_4^u, f_3^u, f_5^u, f_2^u, f_1^u, f_3^u\}$. As an example, one possible mapping and scheduling for service request s_1 is depicted in Fig. 4.2, where the first and second VNFs of s_1 (i.e., f_5 and f_2) are processed in VM_3, and its last VNF is processed in VM_1. Fig. 4.3 and Fig. 4.4 show the processing schedule and assignment schedule of one of the possible mapping and scheduling for the incoming VNFs of the service requests s_i for $i = 1, ..., 5$. In this example, it is assumed that each VM can process one VNF at a time. Comparing Fig. 4.3 and Fig. 4.4, it is obvious that a VNF of a service request can be assigned to a VM, while it is not being processed. This can be due to two reasons, namely the ongoing transmission or the processing queue. Hence, at any time step, more than one service request can be assigned to a VM. While, at any time step, each service request can be assigned to only one VM.

To further explain the example, in the following we explain how service request s_4 is processed. Initially s_4 is assigned to VM_1, thus $x_{4,1}^1(1) = 1$. In the second time step, the second VNF of s_4 is assigned to VM_3. But either due to the transmission or due to s_1 being processed in VM_3, s_4 has to wait. Same continues in the third time step for s_4. In the fourth time step, processing of second VNF of s_1 is finished in VM_3, but s_4 can not be processed in VM_3 due to internal reasons. In the fifth time step, VM_3 starts to process the second VNF of s_4, while s_1 has to wait in the queue. The sixth time step continues similar to the previous time step. In the seventh and eighth time steps, the third and fourth VNFs of s_4 are assigned and processed in VM_4, respectively. In the ninth, tenth and eleventh time steps, the fifth VNF of s_4 is assigned and processed in VM_3. The processing of service request s_4 is completed after 11 time steps.

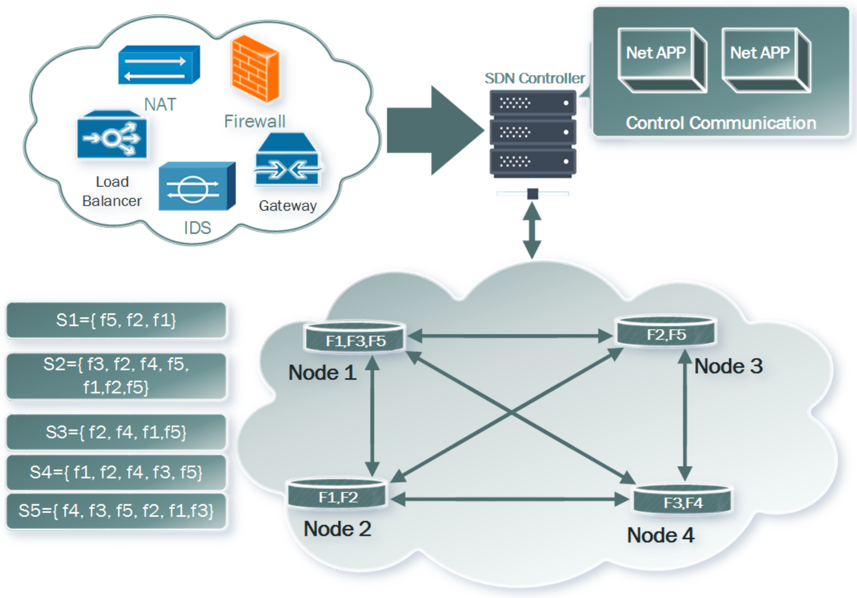


Figure 4.1: Network structure with VM node capabilities

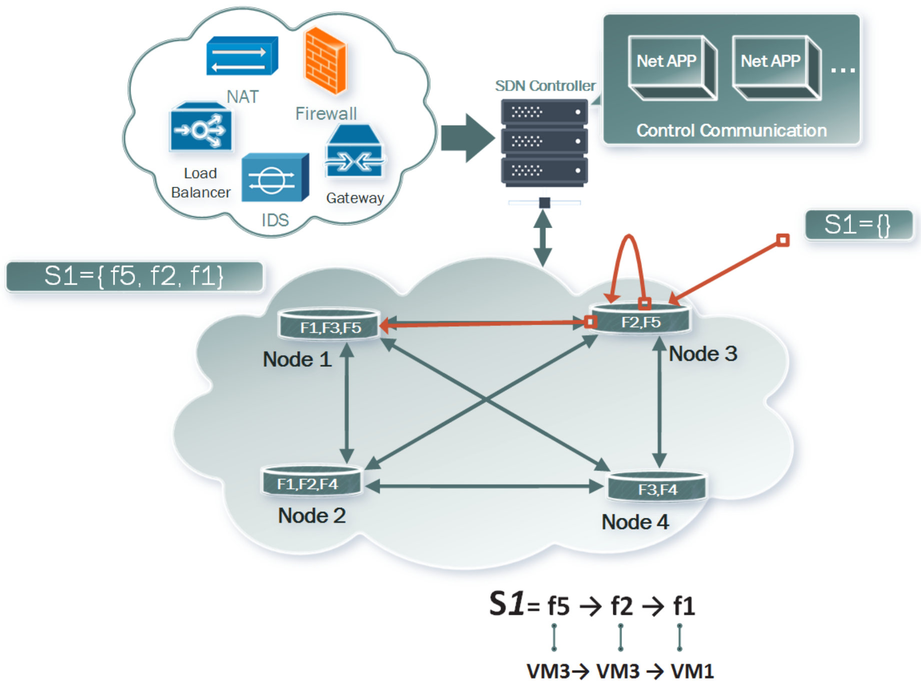


Figure 4.2: one of the possible mappings and schedules for service request s_1

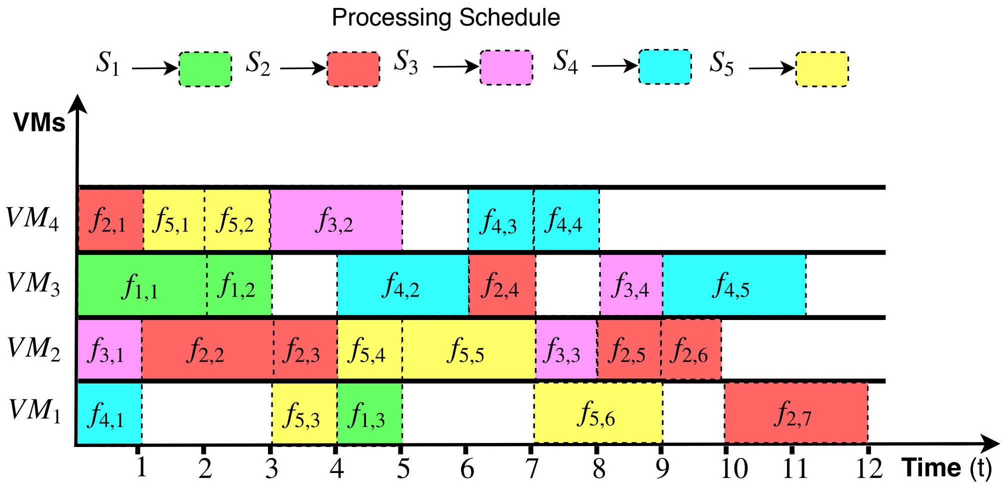

Processing Schedule
S_1 S_2 S_3 S_4 S_5
VMs
VM_4
VM_3
VM_2
VM_1
$f_{2,1}$ $f_{5,1}$ $f_{5,2}$ $f_{3,2}$ $f_{4,3}$ $f_{4,4}$
$f_{1,1}$ $f_{1,2}$ $f_{4,2}$ $f_{2,4}$ $f_{3,4}$ $f_{4,5}$
$f_{3,1}$ $f_{2,2}$ $f_{2,3}$ $f_{5,4}$ $f_{5,5}$ $f_{3,3}$ $f_{2,5}$ $f_{2,6}$
$f_{4,1}$ $f_{5,3}$ $f_{1,3}$ $f_{5,6}$ $f_{2,7}$
1 2 3 4 5 6 7 8 9 10 11 12
Time (t)

Figure 4.3: one of the possible mappings including all incoming VNFs of service requests

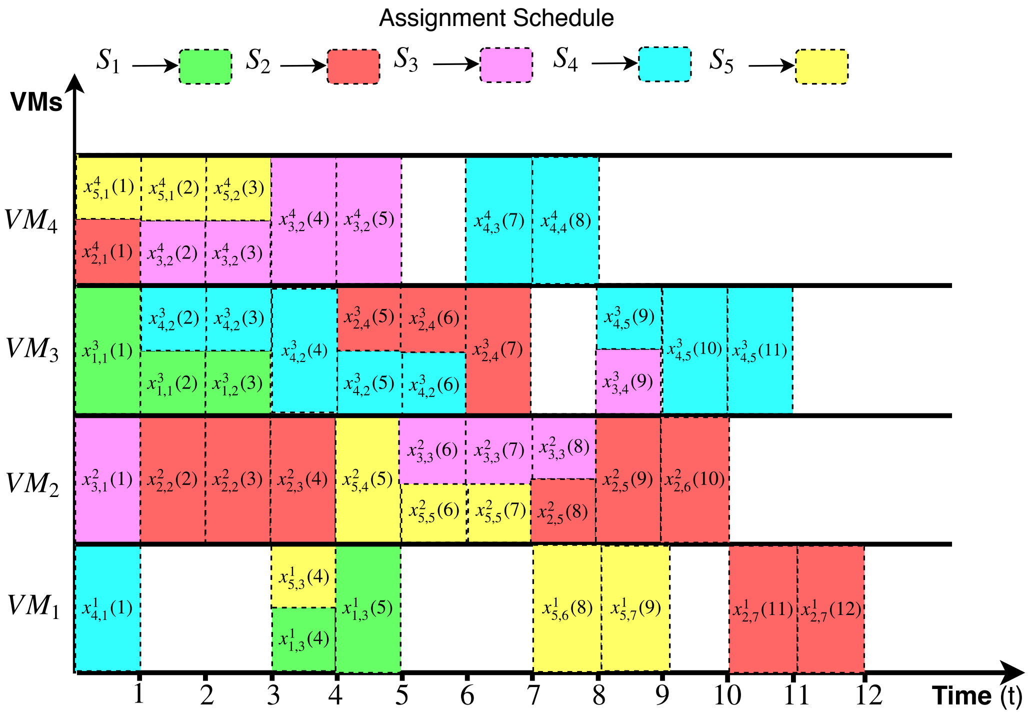

Assignment Schedule
S_1 S_2 S_3 S_4 S_5
VMs
VM_4
VM_3
VM_2
VM_1
$x^4_{5,1}(1)$ $x^4_{5,1}(2)$ $x^4_{5,2}(3)$ $x^4_{3,2}(4)$ $x^4_{3,2}(5)$ $x^4_{4,3}(7)$ $x^4_{4,4}(8)$
$x^4_{2,1}(1)$ $x^4_{3,2}(2)$ $x^4_{3,2}(3)$
$x^3_{1,1}(1)$ $x^3_{4,2}(2)$ $x^3_{4,2}(3)$ $x^3_{4,2}(4)$ $x^3_{2,4}(5)$ $x^3_{2,4}(6)$ $x^3_{2,4}(7)$ $x^3_{4,5}(9)$ $x^3_{4,5}(10)$ $x^3_{4,5}(11)$
$x^3_{1,1}(2)$ $x^3_{1,2}(3)$ $x^3_{4,2}(5)$ $x^3_{4,2}(6)$ $x^3_{3,4}(9)$
$x^2_{3,1}(1)$ $x^2_{2,2}(2)$ $x^2_{2,2}(3)$ $x^2_{2,3}(4)$ $x^2_{5,4}(5)$ $x^2_{3,3}(6)$ $x^2_{3,3}(7)$ $x^2_{3,3}(8)$ $x^2_{2,5}(9)$ $x^2_{2,6}(10)$
$x^2_{5,5}(6)$ $x^2_{5,5}(7)$ $x^2_{2,5}(8)$
$x^1_{4,1}(1)$ $x^1_{5,3}(4)$ $x^1_{1,3}(5)$ $x^1_{5,6}(8)$ $x^1_{5,7}(9)$ $x^1_{2,7}(11)$ $x^1_{2,7}(12)$
$x^1_{1,3}(4)$
1 2 3 4 5 6 7 8 9 10 11 12
Time (t)

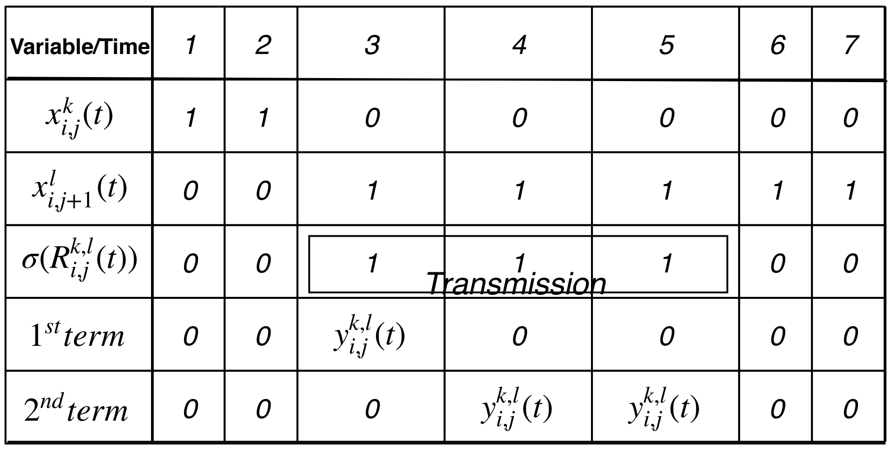
Figure 4.4: All possible values of variable x that reflect the assignment schedule for all incoming service requests

Table 4.2: Assignment Schedule for Example 4.2.5: The values of variable x which are equal to one.

t	s_1	s_2	s_3	s_4	s_5
1	$x^3_{1,1}(1)$	$x^4_{2,1}(1)$	$x^2_{3,1}(1)$	$x^1_{4,1}(1)$	$x^4_{5,1}(1)$
2	$x^3_{1,1}(2)$	$x^2_{2,2}(2)$	$x^4_{3,2}(2)$	$x^3_{4,2}(2)$	$x^4_{5,1}(2)$
3	$x^3_{1,2}(3)$	$x^2_{2,2}(3)$	$x^4_{3,2}(3)$	$x^3_{4,2}(3)$	$x^4_{5,2}(3)$
4	$x^1_{1,3}(4)$	$x^2_{2,3}(4)$	$x^4_{3,2}(4)$	$x^3_{4,2}(4)$	$x^1_{5,3}(4)$
5	$x^1_{1,3}(5)$	$x^3_{2,4}(5)$	$x^4_{3,2}(5)$	$x^3_{4,2}(5)$	$x^2_{5,4}(5)$
6		$x^3_{2,4}(6)$	$x^2_{3,3}(6)$	$x^3_{4,2}(6)$	$x^2_{5,5}(6)$
7		$x^3_{2,4}(7)$	$x^2_{3,3}(7)$	$x^4_{4,3}(7)$	$x^2_{5,5}(7)$
8		$x^2_{2,5}(8)$	$x^2_{3,3}(8)$	$x^4_{4,4}(8)$	$x^1_{5,6}(8)$
9		$x^2_{2,5}(9)$	$x^3_{3,4}(9)$	$x^3_{4,5}(9)$	$x^1_{5,7}(9)$
10		$x^2_{2,6}(10)$		$x^3_{4,5}(10)$	
11		$x^1_{2,7}(11)$		$x^3_{4,5}(11)$	
12		$x^1_{2,7}(12)$			

4.3 Algorithm Design

As mentioned previously, we have three main objectives to achieve: maximizing the acceptance rate for the total number of incoming service requests (4.1), optimizing link utilization (4.4) and minimizing processing time (4.6). This can be written as the



Variable/Time	1	2	3	4	5	6	7
$x^k_{i,j}(t)$	1	1	0	0	0	0	0
$x^l_{i,j+1}(t)$	0	0	1	1	1	1	1
$\sigma(R^{k,l}_{i,j}(t))$	0	0	1	1 Transmission	1	0	0
1^{st} *term*	0	0	$y^{k,l}_{i,j}(t)$	0	0	0	0
2^{nd} *term*	0	0	0	$y^{k,l}_{i,j}(t)$	$y^{k,l}_{i,j}(t)$	0	0

Figure 4.5: Example of the cost function of objective 2

following multi-objective optimization problem,

$$
\begin{aligned}
&\min_{x}\ [-Z_1(x), Z_2(x), Z_3(x)]\\
&st.\ x^l_{i,j+1}(t+1) = 0 \quad \text{if}\\
&\qquad A^k_{i,j}(t) = 0 \text{ or } x^l_{i,j+1}(t) = 0, \text{ for } k,l = 1,...,|\mathcal{V}|\\
&\qquad R^{k,l}_{i,j}(t) = 0 \text{ if } x^l_{i,j+1}(t) = 0,\ \forall (k,l) \in \mathcal{E}\\
&\qquad \textstyle\sum_{i'=1}^{|\mathcal{S}|}\sum_{j'=1}^{|s_i|} R^{k,l}_{i',j'}(t) \le lc_{k,l}, \quad \forall (k,l) \in \mathcal{E}.
\end{aligned}
\tag{4.10}
$$

where $A^k_{i,j}(t) = x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1))$, $i = 1,...,|\mathcal{S}|$, $j = 1,...,|s_i|-1$, and the cost functions $Z_1(x)$, $Z_2(x)$ and $Z_3(x)$ are as defined in (4.1), (4.4) and (4.6), respectively. In this chapter, we find the algorithms NSGA-II [8] and CMOEA/D-DE-ATP [101] are suitable methods for solving our proposed problem (6.12). These algorithms have been explained in chapter 2. In the following, we explain our two proposed algorithms namely, SM-MOEA/D and SM-NSGA-II algorithms, where they utilize NSGA-II [8] and CMOEA/D-DE-ATP [101] algorithms, respectively, which in turn are based on a genetic algorithm. Both proposed algorithms follow the framework explained in Alg. 1 and the flow chart presented in Fig. 4.6. Moreover, both proposed algorithms include all the standard procedures of genetic algorithms that consist of the following steps:

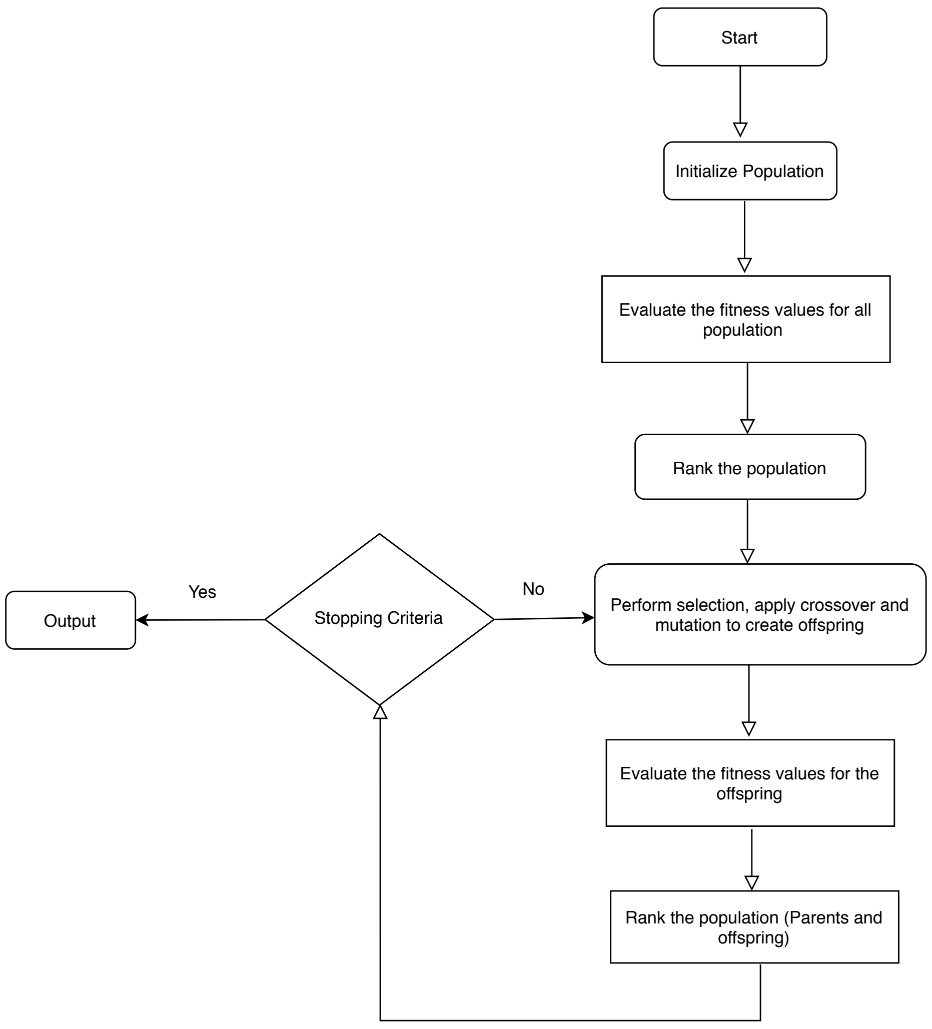

Start
Initialize Population
Evaluate the fitness values for all population
Rank the population
Perform selection, apply crossover and mutation to create offspring
Stopping Criteria
No
Yes
Output
Evaluate the fitness values for the offspring
Rank the population (Parents and offspring)

Figure 4.6: The flow chart of the proposed algorithms procedures

4.3.1 Input

The input of both algorithms are the network parameters, the list of service requests ($\mathscr{S}$), their priorities $W = \{w_i | i = 1,...,|\mathscr{V}|\}$, and the stopping criteria (for terminating the algorithm). The network parameters include the set of VM nodes ($\mathscr{V}$), the set of virtual links ($\mathscr{E}$), the set of unique VNFs supported in the cloud (F), the set of VNFs supported by each VM (denoted by VF) and the transmission capacities of virtual links $\{lc_{k,l} | (k,l) \in \mathscr{E}\}$.

4.3.2 Stopping Criteria

For the SM-MOEA/D algorithm, the stopping criteria are two-fold. First criterion is a predefined maximum limit on the number of iterations, which is set to 300 iterations and 100 population size (the total number of repetition is $(300 * 100)$). Second criterion is defined as the number of iterations where no further enhancement is achieved in the values of the objective functions. Here this value is set to 100 iterations and 100 population size (the total number of repetition is $(100 * 100)$).

For SM-NSGA-II algorithm, the stopping criteria is an upper limit on the number of evaluations which in our simulations is set to 25000.

4.3.3 Output

As outputs, both algorithms return the best feasible solution for mapping and scheduling (i.e., $x_{ij}^{k}(t)$ for $i = 1, ..., |\mathscr{S}|$, $j = 1, ..., |s_i|$ and $k = 1, ..., |\mathscr{V}|$) along with the values of their corresponding objective functions. By feasible solutions, we refer to mapping and scheduling $(x_{ij}^{k}(t))$ that satisfy the constraints of the problem (6.12).

4.3.4 Initialization

The population initialization is the primary task for all evolutionary algorithms. In the initialization phase of Alg. 1, an initial uniformly random population is generated. Each member of the population is a possible mapping and scheduling (defined in terms of variables $x_{i,j}^{k}$), which does not necessarily satisfy the constraints explained in (4.7), (4.8) and (4.9).

In Alg. 1, the variable X refers to a possible mapping and scheduling.

4.3.5 Encoding

A chromosome is a set of genes, which can be encoded in different ways. Encoding is different from one from problem to another, depending on the problem definition. Choosing a suitable chromosome representation will decrease the cost of GA encoding/decoding during the iterations. Symbolic encoding, floating encoding and binary encoding are common encoding methods used in genetic algorithms; the proposed algorithms used a binary encoding. Each candidate solution combination is encoded into a chromosome with a length that is equal to VNF numbers. Every element value of the chromosome is represented as a sub-string (x, y,z), where x is an element of array X corresponding to VNFs assignment of service request s_i, y is an element of array Y of the corresponding installed VMs and z is an element of array Z corresponding to the time step t.

4.3.6 Selection

Choosing a good selection operator of the initial population will affect the quality of the solution significantly. The selection operator selects chromosomes with the highest fitness (objectives) values. The proposed algorithms use a binary tournament as a selection operator. Tournament selection is the operation of running different "tournaments" among a few individuals chosen randomly from the population. In each iteration, the proposed algorithms select two parents's chromosomes from the closest solutions of the sub-problem (individuals which have good fitness (objectives) values). In other words, the selection operator will choose the best two available positions in VMs to process a j-th VNF of a service request s_i at time step t as parents. After that, the selected parents forward chromosomes to the crossover and mutation operators to generate a new individual.

4.3.7 Crossover

During the crossover operation, the parent's output from the selection procedure is recombined to a new offspring chromosome. If the new combination chromosome is better than its parent solution, the proposed algorithm will select it, if not the algorithm will keep the parents's chromosome. There are two different approaches applied in the proposed algorithms for the crossover operator; SM-MOEA/D algorithm uses a DE operator presented in [100], with a pre-described probability rate and SM-NSGA-II algorithm uses simulated binary crossover (SBX) proposed in [8].

4.3.8 Mutation

The mutation operator is applied for different sets of solutions to obtain a new solution by combining some genes together randomly. The mutation is performed as a bit flip to maintain diversity in the population and inhibit premature convergence. In SM-MOEA/D algorithm, the mutation rate is p_m= 1/(parameters dimension) while SM-NSGA-II algorithm has a mutation rate p_m= 1/(no. of variables), which means each bit has a probability of Pm to be flipped. The crossover and mutation operators change VNFs of the service request positions to process them through different VMs, then mates all possibilities formed to generate an excellent new solution depending on the objectives values (4.1, 4.4, 4.6). Finally, the population with the best performance will be selected as a final result. The final result must satisfy the problem constraints (4.7), (4.8) and (4.9). If not, the algorithm removes the solution and select another solution, which can achieve the problem constraints.

4.3.9 Complexity Analysis

The complexity of calculating each one of the constraints (4.7, 4.8, 4.9) is $\mathscr{O}\left(Q|\mathscr{V}|^2\right)$, where $Q = \sum_{i=1}^{|\mathscr{S}|} s_i$. For given mapping and scheduling (X), the complexity of calculating cost functions Z_1, Z_2 and Z_3 are $\mathscr{O}(|\mathscr{S}||\mathscr{V}|)$, $\mathscr{O}\left(Q|\mathscr{V}|^2\right)$ and $\mathscr{O}(Q|\mathscr{V}|T)$, respectively. The detailed complexity of tournament selection, crossover and mutation is very dependent on the setup and parameters of GA, which is out of the scope of this chapter. For further details see [112]. It is obvious that the complexity of calculating the cost functions and the constraints grows linearly with the number of incoming service requests ($|\mathscr{S}|$). The overall complexity of the algorithm depends on the parameters of GA, in particular, the population size and the maximum number of iterations, which can be modified based on the requirements of each specific problem.

4.4 computational results

In this section, we present the simulation results for our proposed scheduling and mapping algorithms, namely SM-MOEA/D and SM-NSGA-II.

4.4.1 Simulation Set-up

1. Simulated Network: we have implemented all the proposed algorithms in MATLAB and Java and run the experiments on a PC with Intel Xeon Gold 6150 2.7GHz, 128 GB of RAM and Linux operating system. In our simulations, the network parameters and the simulation set up have been chosen close to those of [82, 109].

