

UNIVERSIDADE D
COIMBRA

Noé Paulo Lopes Godinho

Optimising Resource Allocation in Fog Computing

Tese no âmbito do Doutoramento em Engenharia Informática, especialização em Arquiteturas, Redes e Cibersegurança, orientada pelo Professor Doutor Luís Filipe dos Santos Coelho Paquete e pela Professora Doutora Marilia Pascoal Curado, e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Agosto de 2024



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Optimising Resource Allocation in Fog Computing

Noé Paulo Lopes Godinho

PhD in Informatics Engineering
PhD Thesis submitted to the University of Coimbra

Advised by Prof. Dr. Luís Filipe dos Santos Coelho Paquete
and Prof. Dr. Marilia Pascoal Curado

August, 2024



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Optimização de Alocação de Recursos em Computação em Névoa

Noé Paulo Lopes Godinho

Doutoramento em Engenharia Informática
Tese de Doutoramento apresentada à Universidade de Coimbra

Orientado pelo Prof. Dr. Luís Filipe dos Santos Coelho Paquete
e pela Prof. Dr. Marilia Pascoal Curado

Agosto, 2024

This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) under PhD grant DFA/BD/7293/2020 (doi: 10.54499/2020.07293.BD) and project grants UIDB/CEC/00326/2020, P2020 SAICTPAC/0011/2015, POCI/01/0247/FEDER/049029, CENTRO-01-0247-FEDER-045929, POCI/01/0247/FEDER/024539 and PTDC/EEI-COM/5284/2020.



Imagination will often carry us to worlds that never were.
But without it, we go nowhere.

Carl Sagan

Acknowledgments

I would like to express my deepest gratitude to my advisors, Professor Luís Paquete and Professor Marilia Curado, whose guidance, encouragement, and unwavering support have been instrumental in the completion of this work. Their insightful feedback and constant availability for discussion have helped shape my research and pushed me to strive for excellence. I am thankful for the countless hours they have invested in mentoring me, and I will always be grateful for their patience and belief in my potential.

My heartfelt thanks also go to Filipe Sequeira, Luís Ventura, Ana Alves, Ana Duarte, João Campos, Inês Valentim, and Kevin Bento, who have been very supportive throughout this journey. Their constant companionship, words of encouragement, and understanding during the challenging times have made this experience more manageable and enjoyable. I am grateful for the many moments of laughter and the invaluable advice they have offered, which have kept me motivated and focused. The camaraderie we share has been a source of strength, and I am fortunate to have such wonderful people in my life.

I would also like to extend my appreciation to all those who have contributed to this project in various ways. Whether it was providing feedback, sharing resources, or simply offering a listening ear, each of you has played a part in the successful completion of this work. Your support has been a vital part of this journey, and I am deeply thankful for your generosity and kindness.

Lastly, I am eternally grateful to my mother, Benilde, whose love and sacrifices have made all of this possible. Her unwavering belief in my abilities and her constant encouragement have been my largest source of strength. I owe much of my success to her, as she has always been my biggest supporter, inspiring me to pursue my dreams with determination and resilience. Her guidance, patience, and unconditional love have been my greatest blessings. This thesis is completely dedicated to my mother, who has been my guiding light and my primary inspiration.

Abstract

Due to the large increase of Internet of Things devices in the last years, the cloud has been proved unsuitable to deal with the high demand of requests generated by them. To deal with this, fog computing was proposed to provide closer computing services in a distributed manner, acting as a middle layer between Internet of Things devices and the cloud. However, these devices have small battery capacity and low computational power. Therefore, strategies to allocate resources efficiently in the form of requests, are of particular interest. These requests may have short deadlines and other requirements that need to be ensured, usually in the form of Quality of Service. The most common problem is known as the Service Placement problem, where the goal is to offload a set of services while minimising some metrics of interest and ensuring that the users' requirements are met.

Another relevant problem is the Virtual Network Embedding problem, which emerges naturally in Network Virtualisation and Network Slicing. In these paradigms, the goal is to virtualise resources in order to allow isolation and easier connectivity between physical devices with an abstraction layer. Thus, there is a need to allocate a set of Virtual Networks efficiently, according to some metrics of interest and constraints.

When the topology of the network changes or users' move throughout the network, Reconfiguration and Migration problems emerge in fog environments. Not only users may move constantly, making it necessary to ensure constant connectivity, but devices in the network may be unavailable, due to battery depletion or overusage. In this scenario, there is a need to ensure a relocation of paths and resources in order to ensure suitable Quality of Service.

In this thesis, the goal is to tackle the following question: how to model and solve these hard resource allocation problems? For each of the previously described problems, a Mixed Integer Linear Programming formulation and heuristic are developed with multiple metrics to optimise and several associated constraints. Since the problems are interconnected, a framework joining all three types of problems is proposed as well. To evaluate each of these approaches, simulation and an in-depth experimental analysis is performed. The heuristics are compared to state-of-the-art approaches. The computational results show that our heuristics are close to the optimal value and better than known state-of-the-art algorithms.

Keywords: Fog Computing; Resource Allocation; Multi-objective Optimisation; Internet of Things; Quality of Service.

Resumo

Devido ao grande aumento de dispositivos da Internet das Coisas nos últimos anos, a núvem tem-se revelado inadequada para lidar com o elevado número de pedidos gerados por esses dispositivos. Para resolver este problema, foi proposta a computação em névoa, que fornece serviços de computação mais próximos de forma distribuída, atuando como uma camada intermédia entre os dispositivos da Internet das Coisas e a núvem. No entanto, estes dispositivos possuem baterias com capacidade pequena e baixo poder computacional. Por conseguinte, estratégias para distribuir os serviços solicitados, agrupados num conjunto de aplicações, são de particular interesse. Estas aplicações podem ter prazos curtos, que precisam de ser cumpridos, e outros requisitos de interesse que precisam de ser assegurados, geralmente na forma de Qualidade de Serviço. Assim, existe a necessidade de resolver problemas de Colocação de Serviços com diferentes métricas e restrições.

Devido à heterogeneidade dos dispositivos em ambientes em névoa, a Virtualização de Redes surgiu naturalmente. Neste paradigma, o objetivo é virtualizar recursos para permitir isolamento e facilitar a conectividade entre dispositivos físicos, utilizando software como uma camada de abstração. Um passo além é o Fatiamento de Redes, onde a própria rede é virtualizada para proporcionar segurança, uso eficiente de recursos e facilitar a configuração dos dispositivos. Ambos os paradigmas necessitam de alocar um conjunto de Redes Virtuais de forma eficiente, de acordo com métricas de interesse.

A Reconfiguração e Migração é um problema comum existente em ambientes de computação em névoa. Não só os utilizadores podem mover-se constantemente, tornando necessário garantir uma conectividade contínua, mas também os dispositivos na rede podem ficar indisponíveis, devido ao esgotamento da bateria ou ao uso excessivo. Neste cenário, é necessário garantir uma realocação de caminhos e recursos para assegurar uma Qualidade de Serviço adequada.

Nesta tese, o objetivo é abordar a seguinte questão: como modelar e resolver estes difíceis problemas de alocação de recursos? Para cada um dos problemas descritos anteriormente, é desenvolvida uma formulação de Programação Linear Inteira Mista, uma heurística com múltiplas métricas a otimizar e várias restrições associadas. Dado que os problemas estão interligados, é também proposta uma framework que junta os três tipos de problemas. Para avaliar cada uma destas abordagens, é realizada uma análise experimental e uma simulação. As heurísticas são comparadas com as abordagens do estado da arte. As soluções obtidas são próximas do valor ótimo e melhores do que os algoritmos do estado da arte.

Palavras-chave: Computação em Névoa; Alocação de Recursos; Otimização Multi-objetivo; Internet das Coisas; Qualidade de Serviço.

Contents

Acknowledgments xi

Abstract xiii

Resumo xv

List of Figures xxii

List of Algorithms xxiii

List of Tables xxvi

Acronyms xxvii

1. Introduction 1
- 1.1. Motivation . . . 1
- 1.2. Problem Statement and Contributions . . . 2
- 1.3. Scientific Outcomes . . . 3
- 1.4. Projects and Grants . . . 4
- 1.5. Outline of the Document . . . 6

2. Concepts Definition 9
- 2.1. Cloud-Fog Continuum . . . 10
  - 2.1.1. Fog Computing . . . 10
  - 2.1.2. Network Virtualisation . . . 11
  - 2.1.3. Network Slicing . . . 12
- 2.2. Optimisation Problems in Fog Computing . . . 12
  - 2.2.1. Metrics . . . 13
  - 2.2.2. Routing and Flows . . . 13
  - 2.2.3. Service Placement . . . 14
  - 2.2.4. Virtual Network Embedding . . . 15
  - 2.2.5. Reconfiguration and Migration . . . 16
- 2.3. Fairness . . . 17
- 2.4. Optimisation Formulation and Techniques . . . 17
  - 2.4.1. Multi-objective Optimisation . . . 18
  - 2.4.2. Integer Linear Programming . . . 21
  - 2.4.3. Heuristics . . . 22
- 2.5. Summary . . . 23