   We consider four different network instances small, medium, large and extra large where the network instant contains $100, 200, 300$ and 400 service requests along with $10, 20, 30$ and 40 VMs, respectively. Moreover, the total capacity for every virtual link is set to 2 Mbps. The bandwidth demand ($d_{i,j}$) of every service request s_i after processing its j-th VNF is distributed uniformly between 30 Kbps and 2030 Kbps. We assume that it takes 1 time steps to process every VNF, independent of the VM node. Furthermore, it is assumed that every VM supports at a maximum 3

Algorithm 1 SM-MOEA/D and SM-NSGA-II

```
1: Input:
     • The stopping criterion
     • Network parameters: $\mathscr{S}, W, \mathscr{V}, \mathscr{E}, F, VF, \{lc_{k,l} | (k,l) \in \mathscr{E}\}$
2: Output:
     • Best feasible mapping and scheduling solution $\{X^1, ..., X^N\}$ ($x_{ij}^k(t)$) satisfying the
       constraints of (6.12)
     • The values of objective functions corresponding to the best feasible mapping
       and scheduling solution
3: Define:
     • X = $\{x_{ij}^k(t) | i = 1, ..., |\mathscr{S}|, j = 1, ..., |s_i|, k \in \mathscr{V}, t = 1, ..., T\}$
4: Initialize:
     • Generate an initial uniformly random population $X^1, \dots, X^N$
5: for $X^i = X^i, ..., X^N$ do
6:     if $X^i$ passes all constraints (4.7), (4.8), (4.9) then
     • Evaluate the cost functions $Z_1(x)$, $Z_2(x)$, $Z_3(x)$ according to (4.1), (4.4), (4.6)
     • Tournament selection
     • Crossover
     • Mutation
     • Choosing Best Feasible Solution
7:     end if
8: end for
9: Selecting the best answer from the population
10: Termination criterion:

     If the stopping criterion (as mentioned in Subsection 4.3.2) is satisfied, then
     stop and return the best mapping and scheduling so far as the output.
```

unique VNFs.

2. VNFs: we select firewall, IDS, load Balancer, Gatway and NAT as functions in this experiment.

3. Chains: sources and targets are distributed uniformly in the simulated network. Poisson distribution are used for the calculation of the arrival rate from the cloud network to the SDN controller. The controller will forward the chains to be processed through the servers using the distribution provided by the proposed algorithms.

4.4.2 Evaluation Scenarios

To evaluate the performance of our algorithms, we apply the proposed algorithms to three different scenarios. All above mentioned network instances (i.e., small, medium, large and extra-large) are simulated for every one of the scenarios. Scenario 1 considers each one of the proposed objective functions individually. Scenario 2 considers all combinations where two out of the three proposed objective functions are selected. Scenario 3 considers all three objective functions simultaneously.

4.4.2.1 Scenario 1

The results for this scenario are reported in Fig. 4.7, Fig. 4.8 and Fig. 4.9, where the optimal value of the three objective functions (i.e., Z_1, Z_2 and Z_3) are reported for all four network instances using both SM-MOEA/D and SM-NSGA-II algorithms.

For example, the optimal value for Z_1 (maximum number of accepted service requests) achieved when we ran the program in a small network environment is 49 achieved by SM-NSGA-II algorithm, while in SM-MOEAD the value is equal to 51 respectively as reported in Fig. 4.7.

Moving to the optimal value for Z_2 (minimize the number of the bottleneck) and Z_3 (minimize the processing time) achieved when we ran the proposed SM-NSGA-II algorithm is , while using SM-MOEAD algorithm the value is equal to 51 as reported in Fig. 4.7 and Fig. 4.8 That is because the number of incoming service requests becomes longer which needs more time to process in the VMs if the number of VMs are fixed (as proposed in this experiment) or need new more VMs to be launched which leads to a higher deployment cost. However, increasing the arrival service requests number lead to increase the probability of bottleneck to be happen in VMs with poor processing capability. From Fig. 4.7, Fig. 4.8 and Fig. 4.9, it is obvious that for individual objective functions, the SM-MOEA/D algorithm is outperforming the SM-NSGA-II algorithm, but as shown in Table 4.3, the computation time of SM-MOEA/D algorithm is higher than that of the SM-NSGA-II. This is partially due to the use of decomposition in the SM-MOEA/D algorithm which will keep searching for more combinations of different VNF placement to serve the arrival service request.

4.4.2.2 Scenario 2

The results for joint optimization of Z_1 and Z_2 are depicted in Fig. 4.10, and those corresponding to joint optimization of Z_2 and Z_3 and joint optimization of Z_1 and Z_3 are depicted in Fig. 4.11 and Fig. 4.12, respectively. From these figures, it is obvious that similar to Scenario 1, the SM-MOEA/D algorithm is outperforming the SM-NSGA-II algorithm, which comes at the expense of greater computational cost.

Increasing the capacity of the links among VMs decreases the optimal value of Z_2. However, after a certain point, increasing the capacity of the links will not have a major effect on the objective functions Z_1 and Z_3. Since, the number of VMs and the combination of unique VNFs supported in each VM will directly impact on the optimal values of Z_1 and Z_3.

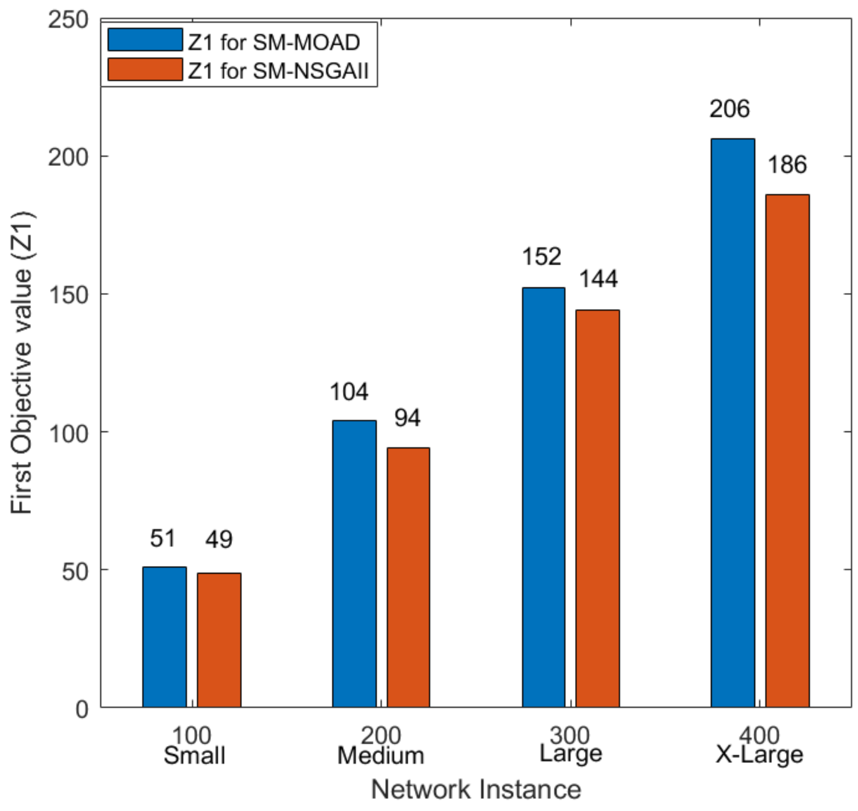


Figure 4.7: The optimal solution achieved by both proposed algorithms for the first objective value via all network instances

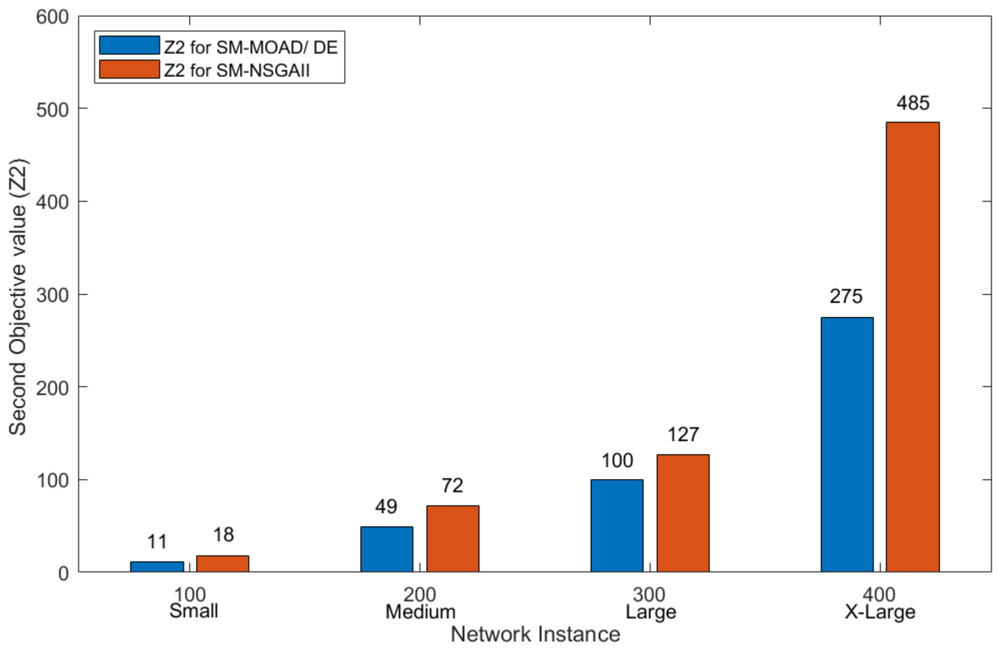


Figure 4.8: The optimal solution achieved by both proposed algorithms for the second objective value via all network instances

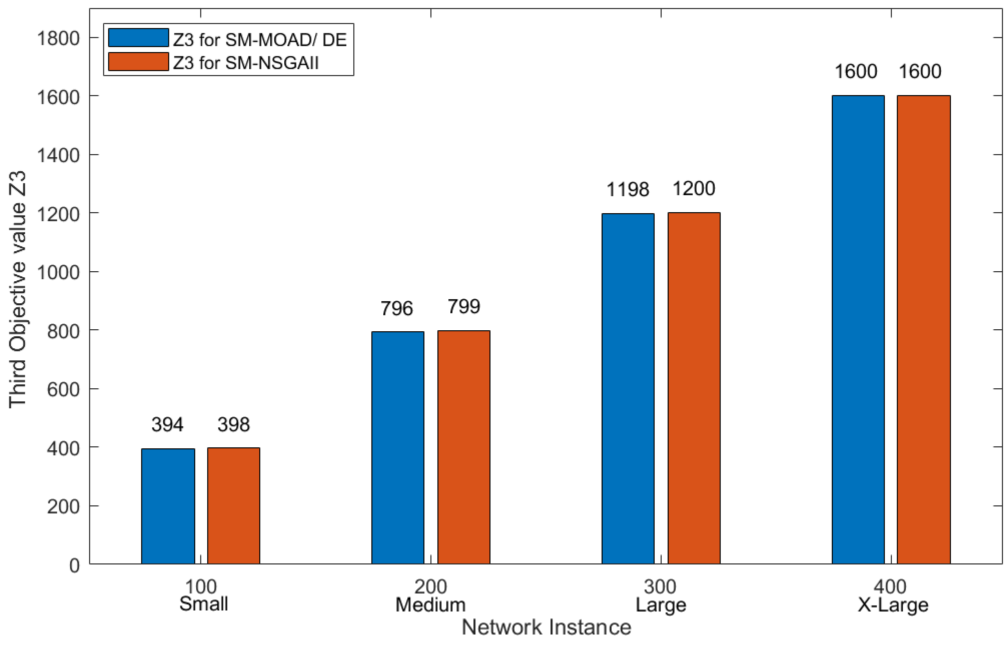


Figure 4.9: The optimal solution achieved by both proposed algorithms for the third objective value via all network instances

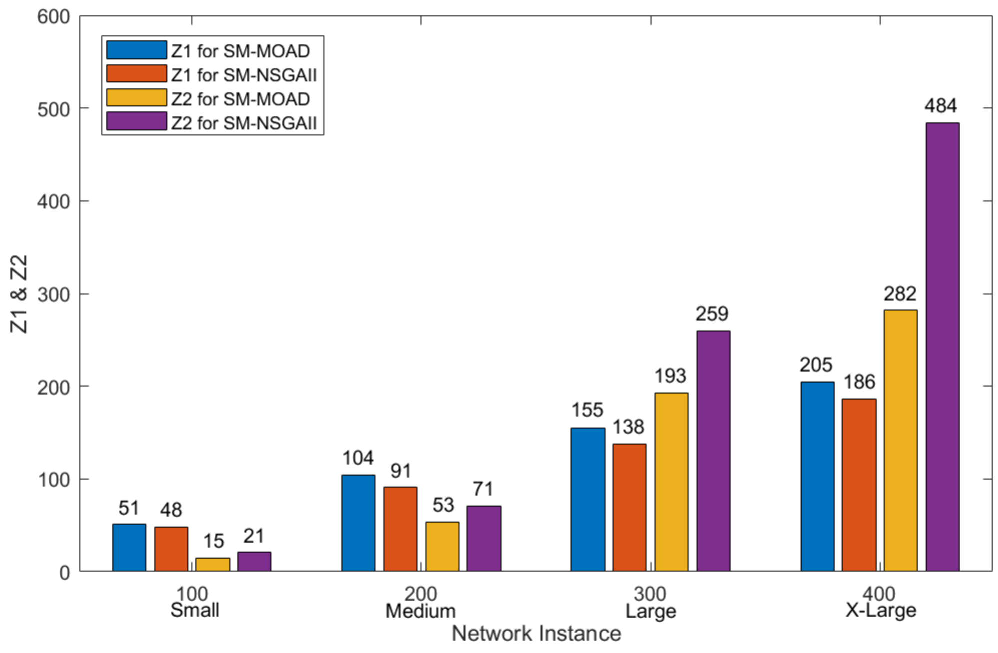


Figure 4.10: The near to optimal solution achieved by both proposed algorithms for the first and the second objective values simultaneously via all network instances

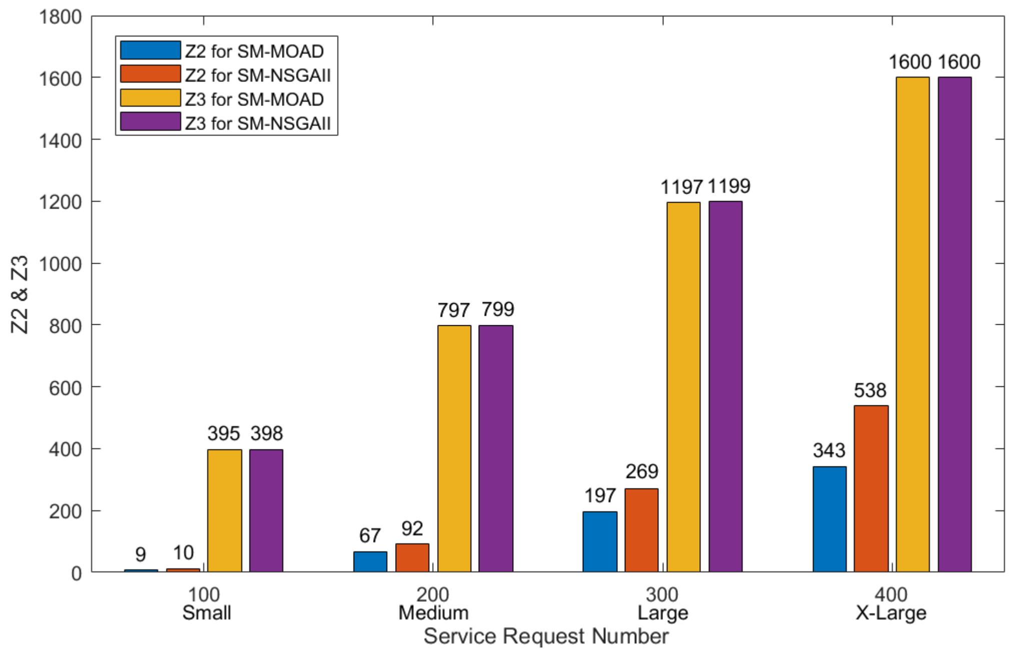


Figure 4.11: The near to optimal solution achieved by both proposed algorithms for the second and the third objective values simultaneously via all network instances

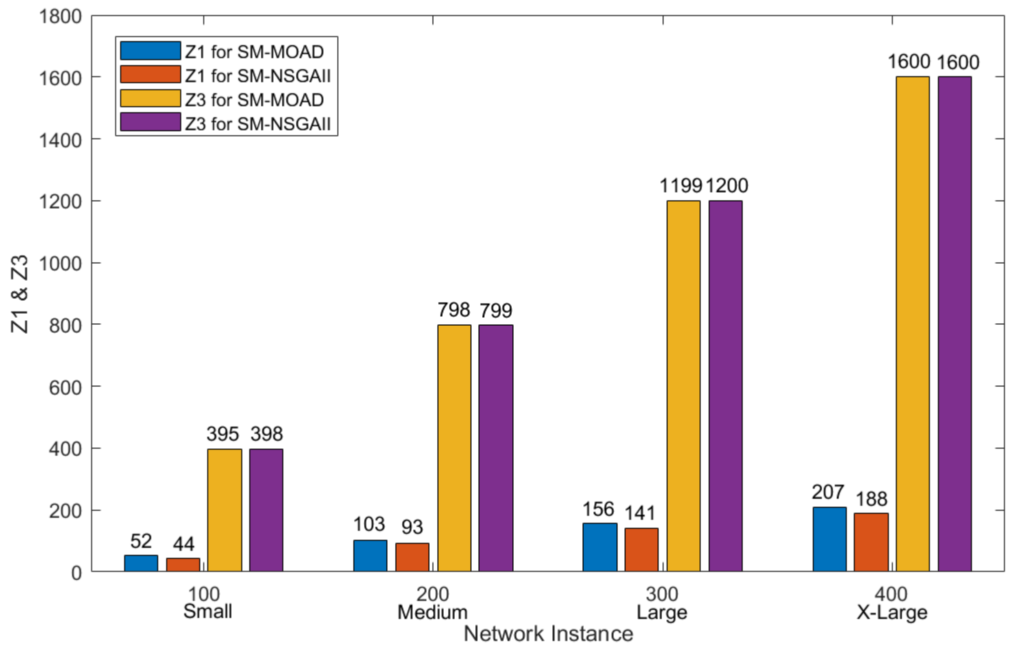


Figure 4.12: The near to optimal solution achieved by both proposed algorithms for the first and the third objective values simultaneously via all network instances

Table 4.3: Computation time (CPU) only for First, Second and Third objective values individually

Alg.	Network Instance	Z_1 (only)	Z_2 (only)	Z_3 (only)
SM-MOEA/D	S	5938	6012	5949
	M	14826	14989	14778
	L	19755	19897	19702
	XL	27823	27857	28021
SM-NSGA-II	S	2034	2011	2023
	M	5963	5994	6012
	L	11098	11163	11055
	XL	15362	15452	15385

4.4.2.3 Scenario 3

The formulation and structure of this scenario is the same as Scenario 1 and 2 but the proposed algorithm is applied to optimize all three objective functions Z_1, Z_2 and Z_3, simultaneously, as formulated in the optimization problem (6.12).

The simulation results presented in Table 7.1 and Fig. 4.13 indicate that the SM-NSGA-II algorithm has better performance in terms of computational time while the SM-MOEA/D algorithm performs better in terms of the optimal results obtained for all three objective functions.

Table 4.4: First, Second and Third objective values for tested algorithms and CPU execution time

Alg.	Network Instance	Z_1	Z_2	Z_3	CPU time (s)
SM-MOEA/D	S	51	15	395	6157
	M	104	61	798	15102
	L	156	134	1198	20123
	XL	208	283	1600	28278
SM-NSGA-II	S	47	19	398	2559
	M	97	77	800	6238
	L	140	198	1199	11478
	XL	186	506	1600	16235

4.5 Conclusion

This chapter presents two algorithms, namely SM-MOEA/D and SM-NSGA-II algorithms for finding near to optimal solutions for mapping and scheduling of the incoming service requests in a cloud. Given a set of VM nodes in the cloud, each VM has particular properties, including the number of supported unique VNFs. The proposed algorithms have the following advantages.

1. They automatically determine the maximum number of the incoming service requests that can be accepted along with the optimal mapping and scheduling for processing the accepted service requests.

2. They can consider both single objective and multiple objective functions for opti-

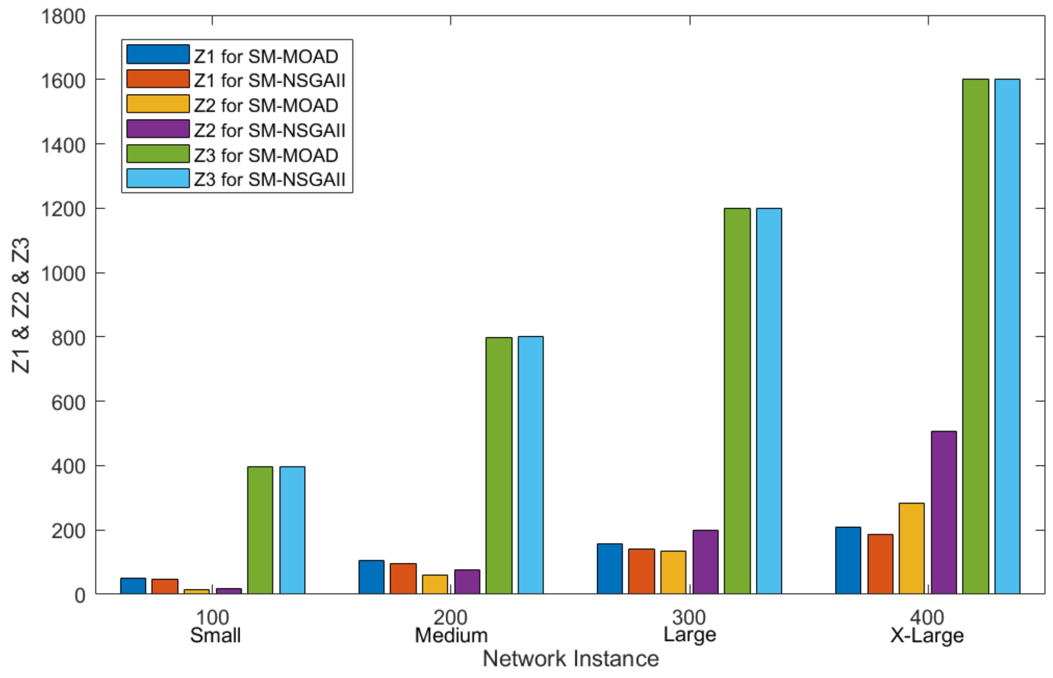


Figure 4.13: The near to optimal solution achieved by both proposed algorithms for the first, second and the third objective values simultaneously via all network instances

mizing the mapping and scheduling of the incoming service requests.

In this chapter, three different objective functions are considered, which are (i) maximizing the number of accepted incoming service requests, (ii) optimizing link utilization and (iii) minimizing the overall processing time of service requests. The model proposed in this chapter takes into account different constraints, including the finite capacity of the links between VMs, the subset of unique VNFs supported in each VM as well as the communication and processing delays. The priority of the incoming service requests is modelled via non-negative weights. The proposed algorithms have been evaluated by simulations for three different scenarios and four different network instances. The simulation results confirm that in general the SM-MOEA/D algorithm outperforms the SM-NSGA-II algorithm in terms of optimal value of the objective functions, which comes at the expense of higher computational cost.

CHAPTER 5

Mapping and Scheduling for Non-Uniform Arrival of VNF Requests

5.1 Introduction

Under the concept of SDN and NFV, SFC is a set of chained instances that are programmed to play an important role for the VNF on virtualization platforms. This allow the virtualization resources, i.e., storage, memory, CPU to be allocated dynamically on each VNF based on VNF traffic request. Moreover, the SFC can be flexibly deployed on those VNFs according to customer request.

Despite the advantages of this process, there are a number of issues that still need to be handled in SFC orchestrating in the NFV environment. One such challenge faced by the NFV concept is to find the optimal mapping and scheduling for the non-uniform sets of the incoming service requests which is the focus of this study in this chapter. These scheduling and mapping have an important impact on the execution time, processing cost, reliability and the performance of the network. The main problem of this model happens when a set of service requests arrive at the cloud at different times asking to be processed on VMs, while some other service requests are still processed on VMs and didn't finish it's processing from the previous time steps as shown in Fig. 5.1. Another challenge that we study in this model is to consider the processing deadline for the incoming requests. In other words, the time which is needed to process the incoming service request has to be less than or equal to the availability time at VMs.

The main difference between the study presented in this study and those of the literature is on the objective functions considered for addressing the NFV-RA problem and for the network formulation scenario. We provide two new evaluation algorithms based on genetic algorithms, namely MOEA/D [7] and NSGA-II [8]. Both algorithms consider the following objectives: maximizing the number of accepted service requests, minimizing the number of bottleneck links, and minimizing the overall processing time. The performance of both algorithms's solution is evaluated with respect to the optimal solution for all three objectives together.

This chapter is organized as follows. Section 5.2 formulates the proposed problem mathematically, considering all objectives, constraints and network structures. Section 5.3 includes the proposed solutions, namely Request Scheduling multi-objective evolutionary algorithm based on decomposition (RSAMOAD) and Request Scheduling Non-dominated Sorting Genetic Algorithm II (RA-NSGA-II). Section 5.4 discusses the simulation results for every individual objective and all three objectives applied together. Section 5.5 concludes this chapter.

5.2 Problem Description and Formulation

Consider the network model represented by undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{v_k | k = 1, ..., |\mathscr{V}|\}$ is the set of VM nodes and $\mathscr{E} = \{(k,l) | k,l = 1, ..., |\mathscr{V}|, k \neq l\}$ is the set of virtual links (VLs) between VMs. $F = \{f_1^u, f_2^u, ..., f_{|F|}^u\}$ is the set of unique VNFs supported in the network.

Each VM supports a subset of VNFs denoted by VF_k for $k = 1,...,|\mathscr{V}|$. For instance, if v_2 runs $\{f_1^u, f_3^u\}$ as shown in Fig. 5.1, that means only VNFs f_1^u and f_3^u of the incoming service requests can be processed in VM_2.

The sequence of network service requests is denoted by $\mathscr{S} = \{s_1,...,s_{|\mathscr{S}|}\}$. Each request $s_i \in \mathscr{S}$ for $i = 1,...,|\mathscr{S}|$ contains a sequence of VNFs denoted by $f_{i,j}$ for $i = 1,...,|\mathscr{S}|$ and $j = 1,...,|s_i|$, where $f_{i,j} \in F$. Service requests can have different number of VNFs and their VNFs should be processed in the given sequence. Let $lc_{k,l}$ for $k \neq l = 1,...,|\mathscr{V}|$ be the transmission capacity of the virtual link (k,l).

In this chapter, we assume that there is a virtual link between every two VMs. We define $d_{i,j}$ for $i = 1,2,..,|\mathscr{S}|$, $j = 1,2,...,|s_i|-1$ as the bandwidth demand of the i-th service request s_i after processing its j-th VNF, and $R_{i,j}^{k,l}(t)$ for $i = 1,2,..,|\mathscr{S}|$, $j = 1,2,...,|s_i|-1$ and $(k,l) \in \mathscr{E}$ as the traffic flow of the i-th service request after processing its j-th VNF, which is passing through the virtual link (k,l) at t-th time interval. In next three subsections, we address three main objectives considered for mapping and scheduling the service requests.