3. Related Work 25
- 3.1. Service Placement . . . 25

3.2. Virtual Network Embedding 33
3.3. Reconfiguration and Migration 39
3.4. Analysis and Open Issues 47

4. Service Placement **51**
4.1. Description 51
4.2. Mixed Integer Linear Programming Formulation 53
4.3. A Heuristic 55
4.4. Experimental Results 56
4.4.1. Setup 56
4.4.2. Experimental Analysis 58
4.5. Summary 61

5. Reconfiguration and Migration **63**
5.1. Description 63
5.2. Mixed Integer Linear Programming Formulation 65
5.3. A Heuristic 69
5.4. Experimental Results 71
5.4.1. Setup 71
5.4.2. Experimental Analysis 72
5.5. Summary 77

6. Virtual Network Embedding **79**
6.1. Description 79
6.2. Mixed Integer Linear Programming Formulation 80
6.3. A Heuristic 85
6.4. Experimental Results 87
6.4.1. Setup 87
6.4.2. Experimental Analysis 88
6.5. Summary 94

7. Hybrid Approach **97**
7.1. Description 98
7.2. Mixed Integer Linear Programming Formulations 100
7.2.1. Virtual Network Embedding 103
7.2.2. Service Placement 104
7.3. Heuristics 107
7.3.1. Virtual Network Embedding (Offline) 108
7.3.2. Virtual Network Embedding (Online) 109
7.3.3. Virtual Network Reconfiguration 110
7.3.4. Service Placement (Offline) 110
7.3.5. Service Placement (Online) 112
7.4. Experimental Results 112
7.4.1. Setup 113
7.4.2. Experimental Analysis 115
7.5. Summary 126

8. Conclusions and Future Work **127**

Bibliography **133**

A. Linearisation of the Mixed Integer Programming Formulations **143**
A.1. Chapter 4 143
A.2. Chapter 5 144
A.3. Chapter 6 145
A.4. Chapter 7 147

List of Figures

2.1. Virtual Network request on a Substrate Network 15
2.2. Network with one user 16
2.3. Graph Coimbra-Lisbon (based on Ehrgott [2005]). Each vertex corresponds to a city, each edge has the value of waiting time and cost, respectively. 18
2.4. Weighted-sum solution ($w_1 = 0.4, w_2 = 0.6$) 20
2.5. A posteriori (all Pareto optimal solutions are considered) 20
2.6. Min-max solution (trade-off between objectives) 21

4.1. Fog-Cloud Architecture 52
4.2. Time needed to execute the applications with different deadlines 57
4.3. Objective function values, formulation and heuristic, with high deadline 58
4.4. Objective function values, lexicographic solutions, with high deadline 59

5.1. Network with one user 64
5.2. Values of energy obtained by the Mixed Integer Linear Programming formulation, heuristic and baseline, with 7-10 nodes and 1 to 2 users 72
5.3. Values of latency obtained by the Mixed Integer Linear Programming formulation, heuristic and baseline, with 7-10 nodes and 1 to 2 users 73
5.4. Objective function values (Equation 5.7) obtained by the Mixed Integer Linear Programming formulation, heuristic and baseline, with 7-10 nodes and 1 to 2 users 74
5.5. Number of rejected users, 15, 30 and 50 nodes, 1 to 400 users . 75

6.1. Mixed Integer Linear Programming formulation and heuristic objective function values 88
6.2. Average energy consumption per node for different α values from 1 to 40 requests 89
6.3. Average normalised resource usage per node for different α values from 1 to 40 requests 90
6.4. Average energy consumption per node for current and future services from 3 to 60 requests 93
6.5. Average normalised resource usage per node for current and future services from 3 to 60 requests 93

7.1. Conceptualisation of the framework 98
7.2. Virtual Network request on a Substrate Network 100

7.3. Execution time (in s) of the Mixed Integer Linear Programming formulation . . . 102
7.4. Offline Virtual Network Embedding with topology size and number of users . . . 115
7.5. Online Virtual Network Embedding (randomised setting), average energy consumption per node . . . 116
7.6. Online Virtual Network Embedding (randomised setting), average bandwidth usage per link . . . 117
7.7. Online Virtual Network Embedding (referenced setting), average energy consumption per node . . . 117
7.8. Online Virtual Network Embedding (referenced setting), average bandwidth usage per link . . . 118
7.9. Comparison of random link failures on different Virtual Network Embeddings . . . 121
7.10. Comparison of random node failures on different Virtual Network Embeddings . . . 121
7.11. Offline Service Placement formulation and heuristic objective function values, 10 to 15 nodes, 1 and 2 applications and 2 services each . . . 122
7.12. Online Service Placement (randomised setting), overall energy consumption . . . 123
7.13. Online Service Placement (randomised setting), overall latency . 123
7.14. Online Service Placement (randomised setting), percentage of services sent to the cloud . . . 124
7.15. Online Service Placement (referenced setting), overall energy consumption . . . 124
7.16. Online Service Placement (referenced setting), overall latency . 125
7.17. Online Service Placement (referenced setting), percentage of services sent to the cloud . . . 125

List of Algorithms

4.1. Max fog usage with fairness heuristic 56

5.1. Single path data communication and migration heuristic 70
5.2. Path procedure for data communication and migration 70
5.3. Intersect procedure for two paths 71

6.1. Current+Future heuristic 86
6.2. Virtual Network Slicing heuristic 86

7.1. Static Virtual Network Embedding heuristic 108
7.2. Dynamic Virtual Network Embedding heuristic 109
7.3. Static Service Placement heuristic 111

List of Tables

2.1. All route combinations from Figure 2.3 and their respective objective values 19

3.1. Overview of Service Placement 32
3.2. Simulation in Service Placement 33
3.3. Overview of Virtual Network Embedding 40
3.4. Simulation in Virtual Network Embedding 40
3.5. Overview of Reconfiguration and Migration 48
3.6. Simulation in Reconfiguration and Migration 48

4.1. Parameters and variables for the Mixed Integer Linear Programming formulation 54
4.2. Type of services and their demands 57
4.3. Energy required regarding CPU usage, 15 applications (85 services) in percentage 60
4.4. Service distribution for 15 applications (85 services) in percentage (Number of services) 60

5.1. Parameters and variables for the Mixed Integer Linear Programming formulation 66
5.2. Parameter values in the network 71
5.3. Parameter values of the users 72
5.4. Execution times (in seconds) for the Mixed Integer Linear Programming formulation (individual objectives (E=Energy, L=Latency) and min-max (MM)), heuristic (H) and baseline (FF), with 7-10 nodes and 1 to 2 users 74
5.5. Execution times (in seconds) for the heuristic and baseline, with 15, 30 and 50 nodes and 1, 100, 200, 300 and 400 users 75
5.6. Number of hops with different topology sizes and link probabilities 76

6.1. Parameters and variables for the Mixed Integer Linear Programming formulation 81
6.2. Parameter values for fog nodes 87
6.3. Service Function Chaining requirements 87
6.4. Standard deviation of the average energy consumption for different values of α for the 40 requests 91
6.5. Standard deviation of the average normalised resource usage for different values of α for the 40 requests 92
6.6. Standard deviation of the average energy consumption for Current Only and Current + Future requests 94

6.7. Standard deviation of the average resource consumption for Current Only and Current + Future requests 95

7.1. Parameters and variables for the Mixed Integer Linear Programming formulations 101
7.2. Parameter values in the network (Foundation [2021]) 113
7.3. Parameter values of the users' Virtual Networks (randomized) . 113
7.4. Parameter values of the users' services (randomized) 113
7.5. Parameter values of the users' Virtual Networks (real scenario (Skarlat et al. [2017]; Navarro-Ortiz et al. [2020])) 114
7.6. Parameter values of the users' services (real scenario (Skarlat et al. [2017])) 114
7.7. Online Virtual Network Embedding, average standard deviation for the average energy consumption 119
7.8. Online Virtual Network Embedding, average standard deviation for the average bandwidth usage 120

Acronyms

5G	5th Generation
DM	Decision Maker
eMBB	enhanced Mobile Broadband
ILP	Integer Linear Programming
IoT	Internet of Things
LP	Linear Programming
MIP	Mixed Integer Programming
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
mMTC	massive Machine Type Communications
MOO	Multi-objective Optimisation
NFV	Network Function Virtualisation
NLP	Non-Linear Programming
NS	Network Slicing
NV	Network Virtualisation
QoS	Quality of Service
SDN	Software-Defined Networking
SFC	Service Function Chaining
SLA	Service Level Agreement
SP	Service Placement
URLLC	Ultra-Reliable Low-Latency Communications
VM	Virtual Machine
VN	Virtual Network
VNE	Virtual Network Embedding
VNF	Virtual Network Function
YAFS	Yet Another Fog Simulator

Chapter 1.