5.2.1 Maximizing the Number of Accepted Service Requests

This objective function maximizes the number of accepted incoming service requests, while ensuring particular network constraints are met. This objective function can be described as below,

$$\max_x Z_1(x) = \sum_{i=1}^{|S|} \sum_{k=1}^{|\mathscr{V}|} w_i \cdot \left(x_{i,1}^k(1) + \sum_{t=2}^{T} U\left(x_{i,1}^k(t) - x_{i,1}^k(t-1) \right) \right) \tag{5.1}$$

where w_i is the weight assigned to each service request s_i reflecting its priority. This weight is normalized to 1. $x_{ij}^k(t)$ is the binary variable indicating the assignment of j-th

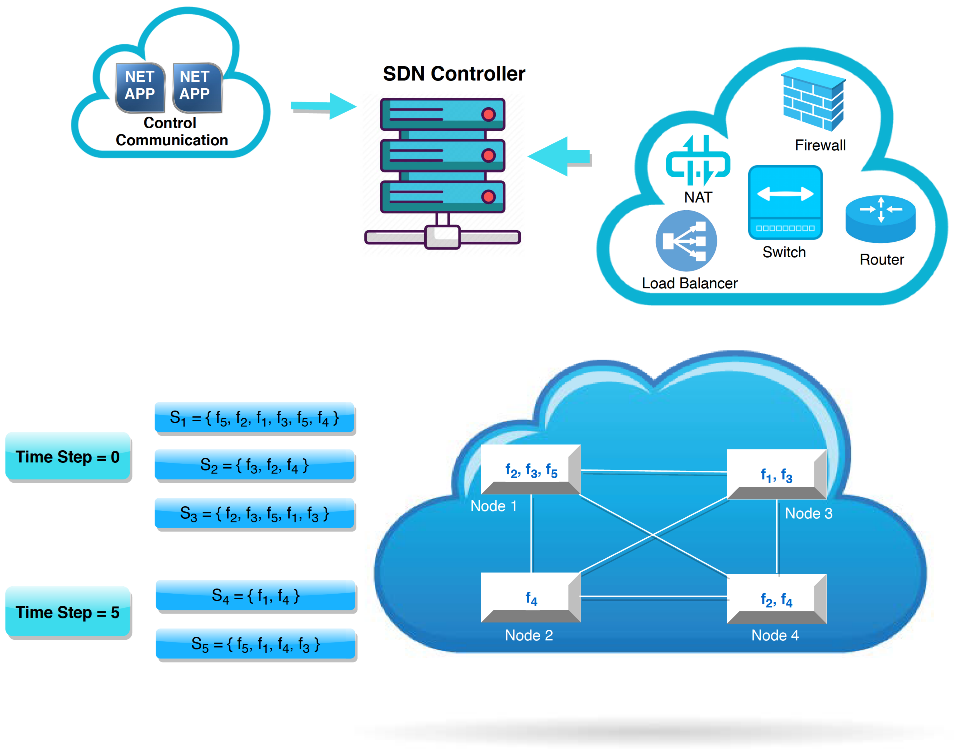
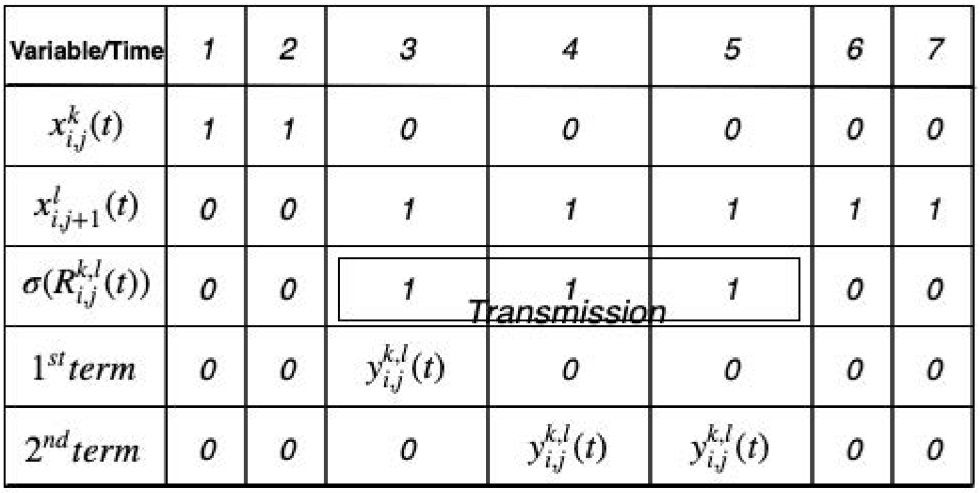


Figure 5.1: Network structure with VM node capabilities



Variable/Time	1	2	3	4	5	6	7
$x_{i,j}^{k}(t)$	1	1	0	0	0	0	0
$x_{i,j+1}^{l}(t)$	0	0	1	1	1	1	1
$\sigma(R_{i,j}^{k,l}(t))$	0	0	1	1 Transmission	1	0	0
1^{st} *term*	0	0	$y_{i,j}^{k,l}(t)$	0	0	0	0
2^{nd} *term*	0	0	0	$y_{i,j}^{k,l}(t)$	$y_{i,j}^{k,l}(t)$	0	0

Figure 5.2: Example of the cost function of objective 2

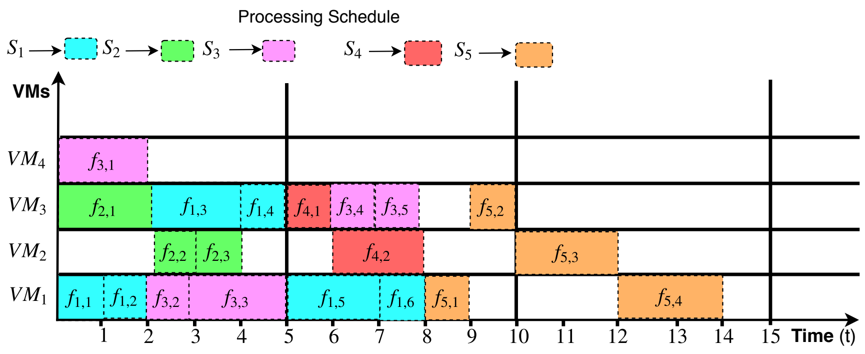


Figure 5.3: one of the possible mappings including all incoming VNFs of service requests

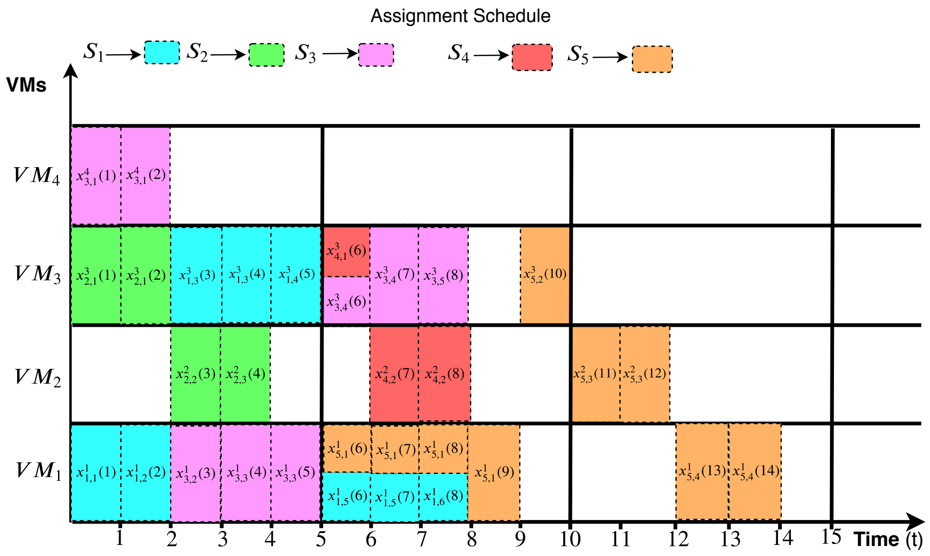


Figure 5.4: All possible values of variable x that reflect the assignment schedule for all incoming service requests every 5 time steps

VNF of the service request s_i to the k-th VM at t-th time interval. In other words,

$$x_{i,j}^{k}(t) = \begin{cases} 1 & \text{If the } j\text{-th VNF of the service} \\ & \text{request } s_i \text{ has been assigned to} \\ & \text{the } k\text{-th VM at } t\text{-th time interval} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

The function $U(\alpha)$ is defined as

$$U(\alpha) = \begin{cases} 1 & \text{for} \quad \alpha > 0 \\ 0 & \text{for} \quad \alpha \leq 0 \end{cases}$$

T is the maximum number of time-steps, i.e., no further service request arrives after T-th time-step and all previously accepted service requests should be processed by T-th time-step.

The first term in the cost function $Z_1(x)$ defined in (5.1), (i.e., $x_{i,1}^k(1)$) counts the number of service requests that are accepted at time step $t = 1$.

While the second term of $Z_1(x)$ (i.e., $\sum_{t=2}^{T} U(x_{i,1}^k(t) - x_{i,1}^k(t-1))$) is for counting the number service requests that are accepted at the following time steps, i.e., $t > 1$. It is assumes that a service is accepted if it can be processed within E time-steps of its arrival, otherwise it is not accepted. Also, when more than one service requests are assigned to the same VM, the service requests are processed based on their priority and the time left until their processing deadline (i.e., the E time steps since arrival of the service request).

5.2.2 Minimizing the Number of Bottleneck links

The available transmission capacity of link (k,l) is the difference between its transmission capacity ($lc_{k,l}$) and its current traffic $\sum_{i'=1}^{|S|} \sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t)$. When the current bandwidth demand (d_{ij}) is higher than the available transmission capacity of the link, i.e., $d_{i,j} > lc_{k,l} - \sum_{i'=1}^{|S|} \sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t)$, the link (k,l) becomes an unscheduled bottleneck.

The aim of the objective defined in this subsection is to minimize the number of unscheduled bottlenecks. In doing so, we defined the following binary variable,

$$y_{i,j}^{k,l}(t) = \begin{cases} 1 & \text{if} \quad d_{i,j} > lc_{k,l} - \sum_{i'=1}^{|S|} \sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t), \\ 0 & \text{Otherwise.} \end{cases} \tag{5.3}$$

where k represent the current VM where the j-th VNF is finished and l represent the next VM where the $(j+1)$-th VNF will be processed. Formally, this objective function is defined as below,

$$\min_x Z_2(x) = \sum_{t=1}^{T}\sum_{i=1}^{|S|}\sum_{j=1}^{|s_i|-1}\sum_{k=1}^{|\mathcal{V}|}\sum_{l=1}^{|\mathcal{V}|} x_{i,j}^{k}(t)\cdot y_{i,j}^{k,l}(t)\cdot x_{i,j+1}^{l}(t+1) + x_{i,j+1}^{l}(t)\cdot \sigma(R_{i,j}^{k,l}(t))\cdot x_{i,j+1}^{l}(t+1)\cdot y_{i,j}^{k,l}(t), \tag{5.4}$$

where

$$\sigma(R_{i,j}^{k,l}(t)) = \begin{cases} 1 & \text{if} \quad R_{i,j}^{k,l}(t) > 0, \\ 0 & \text{Otherwise.} \end{cases} \tag{5.5}$$

In the cost function defined in (5.4), the first term $(x_{i,j}^{k}(t)\cdot y_{i,j}^{k,l}(t)\cdot x_{i,j+1}^{l}(t+1))$ corresponds to the case where the processing of the j-th VNF of s_i has been finished in the k-th VM and next VNF of s_i (i.e., $j+1$-th VNF) has to be processed in the l-th VM.

Thus, at time interval t, a transmission will start from k-th VM to l-th VM. If this transmission is not finished in the t-th time interval, the second term in the cost function (5.4) (i.e., $x_{i,j}^{l}(t)\cdot\sigma(R_{i,j}^{k,l}(t))\cdot x_{i,j}^{l}(t+1)\cdot y_{i,j}^{k,l}(t)$) will be triggered if the link (k,l) becomes an unscheduled bottleneck. The function $\sigma(R_{i,j}^{k,l}(t))$ is for verifying if there is any traffic in the link (k,l) at time interval t, due to the eventual processing of j-th VNF of s_i in the l-th VM. Also, in Fig. 5.2, an example is provided which shows that during the transmission between two VMs, in the first time-step, the first term of the cost function $Z_2(x)$ determines if the link has become an unscheduled bottleneck, while in the following time-steps, the second term of the cost function $Z_2(x)$ determines if the link has become an unscheduled bottleneck.

5.2.3 Minimizing the Overall Processing Time

The processing time of a service request is the number of time steps from the time that the service request was accepted until its processing is finished. From each service request's point of view, it is important to minimize its processing time. This can be reflected in the following cost function,

$$\min_x Z_3(x) = \sum_{i=1}^{|S|} \sum_{j=1}^{|s_i|} \sum_{k=1}^{|\mathcal{V}|} \sum_{t=1}^{T} x_{i,j}^{k}(t). \tag{5.6}$$

In the time steps before s_i is accepted and after its processing is finished, all values of $x_{i,j}^k(t)$ for $j = 1,...,|s_i|$, $k = 1,...,|\mathcal{V}|$ is zero. Therefore, for the accepted service requests, their total processing time is calculated in the cost function Z_3.

Comparing the cost functions $Z_1(x)$ and $Z_3(x)$, it is obvious that $Z_1(x)$ prolongs the processing time to maximize the number of accepted service requests, while considering the constraint imposed by the processing deadline E for each service request. On the other hand $Z_3(x)$ aims at minimizing the overall processing time of all accepted service requests.

5.2.4 Constraints

Here, we describe the constraints that have been considered and imposed to the optimization problem.

1. **Forwarding Constraint:**

   The $(j+1)$-th VNF of the s_i can not be assigned unless the processing of previous VNF has been completed. This dictates the following constraint,

$$\begin{aligned} &x_{i,j+1}^{l}(t+1) = 0 \quad \text{if} \\ &x_{i,j}^{k}(t)\cdot(1 - x_{i,j}^{k}(t+1)) = 0 \,\text{and}\, x_{i,j+1}^{l}(t) = 0, \end{aligned} \tag{5.7}$$

   for $i = 1,...,|\mathscr{S}|$, $j = 1,...,|s_i|-1$, $k,l = 1,...,|\mathcal{V}|$. The term $x_{i,j}^k(t)\cdot(1 - x_{i,j}^k(t+1))$ is equal to one only in the time step where processing of the j-th VNF of service request s_i is finished in the k-th VM.

2. **Assignment & Traffic Constraint:**

   The link (k,l) can not have any traffic due to the $(j+1)$-th VNF of service request

s_i if the $(j+1)$-th VNF of s_i has not been assigned to the l-th VM. This constraint dictates the relation between $R^{k,l}_{i,j}(t)$ and $x^l_{i,j+1}(t)$, where it can be written as

$$R^{k,l}_{i,j}(t) = 0 \text{ if } x^l_{i,j+1}(t) = 0, \tag{5.8}$$

for $i = 1, ..., |\mathscr{S}|$, $j = 1, ..., |s_i| - 1$ and $(k,l) \in \mathscr{E}$.

3. **Link Capacity Constraint:**
   The total traffic flow passing through every link (k,l) (i.e., $\sum_{i=1}^{|\mathscr{S}|}\sum_{j=1}^{|s_i|} R^{k,l}_{i,j}(t)$) should not exceed its capacity ($lc_{k,l}$). This can be shown as the following constraint.

$$\sum_{i=1}^{|\mathscr{S}|}\sum_{j=1}^{|s_i|} R^{k,l}_{i,j}(t) \leq lc_{k,l}, \quad \forall (k,l) \in \mathscr{E}. \tag{5.9}$$

5.2.5 Example of Mapping and Scheduling for the total incoming number of service requests

As shown in Fig. 5.1, the network structure consists of 4 VMs node and each VM support subset of VNFs on it. Moreover, a set of incoming service requests $s_1, ..., s_5$ arrived to the cloud at different time step, 3 service requests s_1, s_2, s_3 are arrived at time step $t = 0$ and 2 service requests s_4, s_5 are arrived at time step $t = 5$. The VNFs of these incoming service requests at time step $t = 0$ are $s_1 = \{f^u_5, f^u_2, f^u_1, f^u_3, f^u_5, f^u_4\}$, $s_2 = \{f^u_3, f^u_2, f^u_4\}$, and $s_3 = \{f^u_2, f^u_3, f^u_5, f^u_1, f^u_3\}$. And, the VNFs of the service requests arrived at time step $t = 5$ are $s_4 = \{f^u_1, f^u_4\}$ and $s_5 = \{f^u_5, f^u_1, f^u_4, f^u_3\}$.

Assume that each VM can process one VNF at a time, Fig. 5.4 shown one of the possibilities to map and schedule the incoming service requests s_i for $i = 1, ..., 5$ when they arrived at the cloud at different. Comparing Fig. 5.3 and Fig. 5.4, we can clearly see that some VNF of the accepted service requests can be assigned to a VM but it is not being process yet. This can be happened either due to the processing queue or due to ongoing transmission. At each time step, it is assumed that each VM can process only one VNF of the accepted service requests, while more than one VNF of the accepted service requests can be assigned to this VM.

The following steps explain in details how the incoming service request s_3 is processed once it's arrived to the cloud at time step $t = 0$ until the last function of the request is processed at $t = 8$. Initially s_3 is assigned to VM_4 and it needs two time steps to be processed, thus $x^1_{3,1}(1) = 1$ and $x^1_{3,1}(2) = 1$. The second and the third VNF of s_4 is assigned to VM_1 at the third, fourth and fifth time steps, so $x^1_{3,2}(3) = 1$, $x^1_{3,3}(4) = 1$ and $x^1_{3,3}(5) = 1$. In the fifth time step $t = 5$, the fifth VNF of s_3 is assigned to process on VM_3, where the the first VNF of the s_3 is arrived to the cloud and assigned to VM_3 at the same time $t = 5$. As a result of the weigh for the the first VNF of the s_4 is larger than the fifth VNF of s_3, so the fifth VNF of s_3 has to wait in the queue one time step until the first VNF of the s_3 finish its processing as shown in Fig. 5.4. So, the fifth VNF of s_3 will process on VM_3 at time step $t = 7$, $x^3_{3,4}(7) = 1$. In the eighth time step, the fifth VNF of s_3 is assigned to process on the same VM_3, thus $x^3_{3,5}(8) = 1$. It takes 8 time steps to finish processing the accepted service request s_3.

5.3 Algorithm Design

Genetic algorithms (GAs) rely on similarity with the genetic structure and behavior of chromosomes within a population of individuals using the following basic idea: Individuals in a population compete for resources and mates. Those individuals who win the most in each competition will create more offspring than those individuals with poor achievements.

Here, to solve the scheduling and mapping problem, we apply two different approaches of GA, namely Pareto dominance and decomposition approaches as explained in Chapter 3. The popular algorithms of Pareto dominance and decomposition approaches are NSGA-II [8] and MOEA/D algorithm [7], respectively. Authors in [101] developed a

Table 5.1: Assignment Schedule for Example 5.2.5: The values of variable x which are equal to one.

t	s_1	s_2	s_3	s_4	s_5
1	$x^1_{1,1}(1)$	$x^3_{2,1}(1)$	$x^4_{3,1}(1)$		
2	$x^1_{1,2}(2)$	$x^3_{2,1}(2)$	$x^4_{3,1}(2)$		
3	$x^3_{1,3}(3)$	$x^2_{2,2}(3)$	$x^1_{3,2}(3)$		
4	$x^3_{1,3}(4)$	$x^2_{2,3}(4)$	$x^1_{3,3}(4)$		
5	$x^3_{1,4}(5)$		$x^1_{3,3}(5)$		
6	$x^1_{1,5}(6)$		$x^3_{3,4}(6)$	$x^3_{4,1}(6)$	$x^1_{5,1}(6)$
7	$x^1_{1,5}(7)$		$x^3_{3,4}(7)$	$x^2_{4,2}(7)$	$x^1_{5,1}(7)$
8	$x^1_{1,6}(8)$		$x^3_{3,5}(8)$	$x^2_{4,2}(8)$	$x^1_{5,1}(8)$
9					$x^1_{5,1}(9)$
10					$x^3_{5,2}(10)$
11					$x^2_{5,3}(11)$
12					$x^2_{5,3}(12)$
13					$x^1_{5,4}(13)$
14					$x^1_{5,4}(14)$

modified version of MOEA/D called CMOEA/D-DE-ATP to solve constraint optimization problems.

Without performing the original CMOEA/D-DE-ATP algorithm itself, we use the framework and apply novel changes. In **step 1** of Algorithm 2, the network parameters are initialized. These include the set of incoming service requests ($\mathscr{S}$), their priority weights $W = \{w_i | i = 1,...,|\mathscr{S}|\}$, the processing deadline (E), the number of VMs ($|\mathscr{V}|$), the set of VNFs supported by each VM (denoted by VF_k for $k = 1,...,|\mathscr{V}|$), and the transmission capacity for each virtual link $\{lc_{k,l} | (k,l) \in \mathscr{E}\}$. Then, we determine the stopping criteria of the algorithm. The stopping criteria in LDNSGAII depends on maximum number of evaluation which is set to 25000. While the stopping criteria in RSAMOAD depends on the population size set to 100 and the number of iteration set to 300 (300*100).

As shown in **step 4**, the population size is chosen random for both algorithms. In

step 5, the algorithm checks the feasibility for output solution of these population by checking all constraints proposed in (5.7), (5.8), (5.9). Then the algorithm calculates the fitness function for each population and apply all genetic operations to the proposed solutions from these populations.

The LDNSGAII algorithm utilizes a binary representation operator, the tournament selection as the selection operator, and a binary crossover operator with mutation rate equal to the inverse of the number of variables.

The RSAMOAD algorithm uses a binary representation operator, the mate selection as the selection operator, and the DE crossover operator with mutation rate equal to the inverse of the Parameters' dimension. Using both algorithms, the final output is a near to optimal mapping and scheduling for the incoming service requests.

Regarding the complexity of the algorithm, there are two main tasks for each population. First task is the calculation of the cost functions Z_1, Z_2 and Z_3 (given in (5.1), (5.4) and (5.6), respectively), where their complexities are of order $\mathcal{O}(|\mathcal{S}||\mathcal{V}|T)$, $\mathcal{O}\left(Q|\mathcal{V}|^2\right)$ and $\mathcal{O}(Q|\mathcal{V}|T)$, respectively, with $Q = \sum_{i=1}^{|\mathcal{S}|} s_i$. The second task is the calculation of the constraints (5.7), (5.8), (5.9), where the complexity of every one of them is equal to $\mathcal{O}\left(Q|\mathcal{V}|^2\right)$. Note that the complexity of GA algorithm depends on the algorithm parameters as well. Some of these parameters include population size and number of iteration.

5.4 Simulation Results

In this section, we present the simulation results for the proposed scheduling and mapping algorithms explained in the previous section.

5.4.1 Simulation Set up

The network structure in this study contains four network cases: small, medium, large and extra-large consists of 100, 200, 300 and 400 service requests and 10, 20, 30 and 40

Algorithm 2 RSAMOAD and RA-NSGA-II

```
1: Input:
     • Network parameters:
       $\mathscr{S}, E, W, \mathscr{V}, \mathscr{E}, F, VF, \{tc_{k,l} | (k,l) \in \mathscr{E}\}$
     • The Algorithm stopping criterion.
2: Output:
     The best feasible solution for ($x_{ij}^k(t)$) which represent the best mapping and
     scheduling for VNF of the incoming service requests. This solution has the best
     value for the cost functions $Z_1$, $Z_2$, $Z_3$ and it satisfies all the constraints (5.7),
     (5.8), (5.9).
3: Define:
     • X = $\{x_{ij}^k(t) | i = 1,...,|\mathscr{S}|, j = 1,...,|s_i|, k \in \mathscr{V}, t = 1,...,T\}$
4: Initialize:
     • Generate the initial population $X^1,\dots,X^N$ uniformly in random.

5: for every $X^i = X^1,...,X^N$ do
6:     if $X^i$ satisfying all the proposed constraints then
     • Evaluate the proposed cost functions $Z_1(x)$, $Z_2(x)$, $Z_3(x)$ according to (5.1), (5.4),
       (5.6)
     • applying the genetic (i.e., representation, selection, crossover and mutation)
       operators to the result
     • Choosing the best Feasible Solution after applying the previous parameters
     •     end if
7: end for
8: Selecting the best result from all populations.
9: Termination criterion:
     • if the stopping criterion is satisfied then,
       Stop and return the final output as the best mapping and scheduling.
```

VM nodes, respectively. The simulations is implemented on Intel Xeon Gold 6150 2.7GHz Core, 2.7GHz, with 32 GB of RAM running Linux platform, and using the MATLAB and Java as a programming languages. Two different scenarios are applied in this model to evaluate the performance of the proposed algorithms. In the first scenario, we assume that non uniform VNFs service requests arrive to the cloud in different time steps (i.e at time step = 0, 5, 15 and 20) to be processed by installed VMs. Each of these service requests does not have any processing deadline to finish it's processing. In the second scenario, we assume that non uniform VNFs service requests arrive to the cloud in different time steps but each of these requests have a processing deadline to be processed in the VMs before reaching that deadline. At each scenario, the proposed algorithm implemented to find the near to optimal solution for every objective individual (Scenario 1), any two combination of the aforementioned objectives (Scenario 2) or for the three objectives together (Scenario 3).

5.4.2 Evaluation Scenarios

The final results of the first scenario is shown in Fig 5.5, Fig 5.6 and Fig 5.7, where the near to optimal value of the every objectives Z_1, Z_2 and Z_3 are reported for all four network cases by applying both RSAMOAD and RS-NSGA-II algorithms. We can clearly see from this figure that Z_2 and Z_3 values decrease gradually when the number of incoming service requests (without processing deadline) increases, while Z_1 increase slightly by applying both algorithms. Scenario 2 illustrates the values for any combination between two of the proposed objectives. For instance, Fig 5.8 shows the final results for the first and the second cost functions (Z_1 & Z_2), Fig 5.9 shows the final results for the first and the second cost functions (Z_2 & Z_3) and Fig 5.10 shows the final results for the first and the second cost functions (Z_1 & Z_3) by using the two developed algorithms in all implementation.

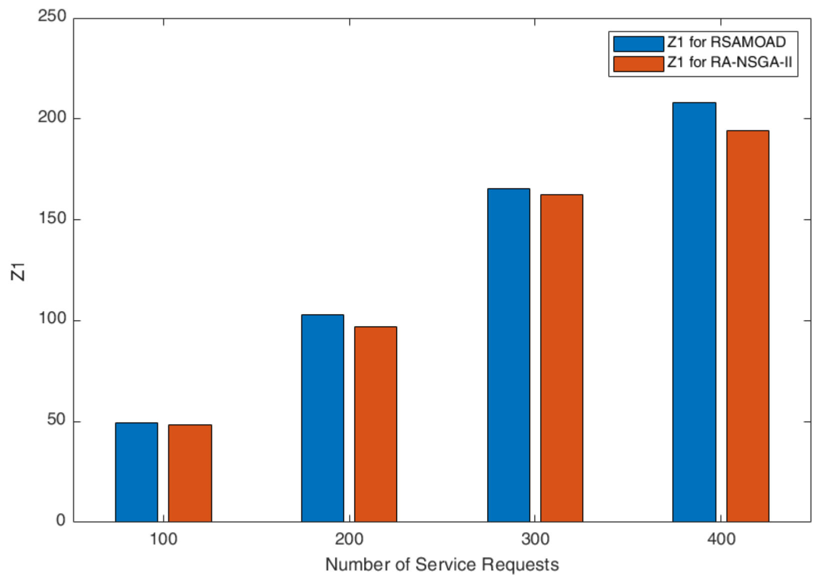


Figure 5.5: The Final results for the first cost function (Z_1) by using the two developed algorithms

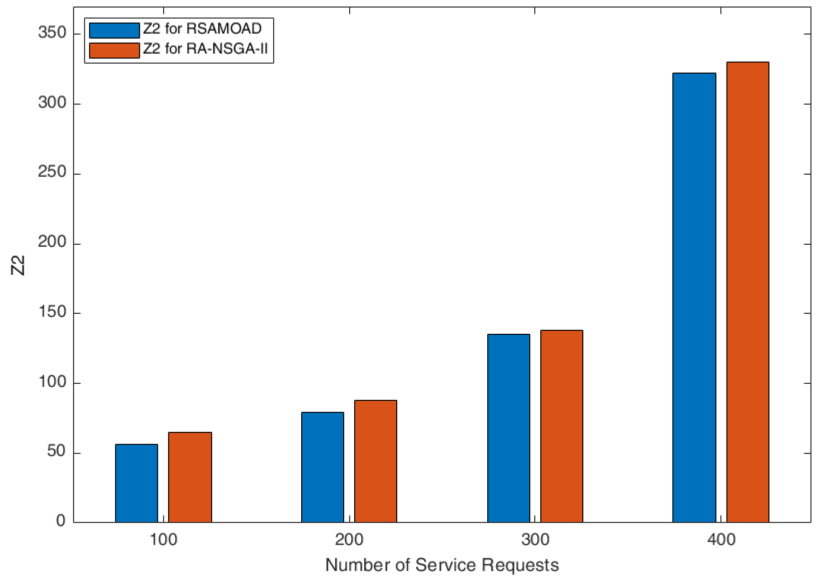


Figure 5.6: The Final results for the second cost function (Z_2) by using the two developed algorithms

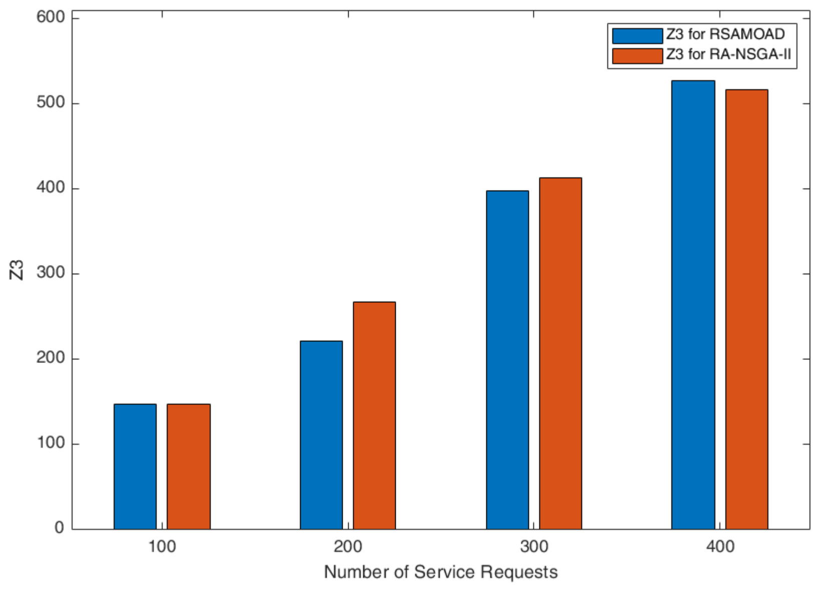


Figure 5.7: The Final results for the third cost function (Z_3) by using the two developed algorithms

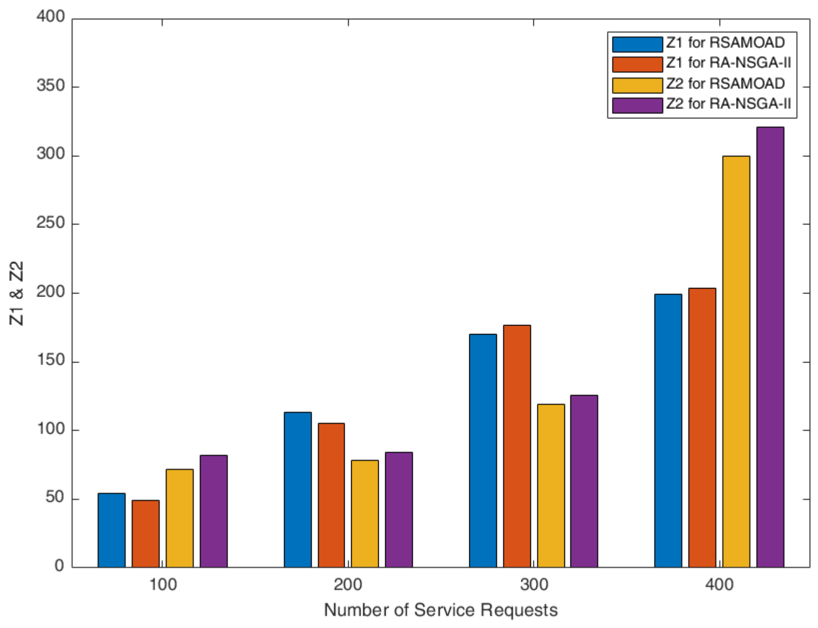


Figure 5.8: The Final results for the first and the second cost functions (Z_1 & Z_2) by using the two developed algorithms

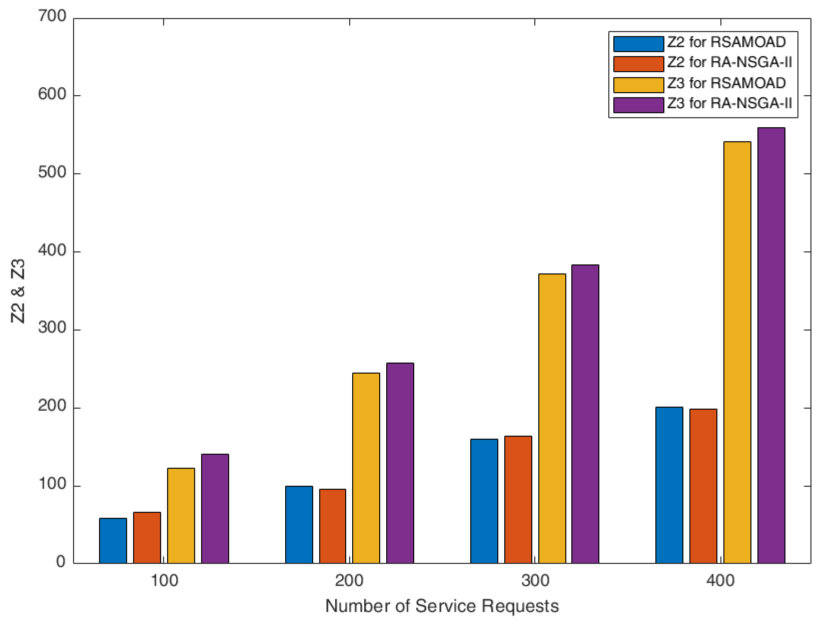


Figure 5.9: The Final results for the second and the third cost functions (Z_2 & Z_3) by using the two developed algorithms

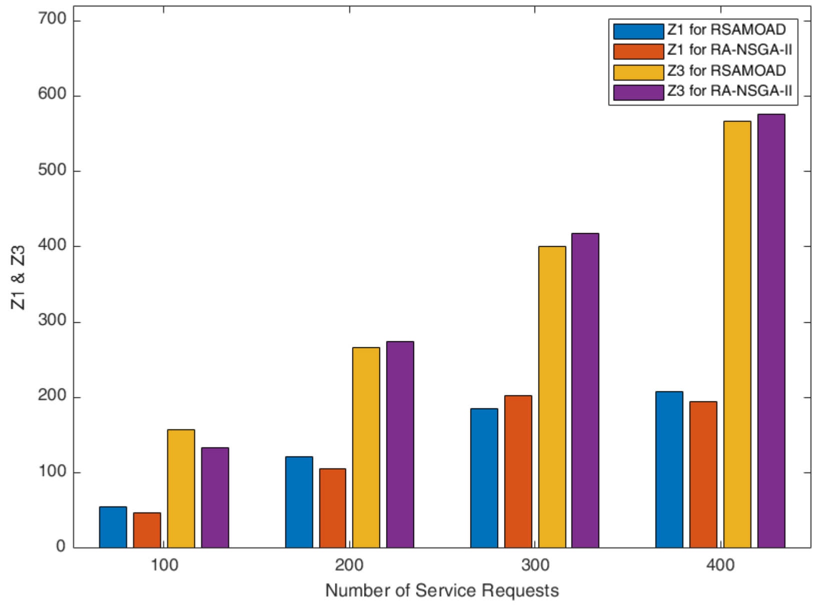


Figure 5.10: The Final results for the first and the third cost functions (Z_1 & Z_3) by using the two developed algorithms

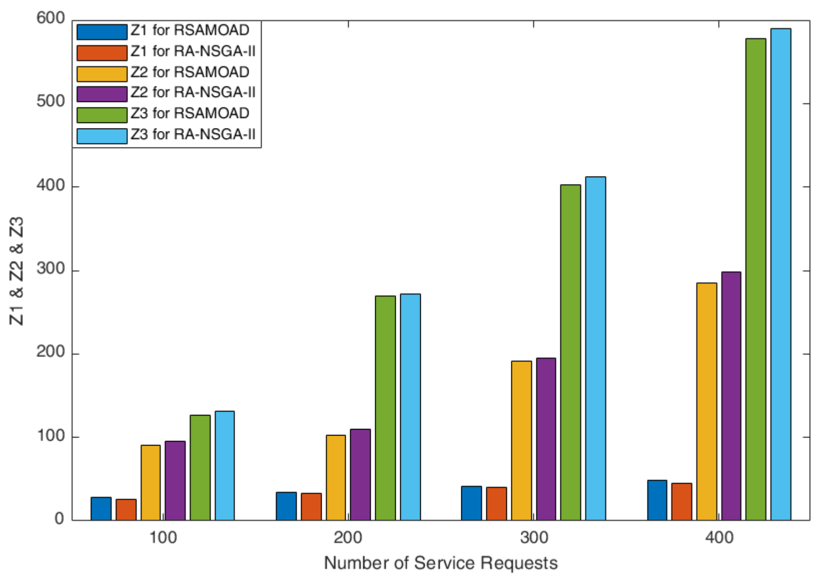


Figure 5.11: The Final results for the three cost functions by using the two developed algorithms considering the expiry time for every service request

Going to the third scenario, the result of Z_1, Z_2 and Z_3 achieved by applying both RSAMOAD and RS-NSGA-II algorithms considering the expiry date and without considering the expiry date for the service requests are shown in Fig. 5.11 and Fig. 5.12 respectively.

What can be clearly seen that the difference between two figures is the value of the first objective decreased. The main reason for this problem is that some of the service requests have not been accepted by the algorithm due to their processing deadline is bigger than the the available time in the VMs.

In Fig. 5.13, We study the impact of receiving four different sets of incoming service request: 500, 600, 700 and 800, respectively, on the cost values and performance time. The simulation results in all scenarios indicate that the RS-NSGA-II algorithm generally has a better performance in terms of Z_1 (i.e., accepting larger number of service requests), but this comes at the cost of higher values for Z_2 (i.e., more congestion) and Z_3 (i.e., longer processing time).

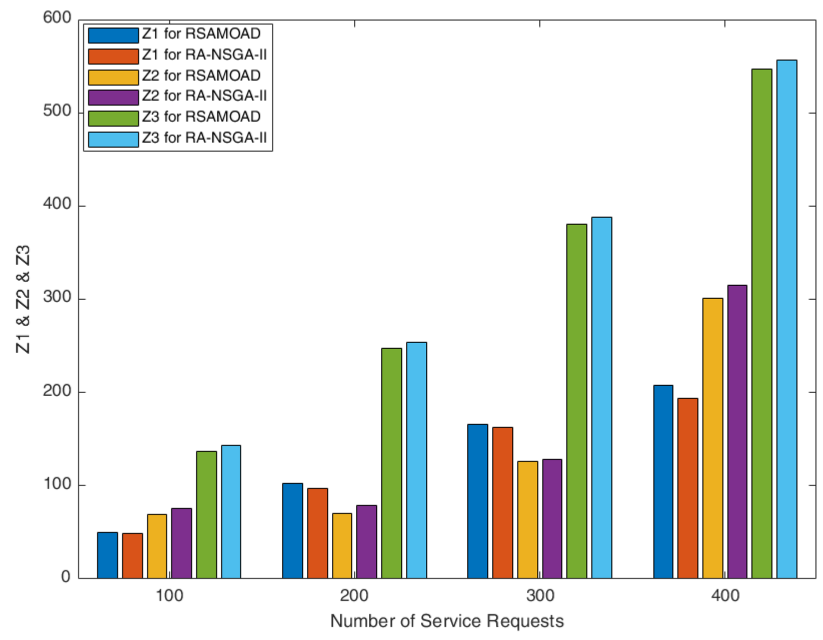


Figure 5.12: The Final results for the three cost functions by using the two developed algorithms without considering the expiry time for every service request

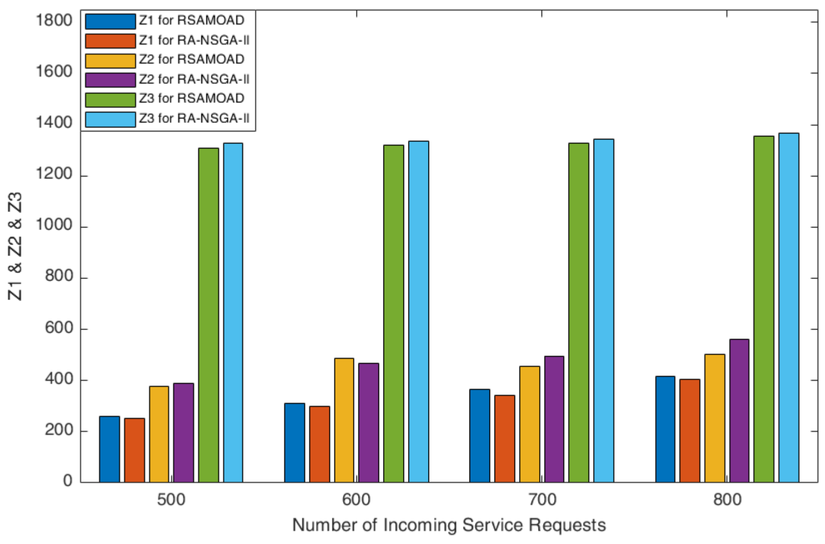


Figure 5.13: The Final results for the three cost functions considering different number of incoming service requests along with fixed number of VMs is 50

5.5 Conclusion

This chapter presents a multi-objective mapping and scheduling algorithm for a set of incoming requests arrive a cloud in different time steps. Given a set of VMs installed on the cloud and connected to each other using virtual links, every VM support particular number of VNF. Those incoming requests need to process in order on these VMs. The proposed algorithm determines the optimal scheduling and mapping for these incoming requests. There are three main objectives that are proposed in this chapter: maximizing the number of accepted service requests, minimizing the number of bottleneck links and the overall processing time. A mathematical formulation is proposed to describe the problem and two different developed algorithms are provided to solve the problem, namely RSAMOAD and RA-NSGA-II. The final results illustrate that RSAMOAD algorithm perform better than RA-NSGA-II algorithm in term of optimal solution for the objective values.

CHAPTER 6

JOINT OPTIMIZATION OF MAPPING VNF CHAINS AND AND SCHEDULING FOR THE SERVICE REQUESTS

6.1 Introduction

This chapter presents a new formulation for the NFV-RA problem by introducing a new optimization model to cover different network scenarios of the NFV-RA problem. In practice, a modern network service is expressed by a service request chain which is composed of a sequence of VNF functions to be processed in order through one or multiple VMs. One such service request example is shown in Fig. 6.1, where different service requests arrive to the cloud at different times and need to be processed through the installed VMs. Every incoming request contains one or more of the following five network functions: network address translation (NAT), firewall, load balance, gate way, and intrusion detection system (IDS). Moreover, every VM supports one or more of the five network functions, has limited network resources (CPU type, capacity). The VMs are connected together via virtual links.

The incoming service request should go through these network functions supported by VMs in order to accomplish the VNFs of the service request. As a result of the limited capacity for every link, an unscheduled bottleneck can be happening in some of these links. Furthermore, as a result of limited computation capacity for every VM, a processing delay can be also happening in every VM.

The major goal of this chapter is to provide a good understanding of the issues related to the mapping and scheduling for a set of incoming service requests to be processed in order through different VMs while taking into consideration the network constraints and different objectives. Generally, there is a set of VNFs request from different clients that need to be processed by network functions on VMs/servers, where each individual request requires a subset of the network functions. We provide an efficient SM-MOEA/D and SM-NSGA-II heuristic algorithm for the automatic mapping and scheduling for the VNFs of the incoming service requests.

In doing so, the given processing order of the VNFs belonging to each service request should be considered, as well as the constraints imposed due to the time limit and the priority for processing each service request. The limited capacity of each VM and the links among them introduces additional restrictions that should be addressed in the model.

6.2 System Model

This subsection includes the developed model for the network and the scenarios addressed in this chapter.

An example network structure considered in this chapter is presented in Fig. 6.1. We model the network as an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. $\mathscr{V} = \{k | k = 1, ..., |\mathscr{V}|\}$ is the set of VM nodes and $\mathscr{E} = \{(k,l) | k,l = 1, ..., |\mathscr{V}|, k \neq l\}$ is the set of virtual links between VMs. $F = \{f_1^u, f_2^u, ..., f_{|F|}^u\}$ is the set of unique VNFs supported in the cloud. The VNFs supported by the k-th VM is denoted by VF_k for $k = 1,...,|\mathcal{V}|$. For instance, if VM_2 runs $\{f_2^u, f_5^u\}$ as shown in Fig. 6.3, that means only VNFs f_2^u and f_5^u of the incoming request can be executed in VM_2. Change according to the example. The number of VNFs that the k-th VM (capacity) can process simultaneously is denoted by c_k for $k = 1,...,|\mathcal{V}|$. Let $lc_{k,l}$ for $k \neq l = 1,...,|\mathcal{V}|$ be the transmission capacity of the virtual link (k,l). In this chapter, we assume that there is a virtual link between every two VMs. We define the incoming sequence of service requests as $\mathscr{S} = [s_1,...,s_{|\mathscr{S}|}]$ and each request $s_i \in \mathscr{S}$ for $i = 1,...,|\mathscr{S}|$ has a sequence with different order of VNFs. Let $f_{i,j}$ for $i = 1,...,|\mathscr{S}|$ and $j = 1,...,|s_i|$ be the j-th VNF of the i-th incoming service request (s_i). We assume that every request s_i for $i = 1,2,...,|\mathscr{S}|$ has different number of VNFs.

The VNF f_{ij} of the incoming request should be processed in the given order through the selected VMs. Furthermore, we assume that each service request s_i has a time limit or expiry time (denoted by E_i), where its processing should be finished before this time limit.

We define $d_{i,j}$ for $i = 1,2,..,|\mathscr{S}|$, $j = 1,2,...,|s_i| - 1$ as the bandwidth demand of the i-th service request s_i after processing its j-th VNF, and $R_{i,j}^{k,l}(t)$ for $i = 1,2,..,|\mathscr{S}|$, $j = 1,2,...,|s_i| - 1$ and $(k,l) \in \mathscr{E}$ as the traffic flow of the i-th service request after processing its j-th VNF, which is passing through the virtual link (k,l) at t-th time interval.

In next four subsections, four main objectives considered in this chapter for mapping and scheduling the service requests are presented.

6.2.1 Maximizing The Acceptance Rate

In this subsection, the first objective function is introduced. This objective function aims at maximizing the number of accepted service requests. This objective function can be defined as below,

$$\max_x Z_1(x) = \sum_{i=1}^{|S|} \sum_{k=1}^{|\mathcal{V}|} w_i \cdot \left(x_{i,1}^k(1) + \sum_{t=2}^{T} U\left(x_{i,1}^k(t) - x_{i,1}^k(t-1) \right) \right) \tag{6.1}$$

where w_i is the weight of each service request s_i, which reflects its priority. This weight is normalized to 1. $x_{ij}^k(t)$ is a binary variable denoting whether if the j-th VNF of the service request s_i has been assigned to the k-th VM at t-th time interval. In other words,

$$x_{i,j}^k(t) = \begin{cases} 1 & \text{If the } j\text{-th VNF of the service} \\ & \text{request } s_i \text{ has been assigned to} \\ & \text{the } k\text{-th VM at } t\text{-th time interval,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

The function $U(\alpha)$ is defined as

$$U(\alpha) = \begin{cases} 1 & \text{for} \quad \alpha > 0 \\ 0 & \text{for} \quad \alpha \le 0 \end{cases}$$

No further service request arrives after T-th time-step and all previously accepted service requests should be processed by T-th time-step. The first term in the cost function $Z_1(x)$ defined in (6.1), (i.e., $x_{i,1}^k(1)$) counts the number of service requests that are accepted at time step $t = 1$. While the second term of $Z_1(x)$ (i.e., $\sum_{t=2}^{T} U(x_{i,1}^k(t) - x_{i,1}^k(t-1))$) is for counting the number service requests that are accepted at the following time steps, i.e., $t > 1$. It is assumes that service requests are accepted considering their expiry time (E_i). In other words if a service request can not be finished within its expiry time (E_i), then it will never be accepted. Also, when the number service requests assigned to a VM is more than its capacity, then the service requests are processed based on their priority. The maximum number of time-steps T represents the time window that network parameters are assumed to be steady.

6.2.2 Optimizing Link Utilization

The available transmission capacity of link (k,l) is the difference between its transmission capacity ($lc_{k,l}$) and its current traffic $\sum_{i'=1}^{|S|} \sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t)$. When the current bandwidth demand (d_{ij}) is higher than the available transmission capacity of the link, i.e., $d_{i,j} > lc_{k,l} - \sum_{i'=1}^{|S|}\sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t)$, the link (k,l) becomes an unscheduled bottleneck. The aim of the objective defined in this subsection is to minimize the number of unscheduled bottlenecks. In doing so, we defined the following binary variable,

$$y_{i,j}^{k,l}(t) = \begin{cases} 1 & \text{if} \quad d_{i,j} > lc_{k,l} - \sum_{i'=1}^{|S|}\sum_{j'=1}^{|s_i|} R_{i',j'}^{k,l}(t), \\ 0 & \text{Otherwise.} \end{cases} \tag{6.3}$$

where k represent the current VM where the j-th VNF is finished and l represent the next VM where the $j+1$-th VNF will be processed. Formally, this objective function is defined as below,

$$\begin{aligned} &\min_x Z_2(x) = \\ &\sum_{t=1}^{T}\sum_{i=1}^{|S|}\sum_{j=1}^{|s_i|-1}\sum_{k=1}^{|\mathcal{V}|}\sum_{l=1}^{|\mathcal{V}|} x_{i,j}^{k}(t)\cdot y_{i,j}^{k,l}(t)\cdot x_{i,j+1}^{l}(t+1) \\ &\qquad + x_{i,j+1}^{l}(t)\cdot \sigma(R_{i,j}^{k,l}(t))\cdot x_{i,j+1}^{l}(t+1)\cdot y_{i,j}^{k,l}(t), \end{aligned} \tag{6.4}$$

where

$$\sigma(R_{i,j}^{k,l}(t)) = \begin{cases} 1 & \text{if} \quad R_{i,j}^{k,l}(t) > 0, \\ 0 & \text{Otherwise,} \end{cases} \tag{6.5}$$

In the cost function defined in (6.4), the first term ($x_{i,j}^{k}(t)\cdot y_{i,j}^{k,l}(t)\cdot x_{i,j+1}^{l}(t+1)$) corresponds to the case where the processing of the j-th VNF of s_i has been finished in the k-th VM and next VNF of s_i (i.e., $j+1$-th VNF) has to be processed in the l-th VM. Thus, at time interval t, a transmission will start from k-th VM to l-th VM. If this transmission is not finished in the t-th time interval, the second term in the cost function (6.4) (i.e., $x_{i,j}^{l}(t)\cdot\sigma(R_{i,j}^{k,l}(t))\cdot x_{i,j}^{l}(t+1)\cdot y_{i,j}^{k,l}(t)$) will be triggered if the link (k,l) becomes an unscheduled bottleneck. The function $\sigma(R_{i,j}^{k,l}(t))$ is for verifying if there is any traffic in the link (k,l) at time interval t, due to the eventual processing of j-th VNF of s_i in the l-th VM.

In Table 6.1, an example of mapping and scheduling has been provided. Also, in Fig. 6.4, an example is provided which shows that during the transmission between two VMs, in the first time-step, the first term of the cost function $Z_2(x)$ determines if the link has become an unscheduled bottleneck, while in the following time-steps, the second term of the cost function $Z_2(x)$ determines if the link has become an unscheduled bottleneck.

6.2.3 Minimizing the Overall Processing Time

From the cloud manager's point of view, it is valuable to minimize the time it takes to process all accepted service requests. This can be formulated as the following cost function,

$$\min_x Z_3(x) = \sum_{i=1}^{|\mathscr{S}|} \sum_{j=1}^{|s_i|} \sum_{k=1}^{|\mathscr{V}|} \sum_{t=1}^{T} x_{i,j}^k(t). \tag{6.6}$$

When processing of service request s_i is finished, all future values of $x_{i,j}^k(t)$ for $j = 1, ..., |s_i|$, $k = 1, ..., |\mathscr{V}|$ are zero. Therefore, for the accepted service requests, their total processing time is calculated in the cost function Z_3.

6.2.4 Minimize the Relative Processing Time

The processing time of an individual service request is the number of time steps starting from the time that the service request was accepted until its processing is finished. The relative processing time is the average of processing times of all accepted service requests. From each individual service request's point of view, it is important to finish its processing as soon as possible and before reaching its expire time. Hence from the service requests point of view, it is important to minimize the relative processing time. This can be reflected in the following cost function,

$$\min_x Z_4(x) = \frac{\sum_{i=1}^{|S|} \left(\sum_{j=1}^{|s_i|} \sum_{k=1}^{|\mathscr{V}|} \sum_{t=1}^{T} x_{i,j}^k(t)\right) \Big/ E_i}{\sum\limits_{i=1}^{|\mathscr{S}|} \sum\limits_{k=1}^{|\mathscr{V}|} x_{i,1}^k(1) + \sum\limits_{t=2}^{T} U\left(x_{i,1}^k(t) - x_{i,1}^k(t-1)\right)}. \tag{6.7}$$

Here E_i is the expiry time for the i-th service request s_i. The numerator of $Z_4(x)$ defined in (6.7) represents the average time it takes to process the accepted service requests normalized by their expiry time. Normalization is to ensure that each term in the summation of the numerator of (6.7) is smaller than one. The denominator of (6.7) is the number of accepted service requests. Hence, $Z_4(x)$ in (6.7) represents the average processing time of service requests normalized by their expiry time.

Comparing four cost functions introduced above, it is obvious that $Z_1(x)$ prolongs the processing time (to maximize the total number of accepted service requests), while $Z_3(x)$ and $Z_4(x)$ minimize the processing time of accepted service requests.

6.2.5 Constraints

Here, we explain the constraints of the model developed in this chapter, and have to be considered in the optimization problem.

1. **VNF's Order:**

   VNFs belonging to a given service request s_i should be processed in the given order. Hence, the processing or transmission of the $(j+1)$-th VNF of service request s_i should start only if the processing of the previous j-th VNF of s_i is completed. In other words, $x^l_{i,j+1}(t+1)$ can be equal to one if $x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1)) = 1$ or $x^l_{i,j+1}(t) = 1$. Or equivalently $x^l_{i,j+1}(t+1)$ is equal to zero for all values of l, if $x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1)) = 0$ and $x^l_{i,j+1}(t) = 0$. This can be stated as the following,

$$\begin{aligned} &x^l_{i,j+1}(t+1) = 0 \quad \text{if} \\ &\qquad x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1)) = 0 \text{ and } x^l_{i,j+1}(t) = 0, \end{aligned} \tag{6.8}$$

   for $i = 1,\ldots,|\mathscr{S}|$, $j = 1,\ldots,|s_i|-1$, $k,l = 1,\ldots,|\mathscr{V}|$. The term $x^k_{i,j}(t)\cdot(1-x^k_{i,j}(t+1))$ is equal to one only in the time step where the processing of the j-th VNF of service request s_i is finished in the k-th VM.

2. **Link Traffic Constraint:** All the assigned traffic of a link should be the traffic of the VNFs assigned to the VM in the destination of the link. This constraint can be written as

$$R_{i,j}^{k,l}(t) = 0 \text{ if } x_{i,j+1}^{l}(t) = 0, \tag{6.9}$$

for $i = 1,...,|\mathscr{S}|$, $j = 1,...,|s_i| - 1$ and $(k,l) \in \mathscr{E}$. Constraint (6.9) means that the link (k,l) can not have any traffic due to the $(j+1)$-th VNF of service request s_i, if the $(j+1)$-th VNF of s_i has not been assigned to the l-th VM. This constraint dictates the relation between $R_{i,j}^{k,l}(t)$ and $x_{i,j+1}^{l}(t)$.

3. **Link Capacity Constraint:** This constraint ensures that the total traffic passing through a link does not exceed the link's capacity.

$$\sum_{i=1}^{|\mathscr{S}|} \sum_{j=1}^{|s_i|} R_{i,j}^{k,l}(t) \le lc_{k,l}, \quad \forall (k,l) \in \mathscr{E}. \tag{6.10}$$

The left hand side of the inequality (6.10) is the total traffic passing through link (k,l), and $lc_{k,l}$ is the capacity of link (k,l).

4. **Expiry Time Constraint:** This constraint guarantees that the processing of every accepted service request will be completed before its expiry-time.

$$\sum_{j=1}^{|s_i|} \sum_{k=1}^{|\mathscr{V}|} \sum_{t=1}^{T} x_{i,j}^{k}(t) \le E_i, \quad \text{for} \quad i = 1,...,|\mathscr{S}|. \tag{6.11}$$

The term $\sum_{j=1}^{|s_i|} \sum_{k=1}^{|\mathscr{V}|} \sum_{t=1}^{T} x_{i,j}^{k}(t)$ is the total processing time of the i-th service request (i.e., s_i) and E_i is the expiry-time of s_i.

6.2.6 Example of Mapping and Scheduling for the total incoming number of service requests

The network structure of this chapter is shown in Fig. 6.1 with 5 VMs, and 4 incoming service requests $s_1,...,s_4$ at time step $t = 0$, 2 incoming service requests s_5, s_6 at time step $t = 5$ and 1 service request s_7 at time step $t = 10$. The VNFs of service requests $s_1,...,s_4$ at time step $t = 0$ are $s_1 = \{f_5^u, f_3^u, f_4^u, f_5^u\}$, $s_2 = \{f_2^u, f_3^u\}$, $s_3 = \{f_4^u, f_1^u, f_4^u, f_2^u, f_5^u\}$, $s_4 = \{f_3^u, f_5^u, f_4^u, f_1^u, f_3^u, f_2^u\}$. And, the VNFs of service requests s_5, s_6 at time step $t = 5$ are $s_5 = \{f_4^u, f_3^u, f_5^u\}$,$s_6 = \{f_4^u\}$, $s_7 = \{f_3^u, f_4^u, f_3^u, f_5^u\}$. And, the VNFs of service requests s_8, s_9 at time step $t = 10$ is $s_8 = \{f_5^u, f_3^u, f_4^u, f_1^u\}$, $s_9 = \{f_4^u, f_2^u, f_5^u\}$. And, the VNFs of service requests s_{10} at time step $t = 15$ is $s_{10} = \{f_1^u, f_2^u, f_3^u, f_4^u\}$. One of the possible mapping and scheduling for theses incoming VNFs of the service requests s_i for $i = 1,...,10$ is shown in Fig. 6.2 and Fig. 6.3. The figures show the processing schedule and assignment schedule for the incoming VNFs at every time step.

Comparing Fig. 6.2 and Fig. 6.3, it is obvious that a VNF of a service request can be assigned to a VM, while it is not being processed. This can be due to two reasons, namely the ongoing transmission or the processing queue. At each time step one or more than one VNF of the accepted service requests can be assigned to a VM. While, each VM can process only one VNF of the accepted service requests.

In the following example, we explain how service request s_4 is processed. Initially s_4 is assigned to VM_4 and it needs two time steps to be processed, thus $x_{4,1}^1(1) = 1$ and $x_{4,1}^1(2) = 1$. In the third time step, the third VNF of s_4 is assigned to VM_5, so $x_{4,1}^1(3) = 1$. However, either because of the second VNF of s_1 is being processed in VM_5 or because of the transmission process, s_4 has to wait to be processed in the fourth time step. In the fifth time step, the fifth VNF of s_4 is assigned and starts to be processed on VM_3, where the s_5 is just assigned to VM_3 at the same time but it has to wait in the queue until s_4 finishes its processing. The sixth VNF of s_4 is assigned to process on VM_1 and the same continues in the seventh, eighth and ninth time steps, respectively. In the tenth and eleventh time steps, the sixth VNF of s_4 is assigned to process on VM_2. It takes 11 time steps to finish processing the service request s_4.

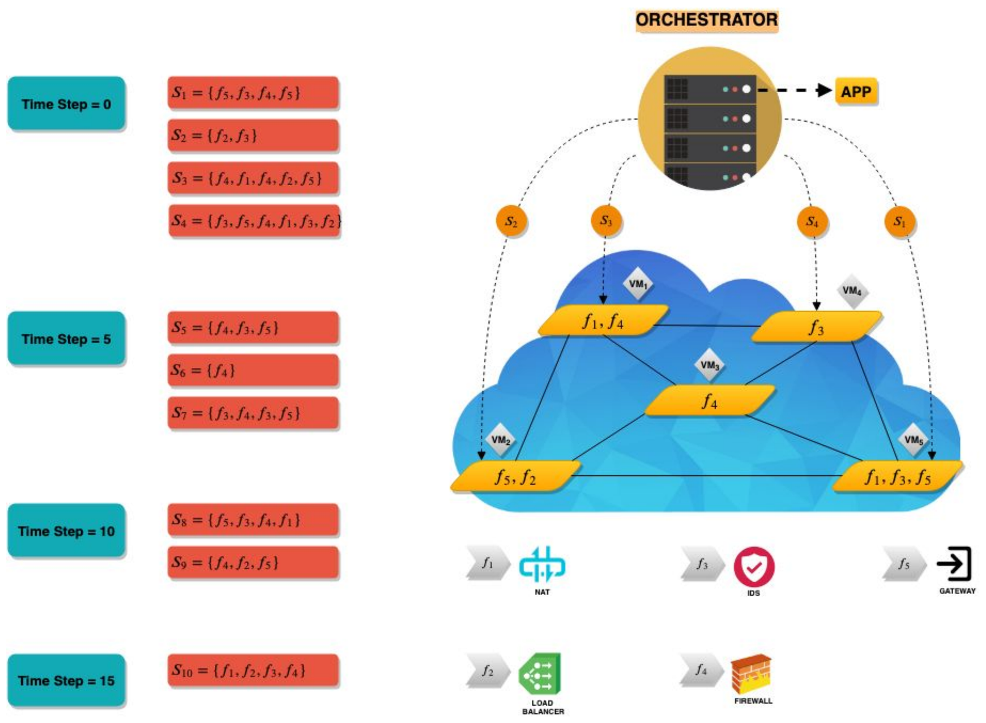


Figure 6.1: one of the possible mappings for the arriving service requests

Table 6.1: Assignment Schedule for Example 7.2.5: The values of variable x which are equal to one.

t	s_1	s_2	s_3	s_4	s_5
1	$x^3_{1,1}(1)$	$x^4_{2,1}(1)$	$x^2_{3,1}(1)$	$x^1_{4,1}(1)$	$x^4_{5,1}(1)$
2	$x^3_{1,1}(2)$	$x^2_{2,2}(2)$	$x^4_{3,2}(2)$	$x^3_{4,2}(2)$	$x^4_{5,1}(2)$
3	$x^3_{1,2}(3)$	$x^2_{2,2}(3)$	$x^4_{3,2}(3)$	$x^3_{4,2}(3)$	$x^4_{5,2}(3)$
4	$x^1_{1,3}(4)$	$x^2_{2,3}(4)$	$x^4_{3,2}(4)$	$x^3_{4,2}(4)$	$x^1_{5,3}(4)$
5	$x^1_{1,3}(5)$	$x^3_{2,4}(5)$	$x^4_{3,2}(5)$	$x^3_{4,2}(5)$	$x^2_{5,4}(5)$
6		$x^3_{2,4}(6)$	$x^2_{3,3}(6)$	$x^3_{4,2}(6)$	$x^2_{5,5}(6)$
7		$x^3_{2,4}(7)$	$x^2_{3,3}(7)$	$x^4_{4,3}(7)$	$x^2_{5,5}(7)$
8		$x^2_{2,5}(8)$	$x^2_{3,3}(8)$	$x^4_{4,4}(8)$	$x^1_{5,6}(8)$
9		$x^2_{2,5}(9)$	$x^3_{3,4}(9)$	$x^3_{4,5}(9)$	$x^1_{5,7}(9)$
10		$x^2_{2,6}(10)$		$x^3_{4,5}(10)$	
11		$x^1_{2,7}(11)$		$x^3_{4,5}(11)$	
12		$x^1_{2,7}(12)$			

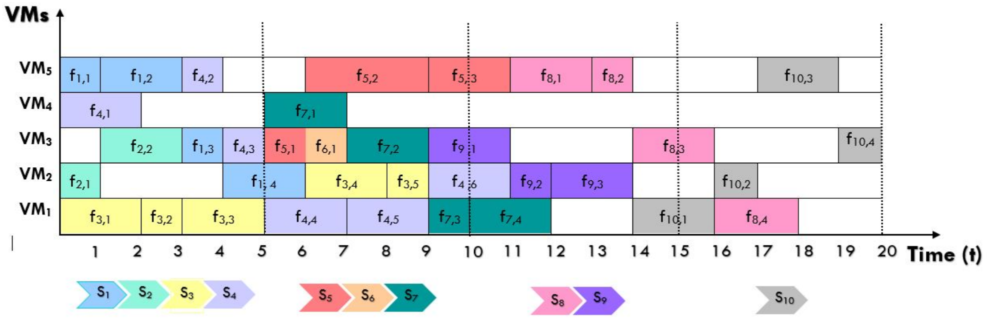


Figure 6.2: one of the possible scheduling for all VNFs of the arriving service requests

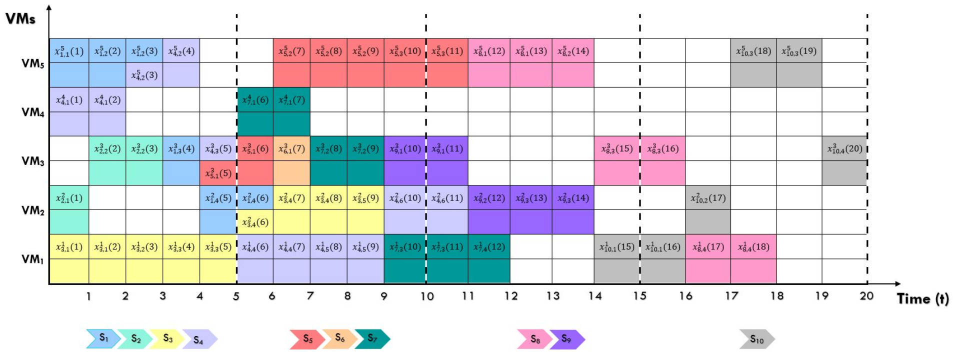


Figure 6.3: All possible values of variable x that reflect the assignment schedule for all incoming service requests

Variable/Time	1	2	3	4	5	6	7
$x^{k}_{i,j}(t)$	1	1	0	0	0	0	0
$x^{l}_{i,j+1}(t)$	0	0	1	1	1	1	1
$\sigma(R^{k,l}_{i,j}(t))$	0	0	1	1 Transmission	1	0	0
$1^{st} term$	0	0	$y^{k,l}_{i,j}(t)$	0	0	0	0
$2^{nd} term$	0	0	0	$y^{k,l}_{i,j}(t)$	$y^{k,l}_{i,j}(t)$	0	0

Figure 6.4: Example of the cost function of objective 2

6.3 Algorithm Design

Consider the problem formulation mentioned in the previous section, four main objectives are proposed in this chapter to be achieved. Particularly, maximizing the acceptance rate (6.1), optimizing link utilization (6.4), minimizing the overall processing (6.6) and minimizing the relative processing time 6.7.

Formally, the whole multi-objective optimization problem can be written as follows:

$$
\begin{aligned}
&\min_{x}\ [-Z_1(x), Z_2(x), Z_3(x), Z_4(x)]\\
&st.\ x^{l}_{i,j+1}(t+1) = 0 \quad \text{if}\\
&\quad A^{k}_{i,j}(t) = 0 \text{ or } x^{l}_{i,j+1}(t) = 0,\ \text{for } k,l = 1,\dots,|\mathscr{V}|\\
&\quad R^{k,l}_{i,j}(t) = 0 \text{ if } x^{l}_{i,j+1}(t) = 0,\ \forall (k,l) \in \mathscr{E}\\
&\quad \textstyle\sum_{i'=1}^{|\mathscr{S}|}\sum_{j'=1}^{|s_i|} R^{k,l}_{i',j'}(t) \le lc_{k,l}, \quad \forall (k,l) \in \mathscr{E}\\
&\quad \sum_{j=1}^{|s_i|}\sum_{k=1}^{|\mathscr{V}|}\sum_{t=1}^{T} x^{k}_{i,j}(t) \le E_i, \quad \text{for} \quad i = 1,\dots,|\mathscr{S}|.
\end{aligned}
\tag{6.12}
$$

where $A^{k}_{i,j}(t) = x^{k}_{i,j}(t)\cdot(1 - x^{k}_{i,j}(t+1))$, $i = 1,\dots,|\mathscr{S}|$, $j = 1,\dots,|s_i|-1$, and the cost functions $Z_1(x)$, $Z_2(x)$ and $Z_3(x)$ are as defined in (6.1), (6.4), (6.6) and (6.7), respectively.

Considering this problem formulation, three main approaches based on GA can be applied to find the near to optimal solution for the problem, namely, pareto dominance-based algorithms (i.e., NSGA-II [8]), indicator-based algorithms and decomposition-based algorithms (i.e., MOEA/D) explained in detail in chapter 3. This section accordingly adopts and presents novel changes in dealing with the two algorithms (NSGA-II) and (CMOEA/D-DE-ATP [101]) to solve the proposed problem. The two new algorithms called and follow the framework explained in Alg. 2. Additionally, both proposed algorithms contains the standard procedures of genetic algorithms initial population, selection, crossover and mutation.

6.3.1 Representation

The candidate solution (i.e., a set of VMs in this problem) is encoded into a chromosome, the chromosome is a set of genes that may be encoded in different ways. The encoding of the problem depends on the problem definition. Several encoding methods are used such as floating encoding, binary encoding and symbolic encoding, however, both of our proposed algorithms use a binary encoding method. Each chromosome has sub-strings length equal to VNF numbers and each VM $k \in |\mathcal{V}|$ composed of a sub-string of the VNF type $\Phi_k = (x_k)$ (VNF can be supported by this VM). While the length of the sub-string is fixed, the length of the chromosome is variable depending on the number of sub-strings (VMs).

6.3.2 Initial population

The initial population generates the sub-string in a chromosome randomly, the number of sub-strings for a position of available VMs is picked randomly from one of possible index $[1, 2, ..., |\mathcal{V}|]$. Each chromosome is represents by and defines $x_{i,j}^{k}(t)$ the possible mapping and scheduling for a set of incoming service requests to be processed on available VMs. Additionally, each chromosome is adjusted to validate the problem constraints (6.8), (6.9), (6.10) and (6.11) and therefore provide a set of feasible solutions that will represent the initial solution.

6.3.3 Selection Operator

Selection is the first operator implemented on the population after the initial population is generated. Selection operator selects good strings in a population and formulates a mating pool. In selection operation, the method of normal selection causes that individual that encodes successful structures to create copies more regularly. The chance that a chromosome will be chosen is proportional to its fitness.

At each generation, the method selects the best chromosome based on its fitness value (objectives values) for creating a new offspring. In this study, we used the tournament selection as a selection operator for the proposed algorithms; it selects some of the individuals randomly from the population and copies the best individual from this group into the population medium used for selection approaches in evolutionary algorithms. In other words, the selection operator selects the best two available VMs to process the j-th VNF of the incoming service request s_i at time step t as parents.

6.3.4 crossover

The outputs from the selection procedure are recombined to a new offspring chromosome using the crossover operator; the operator checks if the new offspring gets the characteristics better than the parents, thus the new combination is fitter than its parents to be forwarded to the mutation operator. In this study, SM-NSGA-II algorithm applies a binary crossover (SBX) proposed in [8] and SM-MOEA/D algorithm applies a DE operator as a crossover operator presented in [100].

6.3.5 Mutation

After crossover operator is implemented, the mutation operator is performed to the individual solution where a gene is changed randomly by a small probability to produce a new chromosome. In this study, the mutation operator rate of the SM-NSGA-II algorithm is p_m= 1/ (no.of variables) while, the mutation operator rate of the SM-MOEA/D algorithm is p_m=1/(parameter dimension), i.e., the probability for each bit to be flipped is pm.

6.3.6 Stopping Criteria

The stopping criterion of the SM-NSGA-II algorithm is the maximum number of the evaluations set to 25000 in this simulation. The stopping criteria of the SM-MOEA/D algorithm is either the maximum number of iterations set to 300 iterations with 100 population size (the total number of repetition is 300*100) in our simulation or if there is no further enhancement obtained for the values of the objective functions after a defined upper limit for the number of iteration (100 iterations in this simulation).

6.3.7 Output

After sorting the output individuals depending on the objectives values (6.1), (6.4), (6.6), (6.7), the population with the best performance will be selected as a final result. This result has to achieve the problem constraints (6.8), (6.9), (6.10) and (6.11). If any of these constraints are not achieved, the algorithm removes the solution and selects another solution, which can meet the problem constraints.

The output of both algorithms is the best feasible solution for mapping and schedule for the VNFs of the arrival service requests. In other words, the output is the best value for the variable $x_{ij}^{k}(t)$ for $i = 1, ..., |\mathscr{S}|$, $j = 1, ..., |s_i|$ and $k = 1, ..., |\mathcal{V}|$ along with the corresponding objective values achieved by this variable x.

6.3.8 Complexity Analysis

The complexity for both algorithms consists of three parts: the calculation of the objective functions, the calculation of the constraints and the calculation of the genetic operators.

For the first part, the complexity of calculating the objective functions Z_1, Z_2, Z_3 and Z_4 are $\mathcal{O}(|\mathscr{S}||\mathcal{V}|T)$, $\mathcal{O}\left(Q|\mathcal{V}|^2\right)$, $\mathcal{O}(Q|\mathcal{V}|T)$ and $\mathcal{O}(|\mathscr{S}||\mathcal{V}|T)$, respectively, with $Q = \sum_{i=1}^{|\mathscr{S}|} s_i$.

For the second part, the complexity of calculating the proposed problem constraints (6.8) (6.9) (6.10) (6.11) can be calculated as follows $\mathcal{O}\left(Q|\mathcal{V}|^2\right)$, where $Q = \sum_{i=1}^{|\mathscr{S}|} s_i$.

For the last part, the complexity of the GA parameters (i.e., Iteration number, population size) and operators (i.e., the tournament selection, crossover, and mutation) vary based on the problem specification; this complexity is out of the scope of this study, more details about this complexity are available on [112].

Algorithm 3 RSAMOAD and RA-NSGA-II

```
1: Input:
     • Network parameters:
       $\mathscr{S}, E, W, \mathcal{V}, \mathscr{E}, F, VF, \{tc_{k,l} | (k,l) \in \mathscr{E}\}$
2: Output:
     • X = $\{x_{ij}^{k}(t) | i = 1,...,|\mathscr{S}|, j = 1,...,|s_i|, k \in \mathcal{V}, t = 1,...,T\}$ along with the values of
       their corresponding objective functions.
3: begin:
     • Generate the initial population $X^1, \ldots, X^N$ uniformly in random.
     • for every $X^i = X^1,...,X^N$ do
     •     if $X^i$ satisfying all the proposed constraints then
     • Evaluate the proposed cost functions $Z_1(x)$, $Z_2(x)$, $Z_3(x)$ according to (6.1), (6.4),
       (6.6)
     • applying the genetic (i.e., representation, selection, crossover and mutation)
       operators to the result
     • Choosing the best Feasible Solution after applying the previous parameters
     •     end if
     • end for
4:   Selecting the best result from all populations.
5:   Termination criterion:
     • The Algorithm stopping criterion.
     •     if the stopping criterion is satisfied then,
       Stop and return the final output as the best mapping and scheduling.
```

6.4 Experiment Result

To evaluate the performance of the (SM-MOEA/D and SM-NSGA-II) algorithms, we conducted extensive simulations considering the VNF service chains network proposed in [89]. It is assumed that every virtual link has a fixed bandwidth equal to 2 Mbps. Furthermore, the VM bandwidth is equal to 8 Mbps and the bandwidth demand ($d_{i,j}$) of every service request s_i after processing its j-th VN ranges from 30 Kbps and 2030 Kbps.

6.4.1 Simulation set up

The number of the incoming service requests varies from 100 to 400, each of which requests contains a VNF chain composed of at most 5 VNF different chains. We scale the number of VMs from 10 to 40 VMs, every VM can support at most 3 VNFs of the total number of VNFs which is 5. Theses VNFs have to include some of the common deployed VNFs such as Firewall (FW), Detection System (IDS), Network Address Translator (NAT), Intrusion, WAN Optimizer, Flow Monitor (FM) and Load Balancer (LB).

The simulations for both algorithms are implemented using Java and MATLAB software and solved on a machine with Intel Xeon Gold 6150 2.7GHz, 128 GB of RAM and Linux operating system.

6.4.2 Evaluation Scenarios

We adopt the scale of the network topologies, simulation set up and network parameters proposed in [89, 109]. One unit of VM capacity indicates the ability to serve one VNF chain of the service request per time, precisely

Two different network scenarios (single objective and multi objectives) and four different network instances are provided in this section to provide insight into the performance of both algorithms. The four different network instances consist of 100, 200, 300 and 400 service requests along with 10, 20, 30 and 40 VMs, respectively. The first network scenario considers every proposed objective (first, second, third and fourth) individually. The second scenario considers all four objectives functions simultaneously.

6.4.2.1 Scenario 1

We applied all the network instances small, medium, large and extra-large along with every objective independently. The results of the first scenario are represented in Fig. 6.5, Fig. 6.6, Fig. 6.7 and Fig. 6.8. We can clearly see that SM-MOEA/D algorithm slightly outperforms SM-NSGA-II algorithm in the four types of network instances, however, we observe that these two algorithms are comparable. Overall, the figures prove that both proposed algorithms are comparable and allow exploring nearly to the optimal solution for every proposed objective individually, any possible combinations of the proposed objectives (if needed) or for all proposed four objectives simultaneously.

6.4.2.2 Scenario 2

The near to optimal results obtained by applying our proposed algorithms for all objective function values are collected in Table 6.2. It is clear that SM-MOEA/D algorithm is better than SM-NSGA-II algorithm in terms of obtaining the near-to-optimal solution for all the objective functions. However, the CPU computational time of SM-MOEA/D algorithm is higher than SM-NSGA-II. It is obvious that when the number of incoming service requests increases, the problem becomes more complicated and takes more computational time to be solved.

6.5 Conclusion

This chapter formulates a mathematical model to find the best mapping and scheduling for a set of incoming service requests to be processed in order through VMs/servers. The model considers four different objectives, in particular, maximizing the acceptance rate, optimizing link utilization, minimizing the overall processing time and minimize the relative processing time, while the model has to satisfy the system constraints,

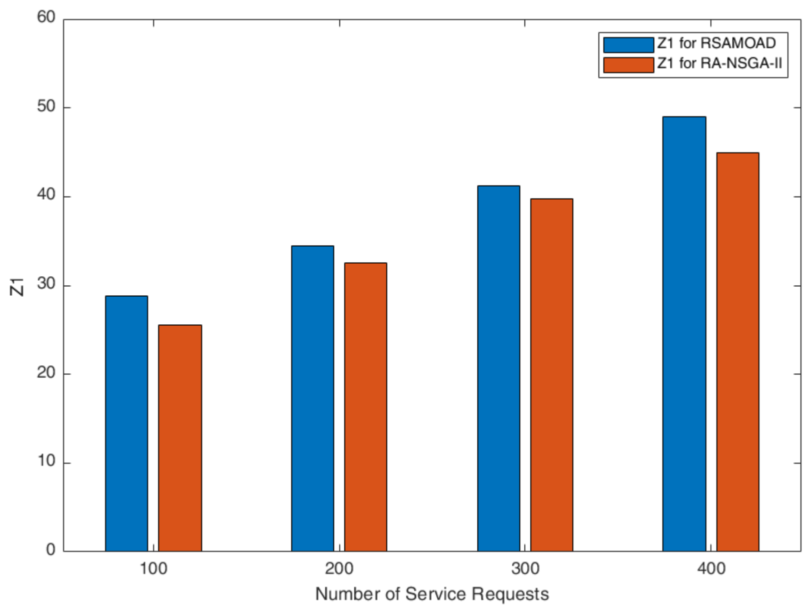


Figure 6.5: The optimal solution achieved by both proposed algorithms for the first objective value via all network instances

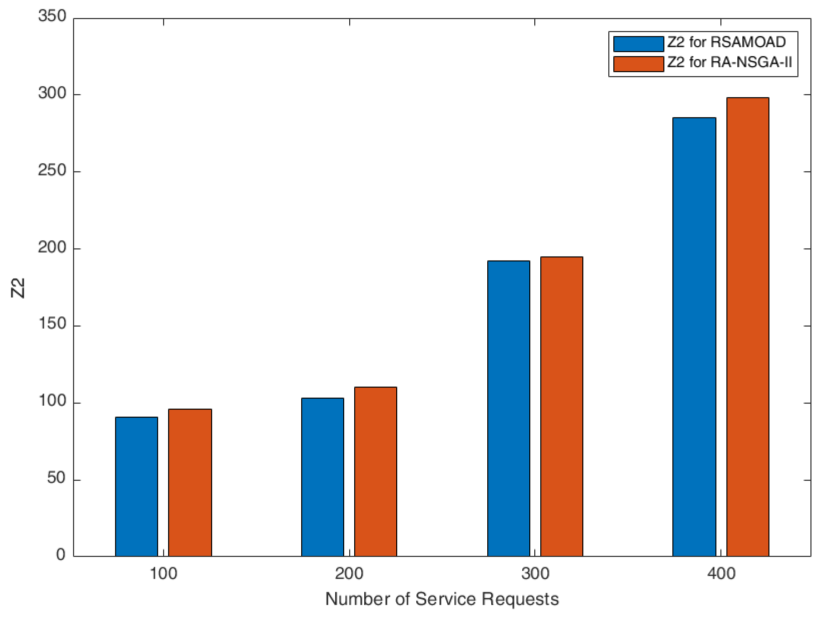


Figure 6.6: The optimal solution achieved by both proposed algorithms for the second objective value via all network instances

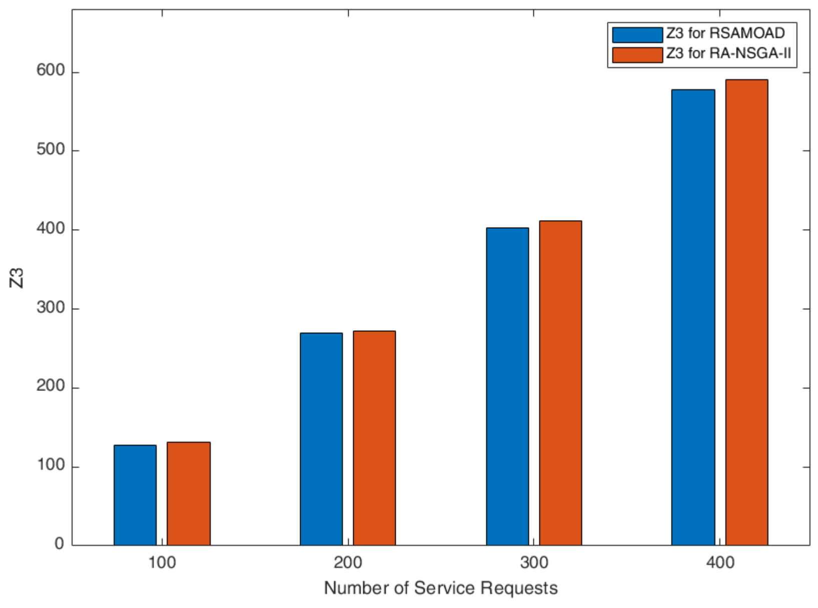


Figure 6.7: The optimal solution achieved by both proposed algorithms for the third objective value via all network instances

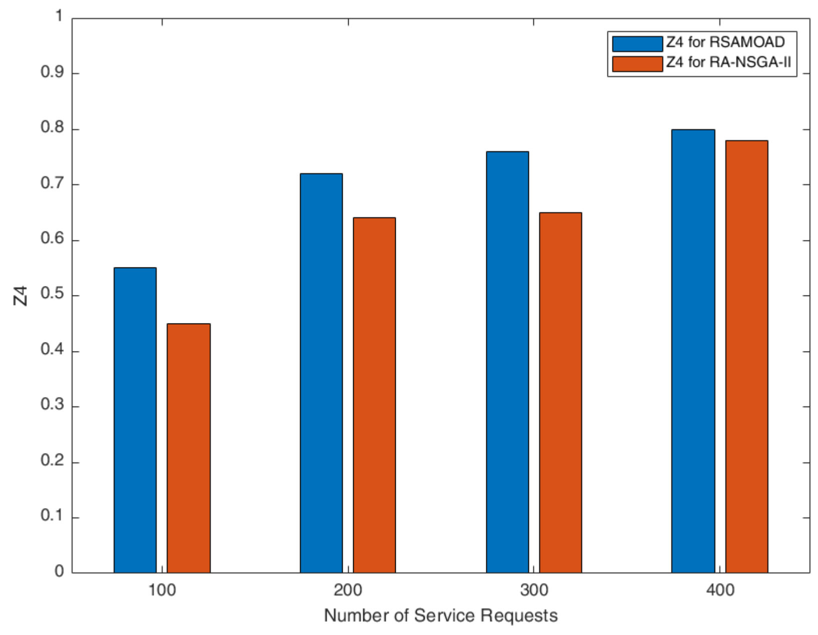


Figure 6.8: The optimal solution achieved by both proposed algorithms for the fourth objective value via all network instances

Table 6.2: First, Second, Third and Fourth objective values for both tested algorithms

Alg.	Network Instance	Z_1	Z_2	Z_3	Z_4
RSAMOEAD	S	51	15	395	0.5
	M	104	61	798	0.66
	L	156	134	1198	0.76
	XL	208	283	1600	0.79
RA-NSGA-II	S	47	19	398	0.42
	M	97	77	800	0.59
	L	140	198	1199	0.68
	XL	186	506	1600	0.65

particularly, VNF order link, traffic constraint, link capacity constraint and expiry time constraint. Two heuristic algorithms are proposed, namely, SM-MOEA/D and SM-NSGA-II to find the near to optimal solution for the proposed problem.

CHAPTER 7

Mapping and Scheduling of Virtual Network Functions using Multi Objective Optimization Algorithm

7.1 Introduction

Within the context of SDN, the problem of resource allocation for a set of incoming VNF service requests has been the focus of many studies. As mentioned previously in chapter 2, given great flexibility provided by combining the NFV and SDN concepts, this chapter will study the difficulty of the placement and scheduling for a set of incoming requests since it arrived to process and transfer between multiple VM nodes running in the same network. The topology of the network and the links among VMs are assumed to be time invariant in this model.

By merging SDN and NFV technologies, the future internet will be a virtualized environment in which the service providers can manage and customize the deployment of the VNFs into the NFV environment. SFC is an order list of network function instances which need to be processed through VMs/servers according to the network policy. Therefore, using the SFC deployment model with NFV and SDN technologies will add more control of the traffic flows for the connected services and manage the network set up.

Being able to determine the best mapping of the incoming service requests could allow operators to implement more of the arrival service requests according to the admission policies which will maximize the overall network revenues. Moreover, the ability to find the best scheduling for the VNFs service chains of the incoming service requests to be processed on the available VMs could allow operators to implement VNFs faster according to their priority which will minimize the execution time and the overall cost.

A new optimization model has been developed to find the near to optimal mapping and scheduling for the incoming VNF service requests in this chapter. This model aims to achieve three objectives functions, namely, minimizing the transmission delays occurring in every link, minimizing the processing capacity for every VM and minimizing the processing delay at every VM under traffic flow, VM resource, processing start and transmission start constraints. Solving the resulting problem by proposing an evolutionary algorithm, the lowest delay multi-objective evolutionary algorithm based on decomposition algorithm (LDMOAD). Simulation results illustrate that the resulting algorithm is scalable while considering delay and it outperforms the genetic bandwidth link allocation (GA-BA) and genetic non-bandwidth link allocation (GA-NBA) algorithms.

The main advantage of using the decomposition approach to solve the proposed problem in this chapter is to have a good scalability and computational efficiency such that all above mentioned objectives and constraints are met by using the updated data received from the orchestrator. The remainder of this chapter is as below. In Section 7.2, the network model and the mathematical formulation for the VNF scheduling and mapping problem are provided. The MOAD-DE algorithm as the proposed solution for the resulting multi-objective optimization problem is proposed in Section 7.3. Section 7.4 discusses the simulation results and Section 7.5 concludes the study of the this chapter.

7.2 NFV Network Model

In this section, the problem of mapping and scheduling for a set of incoming service requests to process through different VMs is investigated. However, the scheduling of VNFs into physical nodes is out of the scope of this chapter. Consider the network model represented by undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ where $\mathscr{V} = \{v_k | k = 1, ..., |\mathscr{V}|\}$ is the set of VMs nodes and $\mathscr{E} = \{(k,l) | k,l = 1,...,|\mathscr{V}|, k \neq l\}$ is the set of virtual links between VMs. $F = \{f_1, f_2, ..., f_{|F|}\}$ is the set of unique VNFs supported in the cloud. Each VM supports a subset of VNFs denoted by VF_k for $k = 1, ..., |\mathscr{V}|$. The sequence of network service requests is denoted by $\mathscr{S} = [s_1, ..., s_{|\mathscr{S}|}]$. Each request $s_i \in \mathscr{S}$ for $i = 1, ..., |\mathscr{S}|$ contains a sequence of VNFs denoted by $f_{i,j}$ for $i = 1, ..., |\mathscr{S}|$ and $j = 1, ..., |s_i|$, where $f_{i,j} \in F$. $f_{i,j}$ is the j-th VNF of the i-th incoming service request (s_i). In this model, we assume that every service request s_i for $i = 1, 2, ..., |\mathscr{S}|$ has different number of VNFs. Let c_k for $k = 1, 2, ..., |\mathscr{V}|$ be the processing capacity of k-th VM. This capacity can be defined based on memory and computing capabilities of physical machine that run each VM. For every VL a transmission capacity can be define as $tc_{k,l}$ for $k = 1, 2, ..., |\mathscr{V}|$ and $l = 1, 2, ..., |\mathscr{V}|$. Let $d_{i,j}$ for $i = 1, 2, .., |\mathscr{S}|$ and $j = 1, 2, .., |s_i|$ be the bandwidth demand of processing the j-th VNF of the i-th arrived service request. And $R_{i,j}^{k,l}$ for $i = 1, 2, .., |\mathscr{S}|$, $j = 1, 2, .., |_i|$, $k = 1, 2, .., |\mathscr{V}|$ and $l = 1, 2, .., |\mathscr{V}|$ be the traffic flow of the j-th VNF of the service request s_i passing through (k,l)-th link.

7.2.1 Minimizing the transmission delay

One of the main objectives in this chapter is to minimize the transmission delay experienced by the traffic flow of the arrived request s_i using the corresponding link (k,l) (where $k,l \in |\mathscr{V}|$ and $k \neq l$) to transfer between VMs. Let $d_{i,j}$ for $i = 1,2,..,|\mathscr{S}|$ and $j = 1,2,..,|s_i|$ be the amount of data (in bits or packets) needed to process the j-th VNF of the i-th arrived service request. The transmission delay of link (k,l) link corresponding to i-th service request can be evaluated by $\frac{d_{i,j}}{tc_{k,l}-R^{k,l}_{i,j}}$. We define variable $\beta_{i,j}$ for $i = 1,2,...,|\mathscr{S}|$ and $j = 1,...,|s_i|-1$ as the starting time of the transmission between the VMs performing the $f_{i,j}$ and the $f_{i,j+1}$, respectively. Note that $f_{i,j}$ is the j-th VNF of the i-th incoming service request s_i. Then, the mathematical formulation for this objective can be described as follows:

$$MinZ_1(X) = \sum_{i=1}^{|\mathscr{S}|} \sum_{j=1}^{|s_i|-1} \left(\beta_{i,j} + \sum_{k=1}^{|\mathscr{V}|} x^k_{i,j} \cdot \sum_{l=1}^{|\mathscr{V}|} \frac{d_{i,j}}{tc_{k,l} - R^{k,l}_{i,j}} \right) \tag{7.1}$$

$$x^k_{i,j} = \begin{cases} 1 & \text{If the } J\text{-th VNF of request} \\ & s_i \text{ can process on VM node } k \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

7.2.2 Minimize the processing capacity

This objective adopts the idea of fair distribution for processing capacity to process all incoming network service requests s_i according to their priorities. Then, the principle goal of this objective is to find the correct VM to process the j-th VNF of the i-th service request s_i according to the current status of the VM processing capacity and bandwidth demand $d_{i,j}$ of the incoming service requests. Consequently, this process will maximize the sufficient capacity for every VM which will maximize the system sufficiently. Formally, this objective can be described as follows.

$$Max\ Z_2(X) = \sum_{k=1}^{|\mathcal{V}|} c_k - \sum_{i=1}^{|S|}\sum_{j=1}^{|s_i|} d_{i,j} \cdot x_{i,j}^{k} \tag{7.3}$$

7.2.3 Minimizing the processing delay

When a sequence of incoming service requests are sent to be processed on VMs, a potential delay may occur during the processing operation. The principal advantage of this objective is to measure the quality of the system by minimizing the processing delay for every service request. This advantage enables the VM to process the incoming service requests quickly. Let us define the starting processing time for j-th VNF of service request s_i as α_{ij}. The mathematical formulation for this objective can be described as follows.

$$MinZ_3(X) = \sum_{i=1}^{|S|}\sum_{j=1}^{|s_i|} \alpha_{i,j} + \sum_{k=1}^{|\mathcal{V}|} x_{i,j}^{k} \cdot q_{i,j}^{k} \tag{7.4}$$

Where $q_{i,j}^{k}$ is a processing delay occurred for the j-th VNF of service request s_i while its processing on the k-th VM.

7.2.4 Constraints

In this model, a set of network constraints are set to meet a feasible solution of the proposed algorithm. The system constraints are described as follows.

(1) The traffic flow $R_{i,j}^{k,l}$ for the j-th VNF of the incoming service request s_i for $i = 1,2,...,|\mathcal{S}|$ passing through the (k,l)-th link should not exceed the link capacity $tc_{k,l}$. Formally, $R_{i,j}^{k,l}(t)$

$$\sum_{k}\sum_{l} x_{i,j}^{k} \cdot R_{i,j}^{k,l} < tc_{k,l} \tag{7.5}$$

(2) There is at most one VM node selected to serve the j-th VNF of service request s_i for $j = 1,2,...,|s_i|$ and for $i = 1,2,...,|\mathscr{S}|$. Formally,

$$\sum_{k} x_{ij}^{k} \leq 1 \tag{7.6}$$

(3) The j-th VNF function of service s_i for $i = 1,2,...,|\mathscr{S}|$ must start to process on k-th VM after the previous VNF function $j-1^{th}$ of the same service request s_i finish its processing. This constraint ensures that all VNFs of the same service requests should be processed in order. Formally,

$$\sum_{i=1}^{|\mathscr{S}|}\sum_{j=2}^{|s_i|}\left(\beta_{i,j-1} + \sum_{k=1}^{|\mathscr{V}|} x_{i,j-1}^{k} \cdot \sum_{l=1}^{|\mathscr{V}|} \frac{d_{i,j}}{tc_{k,l} - R_{i,j}^{k,l}}\right) \leq \alpha_{ij} \tag{7.7}$$

(4) The traffic of the j-th VNF for the same service request s_i after finishing its processing on k-th VM should transfer from this k-th VM to the next l-th VM in order. This constraints ensures that $j+1$-th VNF should not start to transmit to the next VM except if the j-th VNF of request s_i complete their processing on k-th VM. Formally,

$$\sum_{i=1}^{|\mathscr{S}|}\sum_{j=1}^{|s_i|-1} \alpha_{i,j} + \sum_{k=1}^{|\mathscr{V}|} x_{i,j}^{k} \cdot q_{i,j}^{k} \leq \beta_{i,j} \tag{7.8}$$

7.2.5 Example of Mapping and Scheduling for a set of incoming service requests

A clear example of the proposed problem is shown in Figures 7.1 and 7.2. Consider the network structure consists of 7 VMs, and every VM supports different VNF functions on it (eg. 1. Firewall, 2. Video Transcoder, 3. Proxy 4. Nat, 5. VPN and 6. Load Balancing). A set of incoming service requests $s_1,...,s_4$ arrives at time $t = 0$ asking to be processed in these VMs as shown in Fig. 7.1. Every service request contains a subset of VNFs in different order $s_1 = \{f_6, f_3, f_4, f_5, f_1\}$, $s_2 = \{f_4, f_2, f_3, f_5, f_1, f_2, f_5\}$, $s_3 = \{f_2, f_1, f_5\}$, $s_4 = \{f_6, f_3\}$. If these VNFs distribute to process on VMs randomly, then some of the VMs will be busy by processing these VNFs and others will not. For instance, the first VNF of s_1 (i.e., f_6) is

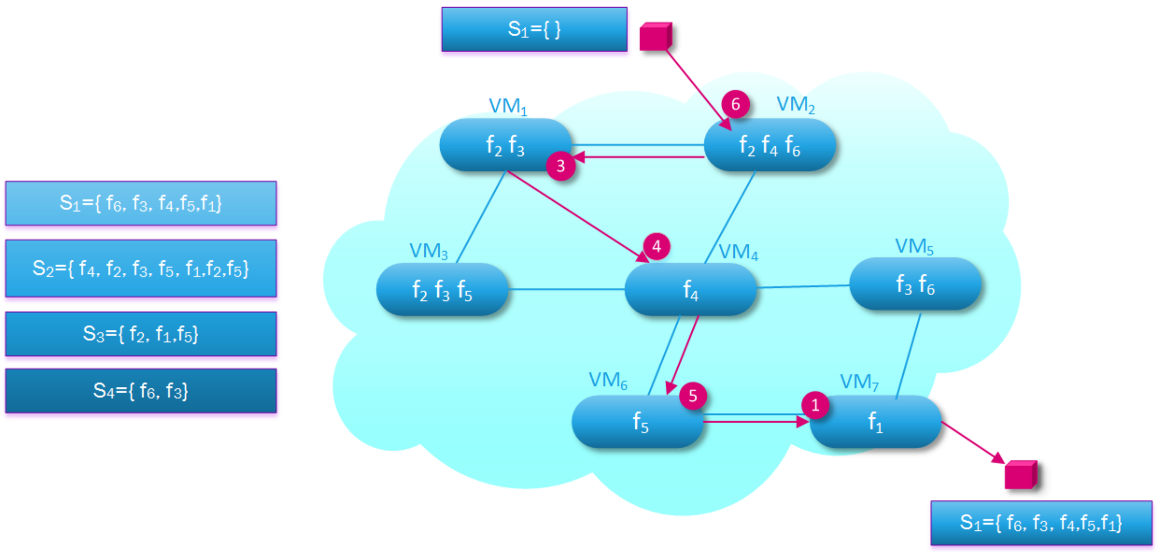


Figure 7.1: Network structure and one of the possible mappings for s_1

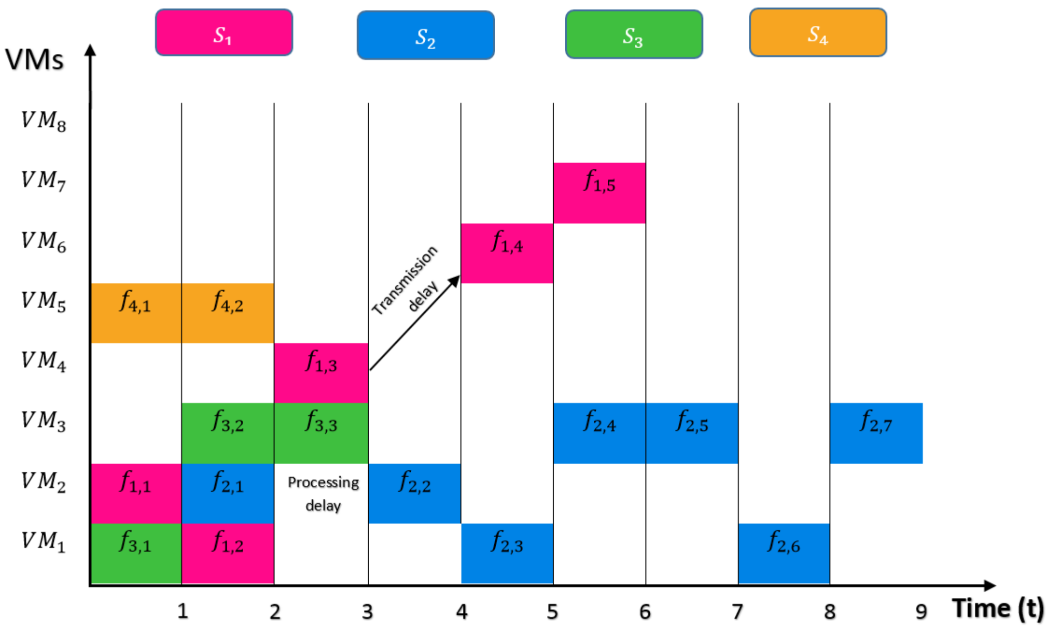


Figure 7.2: Illustration of the possible mappings and schedules for the incoming service requests

processed in VM2, then the second, third, fourth and fifth VNFs of s_1 (i.e., f_3, f_4, f_5, f_1) are processed in VM1,VM4,VM6,VM7, respectively, as shown in Fig. 7.1. It is assumed that each VM can process one VNF at a time. During the transmission of these VNFs from one VM to another, a transmission delay may occur for some of these VNFs as shown in Fig. 7.2. The first VNF of s_2 (i.e., f_3) arrives to the cloud at the same time $t = 0$, this VNF can be processed in VM2 or VM4. In the case of assigning this function (f_3) to VM2 as shown in Fig. 7.2, it has to wait one time slot to be processed (as the VM2 is still processing the first VNF of s_1). However, the function can be assigned to VM4 to avoid this delay. Furthermore, during the processing of these VNFs on VMs, a processing delay can occur for these VNFs as shown in Fig. 7.2, the second VNF f_2 of s_2 is assigned to VM_2 but it has to wait one time slots to start its processing. To optimize all mentioned problems in this example, the first (7.1), second (7.3) and third objectives (7.4) are provided in this study.

7.3 Proposed Algorithm

We use decomposition approaches explained previously in chapter 3 to solve the multi-objective optimization problem proposed in this chapter. It decomposes the problem and solves it as multiple single objective problems. Authors in [101] provided a framework of CMOEA/D-DE-ATP for solving constrained optimization problems. we found CMOEA/D-DE-ATP is the best algorithm which can be used to solve the proposed problem (as single and multi-objective) in this chapter.

Hence, we proposed LDMOAD/DE algorithm based on CMOEA/D-DE-ATP algorithm, Fig. 7.3 shows the flow chart of the LDMOAD/DE algorithm procedures that consist of the following steps. Firstly we initialize the network parameters, number of incoming service requests ($\mathscr{S}$) and the weight W for every service request, VNFs number of each request, VMs number ($\mathscr{V}$), c_k is a processing capacity for every node, set of VNFs supported by

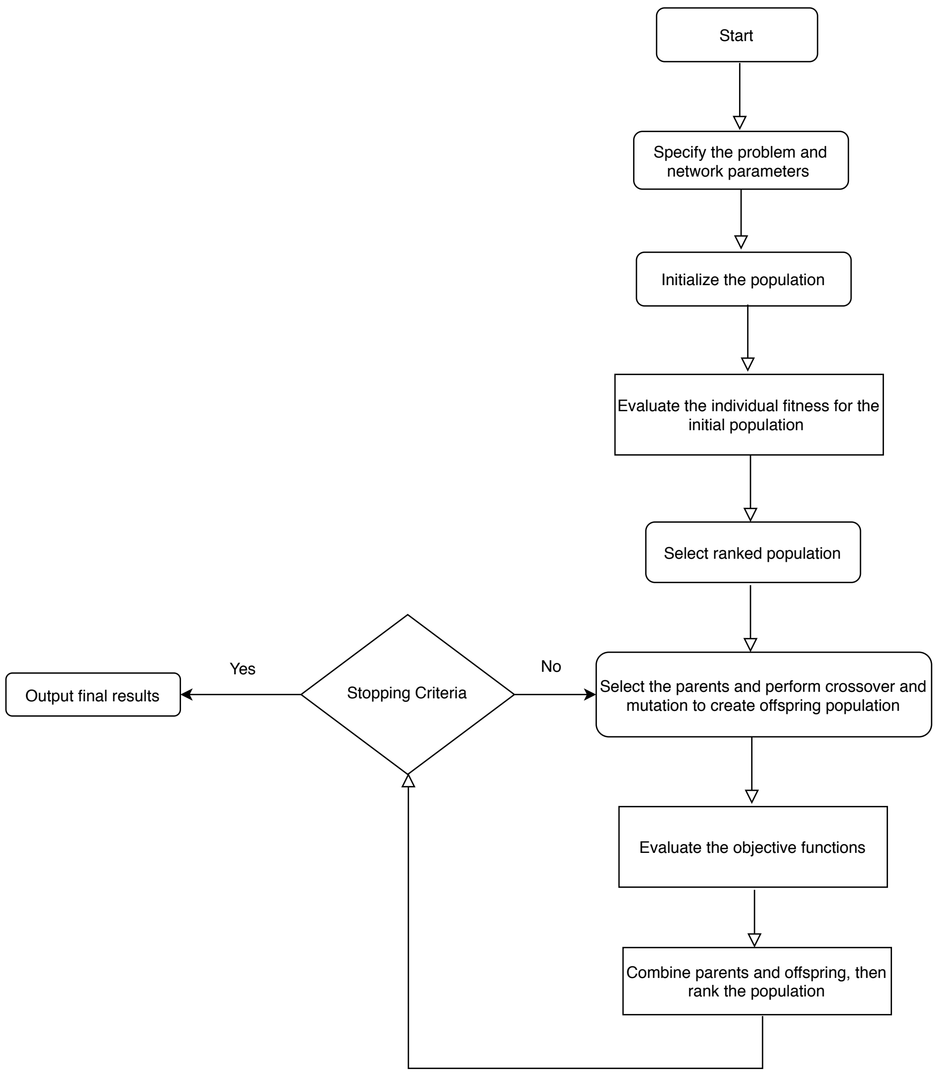

Start
Specify the problem and network parameters
Initialize the population
Evaluate the individual fitness for the initial population
Select ranked population
Select the parents and perform crossover and mutation to create offspring population
Evaluate the objective functions
Combine parents and offspring, then rank the population
Stopping Criteria
No
Yes
Output final results

Figure 7.3: The flow chart of the proposed algorithms procedures

each VM, number of virtual links between VMs ($\mathscr{E}$), the transmission capacity for each of these links $\{tc_{k,l}|(k,l) \in \mathscr{E}\}$.

The stopping criteria has been determined in the LDMOAD/DE algorithm depends on the number of population size (set to 100) and the number of iterations (set to 300), so the total number of repetition is 300*100. Moving to step 4, the proposed algorithm uses binary values as representation operator to represent the main network variable x_{ijk} which describes if the VNF function of request s_i can process on node k. The selection operator which is used in this algorithm mating selection. The proposed algorithm uses DE operator presented in [113]. Finally, the proposed algorithm set the rate equal to $p_m = 1/(Parametersdimension)$.

The algorithm will check the feasibility for output solution by checking all constraints proposed in subsection 7.2.4. The final output will be the near to optimal mapping and scheduling for the incoming requests.

Two main process should be taken into consideration while we calculate the complexity of the proposed algorithm. First LDMOAD algorithm, the algorithm should map the incoming requests to the available VMs. This operation requires $\mathcal{O}\left(Q|\mathcal{S}||\mathcal{V}|^2\right)$ where $Q = \sum_{i=1}^{|\mathcal{S}|} s_i$ computation as shown in Alg. 4. Second GA algorithm, the complexity of the GA algorithm is changeable and depends on the network parameters (e.g population size, number of objectives).

7.4 NUMERICAL RESULTS

The network structure in this chapter contains four network cases: small, medium, large and extra-large. The small network covers 100 service requests (with service request demand distributed frequently from 1 to 100 Kb, respectively) and 10 VM nodes. The medium network contains 200 service requests (with capacity distributed regularly from 1 to 200 Kb, respectively) and 20 VM nodes. The large networks consist of 300 network requests (the capacity spread frequently from 1 to 300 Kb) and 30 VM nodes. Finally, the extra-large network is composed of 400 service requests (the size is spread regularly from 1 to 1000 Kb) and 40 VM nodes. The VM node processing capacities in all network structures set 1000 Mb (except for the extra-large network set 2000 Mb).

The simulation results in this study are near-to-optimal scenarios proposed in [109]. The simulations are implemented using the MATLAB program on Intel Xeon Gold 6150 2.7GHz Core, 2.7GHz, with 128 GB of RAM running Linux platform. Then, we modified

Algorithm 4 LDMOAD/DE

```
1: Input:
   Network parameters ($(|\mathscr{S}|, f_{ij}, |\mathscr{V}|, |\mathscr{E}| c_k, tc_{k,l}, R^{k,l}_{i,j}, d_{i,j})$. The algorithm stopping criterion.
2: Output:
   The set of feasible solutions which represent the mapping and scheduling result for each VNF of the incoming service request $\{X^1, ..., X^N\}$.
3: Define:
   $X = x^k_{ij}$, $\varphi = \Phi_{k,l}$;
4: Initialize:
   $x_{ijk} = \{0\}, \Phi_{k,l} = \{0\}$
   Generate the initial population $X^1, \ldots, X^N$ for the proposed problem by using a uniform random sampling.

5: for service request $i = 1 : |\mathscr{S}|$ do
6:     for VMs $k = 1 : |\mathscr{V}|$ do
7:         for VL $l = 1 : |\mathscr{E}|$ do

8: Calculate the current status capacity of every $VM\ node \rightarrow C_k$
9: Evaluate the fitness values for the three objectives $(FV)^i = F(X^i)$.
10: Applying all the genetic operators to the results.
11: Calculate the start transmitting and processing time. (Algo. 5)
12: Find the best feasible solution for all objective(s) from all population.
13:         end for
14:     end for
15: end for
```

(GA-BA) and (GA-NBA) algorithms [109] and applied them to the main function of the CMOEA/D-DE-ATP algorithm to compare our results with the results of both (GA-BA) and (GA-NBA) algorithms.

For a fair comparison, we applied the first and second objectives, and all extra constraints in subsection 7.2.4 over (GA-NBA) and (GA-NBA) algorithms, as both algorithms only support processing delay objective. Moreover, every k-th VM node support at most 3 VNFs functions on it, while every incoming request has a fixed number of VNFs (set to 5) which are required to process on these VMs. The expected processing delay time for every VNFs to be processed on VM nodes is set to be chosen randomly from [1, 10] milliseconds (ms). Fig. 7.4, 7.5 and 7.6 show the results for the first objective Z1 only,

Algorithm 5 Processing and Transmission Starting Time

```
1:  **Input:**
2:  The current available time for every VM
3:  The current schedule $(S, f_{ij}, K, c_k, N, A, D)$ $(|\mathscr{S}|, f_{ij}, |\mathscr{V}|, |\mathscr{E}| c_k, d_{i,j})$.
4:  **Output:**
5:  The best processing starting time $q_{ij}$ for the $j$-th VNFs of the $i$-th service request to
    process in the available VM.
6:  The best transmission starting time $\alpha_{ij}$ for the $j$-th VNFs of the $i$-th service request
    to transmit using the available virtual link.
7:  **Define:**
8:  t: the index of the available time in each $VM$.
9:  $st_{k,l}$: the index of the starting time.
10: $td_{ij-1}$: transmission delay of $j-1^{th}$ VNF chain of service request $s_i$
11: **for** $k = 1 : VMs$ **do**
12:     **for** $t = 1 : T$ (time intervals **do**
13: Sort the current available time $t$ for every $k$-th VM by the earliest availability
    $(st_{11} < st_{12} .... < st_{1t})$.
14: if the $j$-th VNF of the service request $s_i$ is the first of network service $(j = 1)$
15: if $x_{ij}^k q_{ij}^k < st_{kt}$
16: $q_{ij} = st_{kt}$
17: break;
18: else $q_{ij}^k \leq st_{kt} \& \alpha_{ij-1} + td_{ij-1} \leq q_{ij}$ $q_{ij} = \alpha_{ij-1} + td_{ij-1}$
    break; $q_{ij} = st_{kt}$
19:     **end for**
20: **end for**
21: **for all** V.links $m = 1 : V.links\ number$ **do**
    Sort the current available time t for every V.links connected with VMs by the earliest
    availability and assign it to start time $st_{kt}$. For instance the first virtual machine,
    $st_{11} < st_{12} .... < st_{1t}$
22:     **for all** time interval $t = 1 : TimeInterval$ **do**
    $j^{th}$ VNF chain of service request $s_i$ is the first of network service (j = 1) $x_{ij}^k q_{ij}^k < st_{kt}$
    $q_{ij} = st_{kt}$
    break;
    $q_{ij}^k \leq st_{kt} \& \alpha_{ij-1} + td_{ij-1} \leq q_{ij}$ $q_{ij} = \alpha_{ij-1} + td_{ij-1}$
    break; $q_{ij} = st_{kt}$
23:     **end for**
24: **end for**
```

the second objective Z2 only and the third objective Z3 only using the tested algorithms via a different number of incoming service requests (different network type). We can clearly see that Z1 and Z3 values increase gradually when the number of the incoming service requests increases, while there is a slight difference in Z2 values using the tested algorithms.

The experimental results in Table 7.1 illustrate the optimal values achieved for objectives Z1, Z2 and Z3 simultaneously by applying the proposed algorithm LDMOAD/DE, the modified GA-BA and GA-NBA algorithms. For instance, the optimal solution for the three objectives obtained by LDMOAD/DE algorithm in the large network instance is 17767 (ms), 2789 Kbps and 18235 (ms) with 10235 CPU execution time per run by applying LDMOAD/DE algorithm. While the optimal results obtained by applying a modified GA-NBA and GA-BA algorithms in large network instance for the three objectives are 19896 (ms), 5248 Kbps, 19632 (ms) with 12632 CPU execution time and 162527, 2096, 162643 with 12993 per run CPU time respectively. We can clearly see that our LDMOAD/DE algorithm is better than GA-BA algorithm in both objective values and CPU running time.

Moreover, table 7.1 reveals that there has been a steady rise in the execution time for all algorithms by increasing the network size. However, there is still a small difference in running time between the three algorithms. For instance, the running time by applying GA-NBA algorithm is more than the execution time of LDMOAD/DE algorithm in all network instances due to the way of calculating the processing and transmission starting time. Moreover, the running time by applying GA-BA algorithm is more than the execution time of GA-NBA algorithm in all network instances due to the way of calculating the bandwidth allocation for every virtual link. To conclude, it is clear that our LDMOAD/DE algorithm performs better than the GA-BA and GA-NBA algorithms in both objective Z1, Z2 and Z3 values and the CPU execution time for LDMOAD/DE algorithm is less than the modified GA-BA and GA-NBA algorithms.

Table 7.1: First, Second and Third objective values for LDMOEA/DE, modified GA-NBA and modified GA-BA algorithms and CPU execution time

Alg.	Network Instance	Z_1	Z_2	Z_3	CPU time (s)
LDMOEA/DE	S	5353	1524	5378	243
	M	11448	2798	11320	2954
	L	17767	4106	17285	10365
	XL	23330	5921	23932	18235
Modified GA-NBA	S	7346	1513	7456	350
	M	12639	3352	12584	3562
	L	19896	5248	19632	12632
	XL	29536	7096	29480	20256
Modified GA-BA	S	6876	1507	6934	423
	M	11782	3025	11778	3765
	L	18627	5008	18643	12993
	XL	27660	6682	27611	20882

7.5 Conclusion

This study presents a multi-objective mapping and scheduling algorithm for incoming NFV service requests. Given a set of VMs in the cloud and connected to each other using virtual links, every VM supports particular VNFs and has a particular properties (capacity, cost, and so on). We provide a mathematical formulation for the proposed problem and solve it as a multi-objective optimization problem using a LDMOEA/DE algorithm. The proposed algorithm determines the optimal scheduling and mapping for these incoming requests. There are three main objectives that are proposed in this chapter: minimizing the transmission delays occurring in every link, minimizing the processing capacity for every VM and minimizing the processing delay at every VM. The

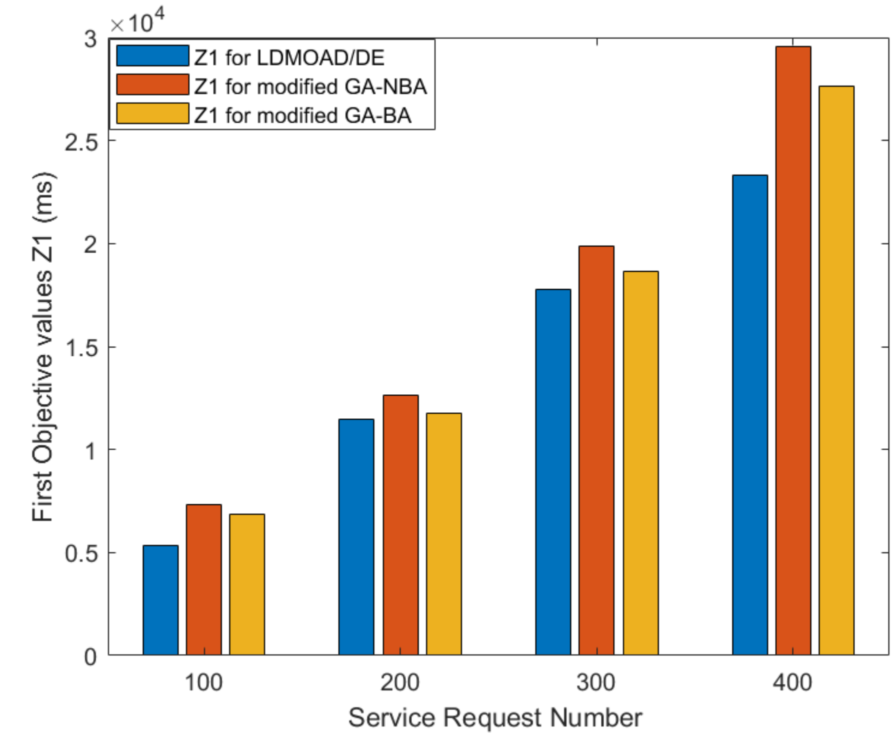


Figure 7.4: The optimal solution achieved by both proposed algorithms for the first objective value via all network instances

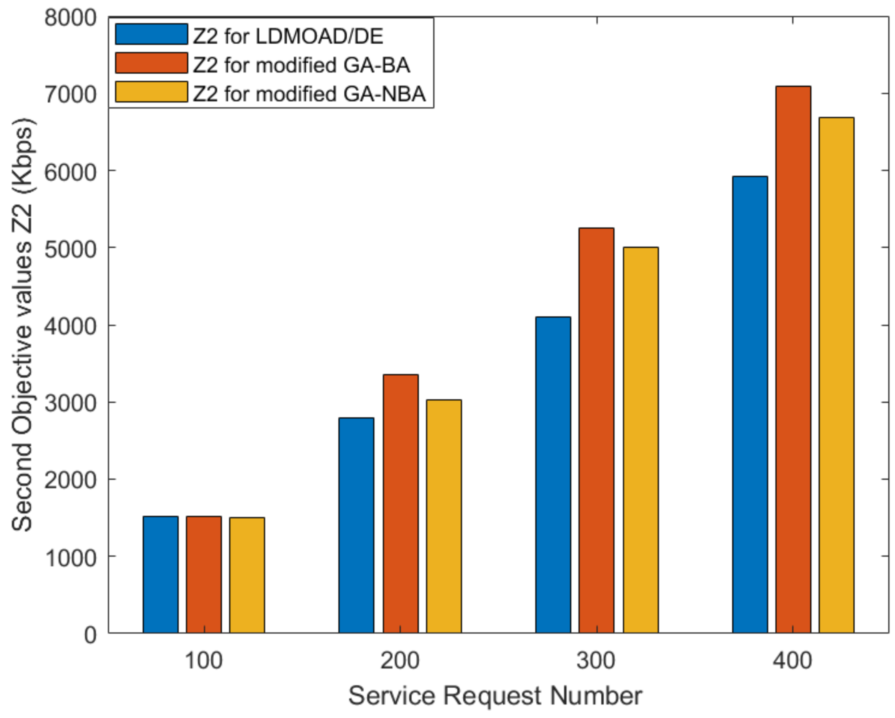


Figure 7.5: The optimal solution achieved by both proposed algorithms for the second objective value via all network instances

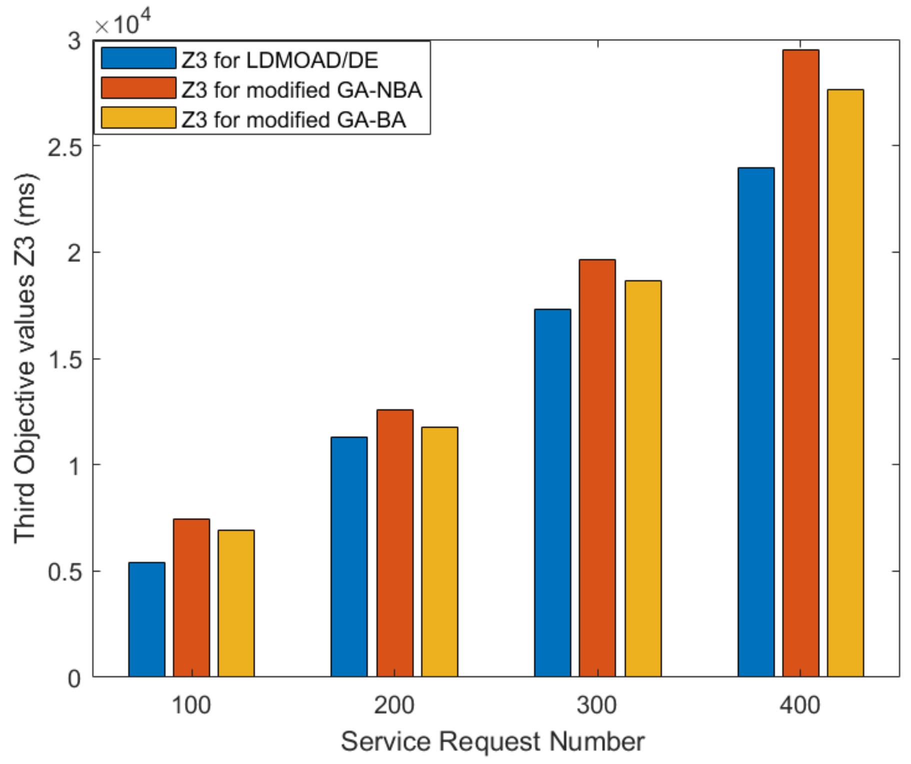


Figure 7.6: The optimal solution achieved by both proposed algorithms for the third objective value via all network instances

experimental results illustrate that the performance of LDMOAD/DE is better than that of the GA-NBA and GA-BA algorithms.

CHAPTER 8

MULTI OBJECTIVE RESOURCE OPTIMISATION FOR NETWORK FUNCTION VIRTUALISATION REQUESTS

8.1 Introduction

NFV is a new research concept for both academia and industry; however, it faces many challenges to be improved by the network operators. One of the main challenges faces NFV-RA problem addressed in this chapter is to find the optimal placement for a set of incoming requests with VNF service chains to serve in suitable VMs such that a set of conflicting objectives are met. Mainly, the focus is placed on maximizing the total saving cost by increasing the total CPU utilization during the processing time and increasing the processing time for every service request in the cloud network. Moreover, we aim to maximize the admitted traffic simultaneously, while considering system constraints.

SFC is a series of network functions that need to pass through a specific service flow. SFC provides flows classification and flow routes policy according to the availability of the network status. Several policies for the VNF service chains set up for different services should be considered.

For example, take a network structure of VMs with limited resources (e.g. VMs' bandwidth, CPU type, performance) and virtual link (each virtual link connected between two VMs is assigned with a limited capacity) as shown in Fig. 8.1. If VM node n_2 ($k \epsilon [1,K]_z$) runs $\{f_1, f_2\}$ on top of it, that means only VNFs chains $f_1$and f_2 can be executed or served at this VM node. If five different VNF requests $s_1 = [f_5, f_2, f_1 f_3, f_4]$,$s_2 = [f_3, f_2, f_4, f_5, f_1]$,$s_3 = [f_2, f_4, f_1, f_5, f_3]$,$s_4 = [f_1, f_2, f_4, f_3, f_5]$ and $s_5 = [f_4, f_3, f_5, f_2, f_1]$ arriving to the cloud need to be processed through the available VMs/servers. According to network structure, the node n_2 will run only functions f_1 and f_2 from these incoming request then will transfer it to another VM to process the other VNFs of the same request. During the processing of VNF service chains in VMs/servers, many physical resources (running time and CPU usage) can be utilized. Running VNFs in the data centre using commercially available hardware is significantly more cost effective than using costly dedicated hardware middle-boxes. Therefore, finding the best placement for the arrival requests with VNF chains to be processed on a suitable VM hosted on data-centres is a timely and vital problem which needs further research.

In this chapter, we formulate the problem as a multi-objective optimization problem and use a Resource Utilization Multi-Objective Evolutionary Algorithm based on Decomposition (RU-MOEA/D) algorithm to find a near-optimal solution for the proposed problem considering the two objectives simultaneously. Extensive simulations are carried out to evaluate the effects of the different network sizes, genetic parameters and the different number of resources on the acceptable ratio of the arrival VNF service chains to run in the available VMs. The empirical results illustrate that the proposed algorithm can solve the problem efficiently and compute the optimal solution for two objectives together within a reasonable running time.

This chapter is organized as follows: Section 8.2 presents a network formulation and

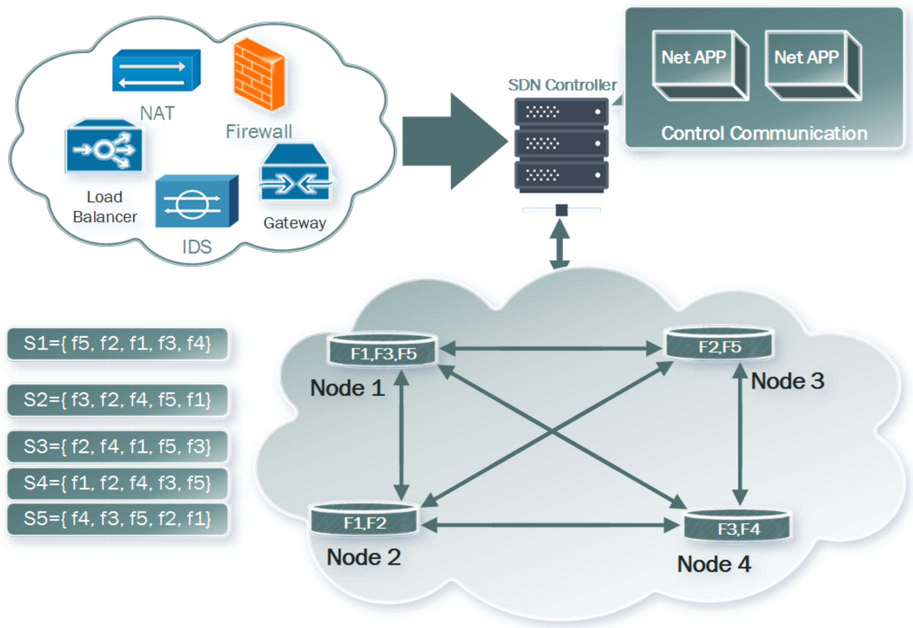


Figure 8.1: NFV Network structure

system modelling for the proposed problem. section 8.3 explains the proposed algorithm to solve the problem. Section 8.4 proposes the simulation results and the evaluation of the proposed solution and other proposed in the literature. Finally, section 8.5 conclude the main contribution proposed in this chapter.

8.2 VNF Mapping and Scheduling

In this section, we study resource optimization problem in NFV network structure, this structure is composed of a set of VMs/servers to run the arrival VNF service requests. Let $M = [m_1, ..., m_k]$ defines the final installed number of VMs in the cloud network, every VM has a capacity $c_m (m \in [1, k]_z)$ which can set by network operators or using a specific evaluation system. These VMs are connected together perfectly via set of virtual links $vl_b (b \in [1,\ L]_z)$, (the placement of these VMs in different servers and the final structure to connect between these VMs together using virtual links is out of the scope of this chapter).

We assume that the virtual links can transfer only one traffic flow R_i per unit time until the complete transmission of all chains of the current network service request is sent. A set of VNFs are supported by the installed VMs can be defined by $F = [f_1, f_2, ..., f_B]$.

8.2.1 Network Formulation

Assume a sequence of incoming service requests are defined as $R = [r_1, r_2, ..., r_d]_z$, where D defines the total number of the incoming service requests. Each service request $r_i \in R$ $\forall i \in [1, D]$ contains a set of VNFs in different order.

Let f_{ij} $\forall i \in [1, D], j \in [1, H]$ defines the j-th VNF function of network service r_i. Every request has a demand capacity $sc_i (i \in [1, D])$, and each request has assigned with traffic flow t_i to transfer from the source VM to the destination. The model proposed in this chapter considers the total buffer required for the j-th VNF of the service request r_i as τ_{ij}.

8.2.2 Definition of Variables

we define x_{ijk} to describe the assignment for the j-th VNF of the request r_i to serve on the available VM.

$$x_{ijk} = \begin{cases} 1 & \text{If the } J-\text{th VNF of the incoming service request } i \\ & \text{is accepted to run at k- th VM} \\ 0 & \text{otherwise} \end{cases} \tag{8.1}$$

Let Φ_{ijk} $(i \in [1, D]_z, j \in [1, H]_z, k \in [1, K]_z)$ describes the estimated processing time which required to process the j-th VNF functions of each service request r_i through the k-th VM. And, Ψ_i $(i \in [1, D]_z)$ defines the estimated processing time which required to process all request s_i.

We define ts_i and $te_i(i \in [1,D]_z)$ to define the starting and ending processing time for j-th VNF chain of request r_i. Simultaneously,α_{ij}indicate the starting forwarding time for j-th chain traffic flow of request r_i passing through v.links $l_m(m \in [1, L]_z)$.

8.2.3 Problem constraints

in this model, a set of system and network constrains are presented and applied for any solution to be feasible. These constraints are described as follow.

1. CPU usage constrain: the demand CPU U_{ij} for j-th VNF chains of request $r_i \forall i \in [1,D]$ to be run in the VM should not be exceed the total usage U_k of every VM node m_k $(k \in [1,K]_z)$.

$$(\delta_{ij} \leq U_k) \tag{8.2}$$

2. VMs available capacity constraint: The incoming requests $r_i \forall i \in [1,E]$ has a traffic flow demand rate R_i, this flow to run in available VM node $n_k(k \in [1,K]_z)$ should not exceed the current available capacity for this VM $c_m(k \in [1,K]_z)$. Formally,

$$\sum_i x_{ijk} * R_i \leq c_m \; (j \in [1,H]_z, k \in [1,K]_z) \tag{8.3}$$

3. VM processing constraint: for j-th VNF chain of incoming request service $r_i(i \in [1,D]_z)$, should be served by at least one available VM. Formally,

$$\sum_{k \in K} x_{ijk} \leq 1 \tag{8.4}$$

8.2.4 Optimization Objective

- **Maximize the total saving cost**

While every VNF function of network service request $r_i \in R$ is processing through VM node m_k $(k \in [1,K]_z)$, there are physical network resources (both CPU usage and time) utilized during this processing time, we define it as a cost in this model. First, CPU utilization is used to predict the system performance and calculated as sum of work controlled by a central processing. Let U_k denotes the CPU capability of every VM node n_k $(k \in [1,K]_z)$.

The core advantage of this objective is to count the total income saving in our system when the algorithm finds the optimal placement for incoming requests and utilize the network physical resources while its processing in a suitable VM node.

In other words, we need to maximize CPU capability U_k of every VM node n_k $(k \in [1,K]_z)$ after serving all j-th VNF chains τ_{ij} of request s_i $\forall i \in [1,E]$ and also maximize the total processing time for every arrival request$s_i \forall i \in [1,E]$ through VM nodes. Formally, the mathematical formula can be describe as follows.

$$Max\ Z_1(X) = \sum_{i=1}^{E} \sum_{j=1}^{H} \sum_{k=1}^{K} [(\Theta * (U_k - \tau_{ij})) + \varphi * (ts_i - te_i)] * x_{ijk} \tag{8.5}$$

$s.t.$ constraint (1) to (3)

where Θ, φ are constants proposed to scale the total saving costs for resources by utilizing the CPU and processing time. In this model, these constraints are set to 0.2.

- **Maximize the admitted traffic**

On one side, we aim to maximize the total quantity of admitted traffic to serve in the suitable VM nodes as expressed in equation. But on the other side, each VM can accept only a limited number of incoming VNF requests due to resource constraint in every VM including storage, CPU, and memory. In this model, we consider only CPU capability. In order to increase Z2, it requires increasing the processing rate in every VM which will lead to higher costs. Therefore, the main goal is to increase the incoming sufficient flow rate while considering a CPU capability for every VM. This objective is described as follow.

$$Max\ Z_2(X) = \sum_{i=1}^{E}\sum_{j=1}^{H}\sum_{k=1}^{K}(\Upsilon_k * x_{ijk}) - sc_i \tag{8.6}$$

$$s.t.\ \text{constraint (1) to (3)}$$

where Υ_k define the unit cost of VM node n_k ($k \in [1,K]_z$) resource consumed by running the incoming requests sc_i

8.3 Proposed Algorithm

In this section, the proposed RU-MOEA/D algorithm is studied in details to get the near optimal placement for arrival VNF chains.

We used the updated framework of the CMOEA/D-DE-ATP schema to solve constrains multi objective VNF chains placement problem. The framework contains all the standard techniques of the genetic algorithm: generating an initial population, representation, selection, crossover, and mutation.

We first initialize the problem parameters (total iterations, decomposition method of choice, total function evaluation number). The binary code is used to represent variable x_{ijk} which defines the VNF function of request r_i if can process on node n_k or not. After that, construct an initial population, representing different individuals of the population, selecting the fittest individuals. In our (RU-MOEA/D) algorithm, the population size is set to 300 (for each population size, the algorithm repeats 300 iteration, so the total number of repetition is 3000 (300*100)), and the initial populations are chosen randomly. Applying a crossover operation that mates individuals, a mating selection is used for selection operation, and DE operator is used for crossover operation. Finally, (1/parameters dimension) is used as a mutation rate in this schema.

The algorithm will stop when it reaches the maximum number of iterations (300 iterations in this model (can be set by the provider)) or only after 100 iterations if there is not any enhancement in the value of the objective. After that, the termination criterion checks the new solution to decide whether the search should stop or continue at every iteration. The final output should satisfy the problem constraints mentioned in subsection 8.2.3, otherwise, the solution will be removed.

8.4 NUMERICAL RESULTS

In this section, we implement RU-MOEA/D algorithm, after modifying the main function of the CMOEA/D-DE-ATP algorithm and apply the simulation settings including the genetic standard techniques explained in the previous section, network topology, and algorithm parameter settings. In RU-MOEA/D, we provide a heuristic approach to maximize and organize the distribution of the incoming VNF chains and find a suitable VM to run on top of it with the cheapest cost. If a VM node has enough resources to host the incoming VNF chains, then the program automatically search to another valid and available VM to run these VNF chains in it.

To evaluate the performance of the RU-MOEA/D algorithm, three different network structures were presented to find the optimal solution of our proposed problem. Firstly, we applied the proposed algorithm to evaluate the first objective and find the maximum number of VNF chains that can run in installed VMs and achieve the minimum cost (explained in detail previously in first objective). Second, the proposed algorithm implemented to evaluate only the second objective while considering the network constraints. Finally, we present simulation results and explanations for the two objectives simultaneously of the proposed algorithm.

Simulation set up: We implement all proposed algorithms in MATLAB and run the experiments on Intel Core i7-5300U CPU, with 32GB of RAM running on Redhat platform. In these network instance, the network contains 10 VM nodes (fully connected between virtual network), and the network service sizes are distributed uniformly 10 KB to 1600 KB respectively depending on their number in every topology. The VM capacity is set to be 10 Mbps, and the link bandwidth is put equally 1 Mbps. Moreover, the total capacity set for every virtual link equals 1 Mbps. Moreover, we assume that the processing time is chosen from time set [0, 10] milliseconds(ms), and the processing delay is set from time [10, 20].
Evaluation: In this model, we used the platform of genetic non-bandwidth link allocation (GA-NBA) algorithm presented in [109]. We adapted this scheme and applied it in the main function of MOEA /D-DE algorithm to achieve a fair comparison between two algorithms. Furthermore, we added the network topology, objective function, and constraints to the algorithm.

Table 8.1 explains the final output near-optimal solutions for the first objective and second objectives achieved by implementing the proposed algorithm via different no. of VNF chains. We can clearly see that the execution time for our schema RU-MOEA/D is less than the running time in GA-NBA algorithm. This is mainly due to the function used to calculate the start processing and transmission time. For example, the execution time for 100 requests is 56 second (s) for RU-MOEA/D scheme while it is 240 (s) for GA-NBA algorithm.

The final outcomes are shown in Fig. 8.4 which were achieved from 30 trials. We can see that our approach with RU-MOEA/D performs slightly better than the outcome of GA-NBA, for instance when the no. of requests are 200, the first and second objectives values are 4053 and 8421 respectively ac hived by our RU-MOEA/D algorithm, while, the first and second objectives values are 3990 and 8359 respectively achieved by GA-NBA.

The core advantage of the algorithm is that we can use it to find the optimal solution for single or multi objectives problem at the same time. For instance, fig. 8.4 summarizes the final results of the first objective only when we applied both algorithms to solve the problem via different number of arrival requests. The graph shows that there has been a gradual increase in the value of objective Z1 achieved by applying RU-MOEA/D scheme.

Table 8.1: Near optimal solution values achieved for two objectives simultaneously

Alg.	Service No.	Z1	Z2	CPU time (s)
RU-MOEA/D	50	300	885	36
	100	2119	5762	56
	150	3237	7961	78
	200	4053	8421	102
	250	5137	9368	131
	300	6239	9856	163
GA-NBA	50	278	803	41
	100	1198	3707	240
	150	3139	7887	365
	200	3990	8359	523
	250	5089	9275	749
	300	6165	9345	163

Similar to the objective Z2, we applied both algorithms to find the optimal solution for objective Z2 as shown in Fig. 8.4. What can be clearly seen in this figure is the slight difference between the two objective values but RU-MOEA/D scheme is better than GA-NBA in most of the test instances.

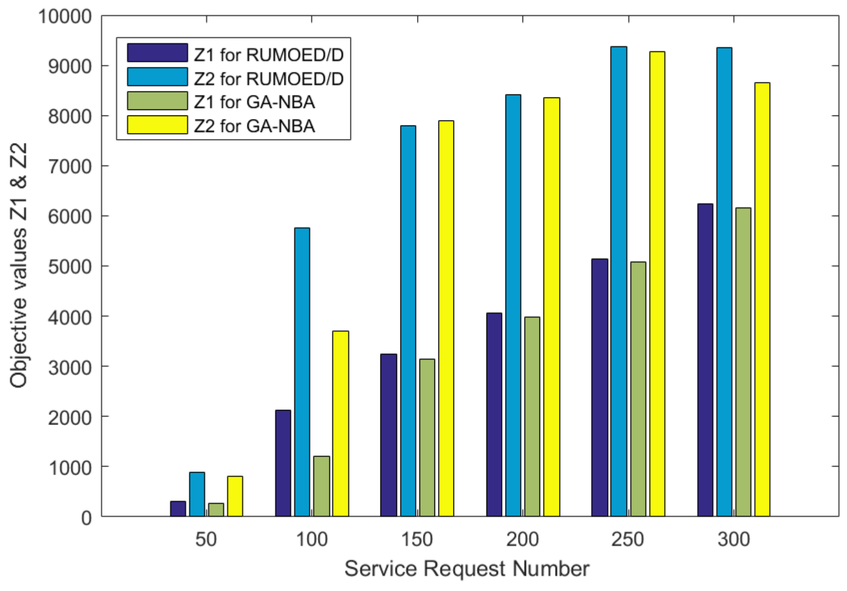


Figure 8.2: Near optimal VNF chains placement achieved by finding the best values for the first and second objectives together through various service requests number

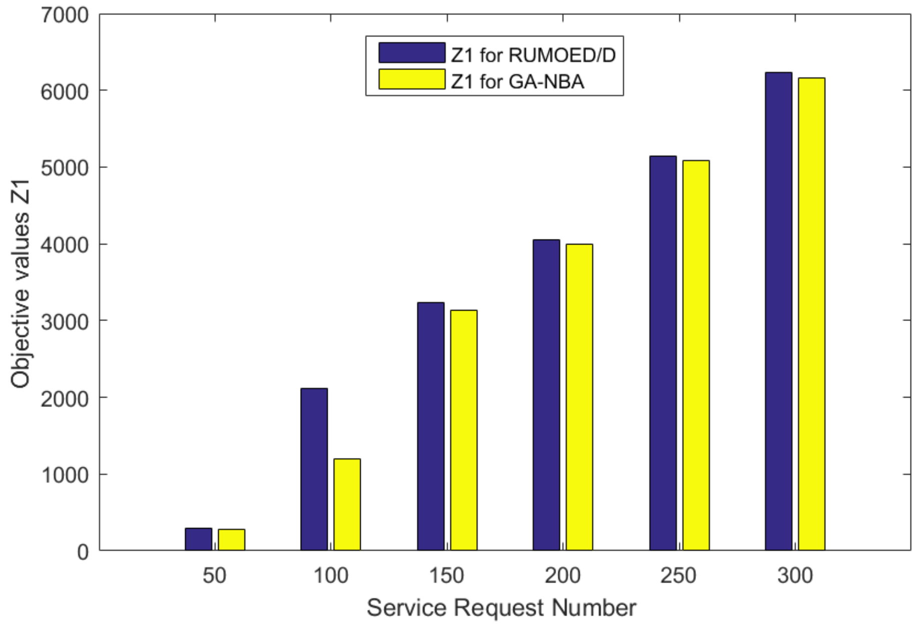


Figure 8.3: First Objective values via different numbers of incoming service request

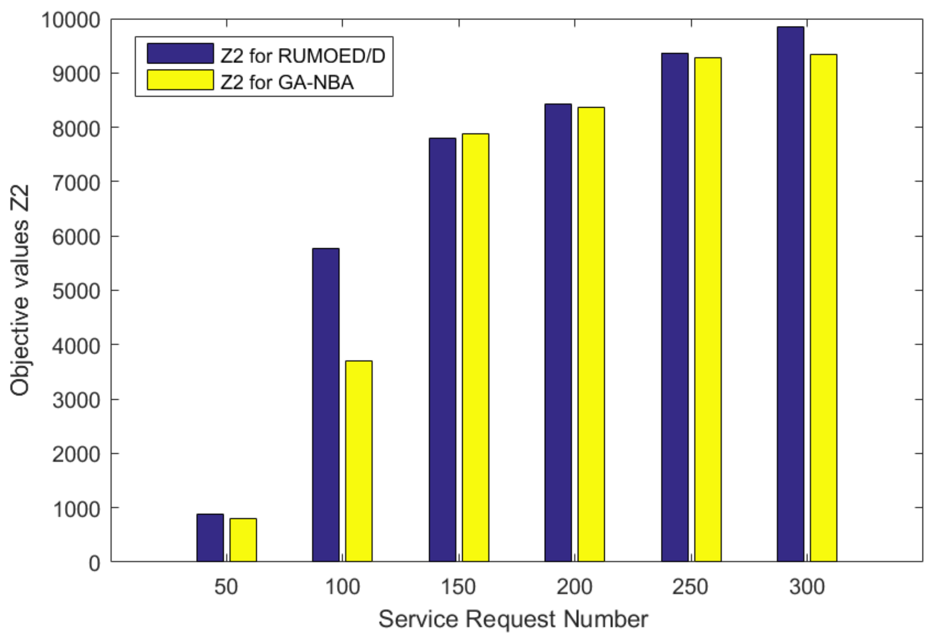


Figure 8.4: The outperform of the proposed algorithms by achieving the near to optimal solution for the second objective value after changing the number of incoming service requests

8.5 Conclusion

In this chapter, we considered one of the main problems faced by NFV which is to find the best placement for the incoming VNF service chains to process through suitable VMs with the aim of maximizing the CPU utilization of the machines running in the datacenters and minimize the network cost simultaneously. We formulated the mathematical model for the two objectives and considering system and network constraints. We proposed a heuristic RU-MOEA/D algorithm to find the near optimal solution for both objectives simultaneously with low computational complexity compared with the GA-NBA algorithm.

CHAPTER 9

Conclusion and Future work

9.1 Conclusion

VNF service requests resource allocation is a vital problem to be solved for mapping and scheduling incoming service requests in the NFV environment. This thesis has presented the implementation and evaluation to find the near-to-optimal solution for mapping and scheduling of VNFs of the incoming service requests.

Assume the NFV environment consists of a set of VM nodes each of which has specific properties and supports a particular number of the unique VNFs, and also a set of incoming service requests, each of which has a predefined non-negative weight, for which of this weight reflects the request priority.

A formal mathematical formulation for different network scenarios and several optimization strategies have been presented. We studied the relationships of mapping the incoming service requests to the available VMs then finding the best schedule for the VNFs of these requests to process in order on different VMs. Moreover, we proposed different network constraints and several approaches to cover all the network scenarios for this problem.

The first formulation aims to maximize the number of accepted incoming service requests, and to minimize the link utilization and the overall processing time of service requests, while considering the capacity of the links between VMs, as well as the forwarding transmission and processing delays constraints. The developments of the second goal consider the scenario of accepting different service requests which arrive to the cloud periodically. The developments of the optimization problem have been done by maximizing the acceptance rate of the non-uniform arrival service requests, minimizing the number of bottleneck links and the overall processing time.

The optimization model of the third goal has been developed to consider the uniform and non-uniform arrival of the incoming service requests and also the model considers the expiry time of the incoming service requests to be processed in the VMs. The model aims to achieve four objectives functions, namely, maximizing the acceptance rate, minimizing the number of bottleneck links, the overall processing time and the relative processing time, while considering the VM capacity, link traffic, VNF's order, expiry time and link capacity constraints.

The optimization model developed in the fourth formulation minimizes the processing delay at every VM, the transmission delays occurring at every link, and processing capacity of every VM, while taking into consideration the delay constraints. In the fifth and final scenario, the optimization model aims to minimize the processing time for every accepted service request, and maximize the number of accepted service requests, while considering the system constraints, including the processing capacity of the VMs.

All five scenarios have been addressed as both a single-objective and a multi-objective optimization problem where two different evolutionary algorithms based on genetic algorithm have been applied for solving both cases. In particular, the two algorithms are based on multi-objective evolutionary algorithm based on decomposition (MOEA/D), and Non-dominated Sorting Genetic Algorithm II (NSGA-II).

The numerical simulations show that the proposed algorithms solve the first, second and the third scenario efficiently and converge to the near to optimal solution. With respect to the last two scenarios, the numerical evaluations demonstrate that the developed algorithm are scalable and that outperform the evolutionary algorithms proposed in the literature, the GA-BA and GA-NBA algorithms. The execution times of all proposed algorithms are also analyzed for all five scenarios.

9.2 Future Work

The research of optimizing NFV resource allocation is largely needed for deployment of future network architectures based on NFV. The scope of this research topic provides many directions for future work:

- Propose an organized way to select a suitable network link and VM elastically depending upon the number of service requests, VNF chain demands and network constraints. In particular, determine the best route for VNFs for each service request to be transferred between VMs given network link availability. Further, identify the best VM to process VNFs for arriving service requests according to VM capacity and usage. The solution should achieve multiple objectives simultaneously and also check whether the VNFs for arriving requests should be deployed through additional VMs or be deployed through resources provided by existing VMs.
- Develop a new model that combines VNF-FGE with mapping (locations of the VNFs to be processed) and scheduling (time for VNFs of arriving service requests to start transmission or processing) problems presented in this thesis. This combined model would aim to identify the best mapping and scheduling of incoming service

requests. Further it would simultaneously find the best deployment of VNFs to be executed through different VMs. The new model would explore service execution time, network performance and costs (processing and transmission).

- Propose an efficient scheme to deploy VNF chains across VMs and effectively schedule the arriving service requests to be processed by the VMs. This should maximize utilization of VMs, minimize the transmission and processing of each VNF for each service request and simultaneously minimize the execution time to successfully process these service requests.
- Investigate the performance of different genetic algorithms by applying different GA operators to solve the VNF-RA problem. Given the outcomes of the results, choose the best selection, crossover, and mutation operators to be applied in the NFV-RA algorithms.
- Propose an efficient algorithm to transmit the VNFs for individual service requests between VMs, such that varying objectives such as operating cost, energy-saving, failure recovery, and load balancing may be met.
- Explore other improvements that could be achieved by applying and testing the proposed algorithms using a real test-bed scenario. The results obtained from this real test-bed scenario will help further development of NFV-RA to mapping, scheduling and routing problems.

BIBLIOGRAPHY

[1] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[2] D. Kreutz, F. M. V. Ramos, P. E. Ver, X00Ed, Ssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] M. R. Sama, L. M. Contreras, J. Kaippallimalil, I. Akiyoshi, H. Qian, and H. Ni, "Software-defined control of the virtualized mobile packet core," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 107–115, 2015.

[4] J. G. Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. PP, no. 99, p. 1, 2016.

[5] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[6] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi, "Latency-aware composition of virtual functions in 5G," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–6.

[7] Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[9] N. Nikaein, E. Schiller, R. Favraud, K. Katsalis, D. Stavropoulos, I. Alyafawi, Z. Zhao, T. Braun, and T. Korakis, “Network store: Exploring slicing in future 5g networks,” in *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*. ACM, 2015, pp. 8–13.

[10] Y. Li and M. Chen, “Software-defined network function virtualization: A survey,” *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[11] P. Pate, “NFV and SDN: What’s the Difference?” 2013. [Online]. Available: https://www.sdxcentral.com/articles/contributed/nfv-and-sdn-whats-the-difference/2013/03/

[12] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On scalability of software-defined networking,” *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.

[13] P. Newman, G. Minshall, and T. L. Lyon, “Ip switching-atm under ip,” *IEEE/ACM Transactions on networking*, no. 2, pp. 117–129, 1998.

[14] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “NOX: towards an operating system for networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[15] G. ETSI, “Network functions virtualisation (nfv): Architectural framework,” *ETsI Gs NFV*, vol. 2, no. 2, p. V1, 2013.

[16] N. ETSI, “Network functions virtualisation (NFV); terminology for main concepts in NFV,” *Group Specification, Dec*, 2014.

[17] P. Quinn and T. Nadeau, “Service function chaining problem statement,” *draft-ietf-sfc-problem-statement-07 (work in progress)*, 2014.

[18] R. Mijumbi, “Self-managed resources in network virtualisation environments,” 2014.

[19] A. Belbekkouche, M. M. Hasan, and A. Karmouch, “Resource discovery and allocation in network virtualization,” *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1114–1128, 2012.

[20] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, “Virtual network embedding: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[21] J. F. Botero, M. Molina, X. Hesselbach-Serra, and J. R. Amazonas, “A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems,” *Journal of Network and Computer Applications*, vol. 36, no. 6, pp. 1735–1752, 2013.

[22] B. Wang, X. Chang, J. Liu, and J. K. Muppala, “Reducing power consumption in embedding virtual infrastructures,” in *2012 IEEE Globecom Workshops*. IEEE, 2012, pp. 714–718.

[23] M. Yu, Y. Yi, J. Rexford, and M. Chiang, “Rethinking virtual network embedding: substrate support for path splitting and migration,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[24] Y. Chen, J. Li, T. Wo, C. Hu, and W. Liu, “Resilient virtual network service provision in network virtualization environments,” in *2010 IEEE 16th International Conference on Parallel and Distributed Systems*. IEEE, 2010, pp. 51–58.

[25] E. Amaldi, S. Coniglio, A. M. C. A. Koster, and M. Tieves, “On the computational complexity of the virtual network embedding problem,” *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213–220, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571065316300336

[26] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer, “Energy efficient virtual network embedding,” *IEEE Communications Letters*, vol. 16, no. 5, pp. 756–759, 2012.

[27] J. F. Botero and X. Hesselbach, “Greener networking in a network virtualization environment,” *Computer Networks*, vol. 57, no. 9, pp. 2021–2039, 2013.

[28] S. Liu, Z. Cai, H. Xu, and M. Xu, “Security-aware virtual network embedding,” in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 834–840.

[29] SDNCentral, “IETF (Internet Engineering Task Force).” [Online]. Available: https://www.sdxcentral.com/listings/ietf/

[30] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, “Near optimal placement of virtual network functions,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 1346–1354.

[31] S. Lange, S. Gebert, J. Spoerhase, P. Rygielski, T. Zinner, S. Kounev, and P. Tran-Gia, “Specialized Heuristics for the Controller Placement Problem in Large Scale SDN Networks,” in *Teletraffic Congress (ITC 27), 2015 27th International*. IEEE, 2015, pp. 210–218.

[32] T. Taleb, M. Bagaa, and A. Ksentini, “User mobility-aware Virtual Network Function placement for Virtual 5G Network Infrastructure,” in *IEEE International Conference on Communications*, vol. 2015-Septe. Institute of Electrical and Electronics Engineers Inc., sep 2015, pp. 3879–3884.

[33] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, “Network function placement for NFV chaining in packet/optical datacenters,” *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

[34] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Wood, “Virtual function placement and traffic steering in flexible and dynamic software defined networks,” in *The 21st IEEE International Workshop on Local and Metropolitan Area Networks*. IEEE, 2015, pp. 1–6.

[35] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin, “Improve service chaining performance with optimized middlebox placement,” *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 560–573, 2015.

[36] J. Soares, C. Gonçalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, R. L. Aguiar, and S. Sargento, “Toward a telco cloud environment for service functions,” *IEEE Communications Magazine*, vol. 53, no. 2, pp. 98–106, 2015.

[37] B. Addis, D. Belabed, M. Bouet, and S. Secci, “Virtual network functions placement and routing optimization,” *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pp. 171–177, nov 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7335301

[38] P. Veitch, M. J. McGrath, and V. Bayon, “An instrumentation and analytics framework for optimal and robust NFV deployment,” *IEEE Communications Magazine*, vol. 53, no. 2, pp. 126–133, 2015.

[39] E. S. Correa, L. A. Fletscher, and J. F. Botero, “Virtual data center embedding: A survey,” *IEEE Latin America Transactions*, vol. 13, no. 5, pp. 1661–1670, 2015.

[40] Z. Kong, C.-Z. Xu, and M. Guo, “Mechanism design for stochastic virtual resource allocation in non-cooperative cloud systems,” in *2011 IEEE 4th International Conference on Cloud Computing*. IEEE, 2011, pp. 614–621.

[41] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, “Data center network virtualization: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 909–928, 2012.

[42] S. Gass and T. Saaty, “The computational algorithm for the parametric objective function,” *Naval Research Logistics (NRL)*, vol. 2, no. 1,Äê2, pp. 39–45, 1955.

[43] I. E. Grossmann, “Advances in mathematical programming models for enterprise-wide optimization,” *Computers & Chemical Engineering*, vol. 47, pp. 2–18, 2012.

[44] A. Makhorin, “GLPK (GNU linear programming kit),” *http://www. gnu. org/s/glpk/glpk. html*, 2008.

[45] H. Moens and F. D. Turck, “VNF-P: A model for efficient placement of virtualized network functions,” in *10th International Conference on Network and Service Management (CNSM) and Workshop*, 2014, pp. 418–423.

[46] A. Baumgartner, V. S. Reddy, and T. Bauschert, “Combined virtual mobile core network function placement and topology optimization with latency bounds,” in *2015 Fourth European Workshop on Software Defined Networks*. IEEE, 2015, pp. 97–102.

[47] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Optimal network function virtualization realizing end-to-end requests," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[48] R. Riggio, T. Rasheed, and R. Narayanan, "Virtual network functions orchestration in enterprise WLANs," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 1220–1225.

[49] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," Chicago, Illinois, USA, pp. 33–38, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2627585.2627592

[50] J. F. Riera, X. Hesselbach, M. Zotkiewicz, M. Szostak, and J.-F. Botero, "Modelling the NFV forwarding graph for an optimal network service deployment," in *2015 17th International Conference on Transparent Optical Networks (ICTON)*. IEEE, 2015, pp. 1–4.

[51] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. IEEE, 2015, pp. 255–260.

[52] R. Bruschi, A. Carrega, and F. Davoli, "A game for energy-aware allocation of virtualized network functions," *Journal of Electrical and Computer Engineering*, vol. 2016, p. 2, 2016.

[53] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling Wireless Virtual Networks Functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, 2016.

[54] M. Bagaa, T. Taleb, and A. Ksentini, “Service-aware network function placement for efficient traffic handling in carrier cloud,” in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2014, pp. 2402–2407.

[55] B. Németh, J. Czentye, G. Vaszkun, L. Csikor, and B. Sonkoly, “Customizable real-time service graph mapping algorithm in carrier grade networks,” in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 28–30.

[56] M. Bouet, J. Leguay, T. Combe, and V. Conan, “Cost‚Äêbased placement of vDPI functions in NFV infrastructures,” *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.

[57] Q. Zhang, X. Wang, I. Kim, P. Palacharla, and T. Ikeuchi, “Vertex-centric computation of service function chains in multi-domain networks,” in *2016 IEEE netsoft conference and workshops (NetSoft)*. IEEE, 2016, pp. 211–218.

[58] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “On orchestrating virtual network functions,” in *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 50–56.

[59] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, “Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 98–106.

[60] J. Elias, F. Martignon, S. Paris, and J. Wang, “Efficient orchestration mechanisms for congestion mitigation in nfv: Models and algorithms,” *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 534–546, 2015.

[61] R. Mijumbi, J. Serrat, J.-L. Gorricho, J. Rubio-Loyola, and S. Davy, "Server placement and assignment in virtualized radio access networks," in *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 398–401.

[62] P. Bellavista, F. Callegati, W. Cerroni, C. Contoli, A. Corradi, L. Foschini, A. Pernafini, and G. Santandrea, "Virtual network function embedding in real cloud environments," *Computer Networks*, vol. 93, pp. 506–517, 2015.

[63] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Demand-aware network function placement," *Journal of Lightwave Technology*, vol. 34, no. 11, pp. 2590–2600, 2016.

[64] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling Wireless Virtual Networks Functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, 2016. [Online]. Available: file:///C:/Users/12438608/Dropbox/Australia/PhD/MahmoudGamalPhDstudent/Paperstoread/ResourceAllocation/SchedulingWirelessVirtualNetworksFunctions.pdf

[65] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba, "Service function chaining simplified," *arXiv preprint arXiv:1601.00751*, 2016.

[66] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service function chaining in next generation networks: State of the art and research challenges," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 216–223, 2016.

[67] F. Schneider, T. Egawa, S. Schaller, S.-i. Hayano, M. Schöller, and F. Zdarsky, "Standardizations of sdn and its practical implementation," *NEC Technical Journal*, vol. 8, no. 2, pp. 16–20, 2014.

[68] M. Boucadair, C. Jacquenet, R. Parker, D. Lopez, J. Guichard, and C. Pignataro, "Service Function Chaining: Framework and Architecture. Internet-Draft draft-boucadairsfc-framework-02," *Active Internet-Draft, IETF Secretariat, Tech. Rep.*, 2014.

[69] G. Nagy and S. Salhi, "Location-routing: Issues, models and methods," *European journal of operational research*, vol. 177, no. 2, pp. 649–672, 2007.

[70] C. Prodhon and C. Prins, "A survey of recent research on location-routing problems," *European Journal of Operational Research*, vol. 238, no. 1, pp. 1–17, 2014.

[71] D. Joseph and I. Stoica, "Modeling middleboxes," *IEEE network*, vol. 22, no. 5, pp. 20–25, 2008.

[72] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, V. Sekar, and A. Akella, "Stratos: A network-aware orchestration layer for virtual middleboxes in clouds," *arXiv preprint arXiv:1305.0209*, 2013.

[73] S. Sahhaf, W. Tavernier, J. Czentye, B. Sonkoly, P. Sköldström, D. Jocha, and J. Garay, "Scalable architecture for service function chain orchestration," in *2015 Fourth European Workshop on Software Defined Networks*. IEEE, 2015, pp. 19–24.

[74] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service chaining using virtual network functions in network-enabled cloud systems," in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*. IEEE, 2015, pp. 1–3.

[75] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *International Colloquium on Structural Information and Communication Complexity*. Springer, 2015, pp. 104–118.

[76] M. T. Beck and J. F. Botero, “Coordinated Allocation of Service Function Chains,” in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[77] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, “Distributed Service Function Chaining,” *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2479–2489, 2017.

[78] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, 2014, pp. 7–13.

[79] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, “The dynamic placement of virtual network functions,” in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[80] T. W. Kuo, B. H. Liou, K. C. J. Lin, and M. J. Tsai, “Deploying chains of virtual network functions: On the relation between link and server usage,” in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[81] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, “A Scalable Approach for Service Chain Mapping With Multiple SC Instances in a Wide-Area Network,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 529–541, 2018.

[82] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and S. Davy, “Design and evaluation of algorithms for mapping and scheduling of virtual network functions,” in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, 2015, pp. 1–9.

[83] X. Li and C. Qian, "Low-complexity multi-resource packet scheduling for network function virtualization," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1400–1408.

[84] H. Huang, P. Li, S. Guo, W. Liang, and K. Wang, "Near-Optimal Deployment of Service Chains by Exploiting Correlations between Network Functions," *IEEE Transactions on Cloud Computing*, p. 1, 2017.

[85] M. Yoshida, W. Shen, T. Kawabata, K. Minato, and W. Imajuku, "MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure," *APNOMS 2014 - 16th Asia-Pacific Network Operations and Management Symposium*, dec 2014.

[86] Q. Zhang, Y. Xiao, F. Liu, J. C. S. Lui, J. Guo, and T. Wang, "Joint Optimization of Chain Placement and Request Scheduling for Network Function Virtualization," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 731–741.

[87] L. Qu, C. Assi, and K. Shaban, "Delay-Aware Scheduling and Resource Optimization With Network Function Virtualization," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3746–3758, 2016. [Online]. Available: file:///C:/Users/12438608/Dropbox/Australia/PhD/MahmoudGamalPhDstudent/Paperstoread/ResourceAllocation/Delay-AwareSchedulingandResourceOptimization.pdf

[88] I. Jang, D. Suh, S. Pack, and G. Dán, "Joint Optimization of Service Function Placement and Flow Distribution for Service Function Chaining," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2532–2541, 2017.

[89] L. Wang, Z. Lu, X. Wen, R. Knopp, and R. Gupta, “Joint optimization of service function chaining and resource allocation in network function virtualization,” *IEEE Access*, vol. 4, pp. 8084–8094, 2016.

[90] A. Abraham and L. Jain, “Evolutionary multiobjective optimization,” in *Evolutionary Multiobjective Optimization*. Springer, 2005, pp. 1–6.

[91] S. Kukkonen and J. Lampinen, “GDE3: The third evolution step of generalized differential evolution,” in *2005 IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2005, pp. 443–450.

[92] E. J. Hughes, “Evolutionary many-objective optimisation: many once or one many?” in *2005 IEEE congress on evolutionary computation*, vol. 1. IEEE, 2005, pp. 222–227.

[93] V. Khare, X. Yao, and K. Deb, “Performance scaling of multi-objective evolutionary algorithms,” in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2003, pp. 376–390.

[94] N. Beume, B. Naujoks, and M. Emmerich, “SMS-EMOA: Multiobjective selection based on dominated hypervolume,” *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[95] J. Bader and E. Zitzler, “HypE: An algorithm for fast hypervolume-based many-objective optimization,” *Evolutionary computation*, vol. 19, no. 1, pp. 45–76, 2011.

[96] T. Wagner, N. Beume, and B. Naujoks, “Pareto-, aggregation-, and indicator-based methods in many-objective optimization,” in *International conference on evolutionary multi-criterion optimization*. Springer, 2007, pp. 742–756.

[97] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[98] M. Ehrgott, "A discussion of scalarization techniques for multiple objective integer programming," *Annals of Operations Research*, vol. 147, no. 1, pp. 343–360, 2006.

[99] K. Klamroth and T. Jørgen, "Constrained optimization using multiple objective programming," *Journal of Global Optimization*, vol. 37, no. 3, pp. 325–355, 2007.

[100] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE transactions on evolutionary computation*, vol. 13, no. 2, pp. 284–302, 2009.

[101] M. A. Jan and Q. Zhang, "MOEA/D for constrained multiobjective optimization: Some preliminary experimental results," in *Computational Intelligence (UKCI), 2010 UK Workshop on*. IEEE, 2010, pp. 1–6.

[102] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary computation*, vol. 4, no. 1, pp. 1–32, 1996.

[103] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary computation*, vol. 7, no. 1, pp. 19–44, 1999.

[104] D. E. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex systems*, vol. 3, no. 5, pp. 493–530, 1989.

[105] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 2, research topics," *University computing*, vol. 15, no. 4, pp. 170–181, 1993.

[106] C. Reeves and J. E. Rowe, *Genetic algorithms: principles and perspectives: a guide to GA theory*. Springer Science & Business Media, 2002, vol. 20.

[107] T. Kellegöz, B. Toklu, and J. Wilson, “Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem,” *Applied Mathematics and Computation*, vol. 199, no. 2, pp. 590–598, 2008.

[108] A. C. Koenig, “A study of mutation methods for evolutionary algorithms,” *University of Missouri-Rolla*, 2002.

[109]

[110] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, “Toward a software-based network: integrating software defined networking and network function virtualization,” *IEEE Network*, vol. 29, no. 3, pp. 36–41, 2015.

[111] S. Sun, M. Kadoch, L. Gong, and B. Rong, “Integrating network function virtualization with SDR and SDN for 4G/5G networks,” *IEEE Network*, vol. 29, no. 3, pp. 54–59, 2015.

[112] K. Deb, “Multi-objective optimization,” in *Search methodologies*. Springer, 2014, pp. 403–449.

[113] Q. Zhang, W. Liu, and H. Li, “The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances,” in *Evolutionary Computation, 2009. CEC’09. IEEE Congress on*. IEEE, 2009, pp. 203–208.