Introduction

Contents

1.1. Motivation **1**
1.2. Problem Statement and Contributions **2**
1.3. Scientific Outcomes **3**
1.4. Projects and Grants **4**
1.5. Outline of the Document **6**

This chapter aims to introduce the motivation, problem statement and contributions, scientific outcomes, projects and grants, as well as to provide an overview of the entire thesis.

1.1 Motivation

With the expected use of 5th Generation (5G) technology in the next few years and due to the high increase of Internet of Things (IoT) devices, fog computing is becoming a promising paradigm to deal with IoT demands (Hu et al. [2017]; Yi et al. [2015]; Mukherjee et al. [2018]; Mouradian et al. [2018]). Fog computing was first proposed by Bonomi et al. [2012]. This paradigm aims to handle the high increase of IoT devices and to focus on additional features, such as mobility, that are not well-supported in cloud environments. Fog computing is composed of different devices, such as local servers placed in strategic locations by Internet Service Providers, IoT devices, routers, and switches, called *fog nodes*, which receive data offloaded from user devices to compute or store temporarily services or information. Depending on the scenario, it is possible to have several IoT devices running simultaneously both in the user and in the fog domain.

Resource allocation problems are widely found on networking (Fischer et al. [2013]; Sonkoly et al. [2021]). This set of problems arise when there is a need to offload resource into the network to ensure some metrics of interest. The most common resource allocation problem is the Service Placement (SP) Problem (Huang et al. [2009]), which is a combination of routing and task offloading. The goal is to find a path between a user and a specific device on the network

and to decide where a set of resources should be allocated within the path. The most relevant metrics of interest in fog computing are energy and latency, since there is a need to ensure Quality of Service (QoS) and to decrease the energy consumption due to limitations, such as battery powered devices. Furthermore, it is necessary to avoid overusage of devices, since their computational capabilities are lower when compared to the cloud.

Due to the heterogeneity of devices in fog environments, there is a need to virtualise resources to allow isolation and easier connectivity between physical devices by software as an abstract layer. Thus, Network Virtualisation (NV) (Mijumbi et al. [2016]) has been proposed to decouple hardware and software. Each network function can be thought as a set of virtualised resources to execute a certain service. A generalised example of a network function is a Virtual Machine (VM). Thus, when there are a set of services to be allocated to each user, a Virtual Network (VN) is of interest. In this case, each network function, called *virtual node* hereafter, has a set of virtual links to connect these functions. Each link in a substrate network may be mapped to several VN links, such that the sum of bandwidth capacity of these virtual links does not exceed the capacity of the substrate link. The same applies to the substrate and virtual nodes. A step further is Network Slicing (NS), where not only resources are virtualised, but the whole network. A division through logical layers is performed in order to improve security, resource usage, and configuration of the network. However, several issues emerge from these paradigms, such as how to map virtual components to the substrate components (Virtual Network Embedding (VNE) problem (Fischer et al. [2013])) and how to deal with failures on the substrate network such that reliability is improved.

Other concerns when dealing with resource allocation in fog environments are the occurrence of failures on the network and unreachability of devices. Reconfiguration (commonly used in VNE) and migration (commonly used in SP) problems deal with these scenarios. The goal is to change the location of paths and resources in such a way that QoS is not affected and to be completely transparent to the users when these issues occur.

From the described problems, a question arises: how to model and solve these hard resource allocation problems? The next section provides the problem statement and the contributions developed to solve the problems above.

1.2 Problem Statement and Contributions

Resource allocation problems are usually tackled as optimisation problems, where several conflicting objectives arise, such as energy and latency, in the context of networking. Thus, techniques that allow us to tackle these issues and allow distribution of resources according to specified metrics are very relevant. Chapter 3 shows that most existing works do not take both views into consideration.

The goal of this thesis is to tackle resource allocation problems in fog environments from two perspectives: we consider fairness in order to distribute resources

according to a metric of interest. Fairness is considered as a model to minimise the maximum deviation of a given metric, i.e., to limit the overusage of devices. Moreover, we consider optimisation models that take into account multiple objectives simultaneously. In particular, the focus is on the notion of compromise solution, which is a trade-off solution between objectives without any explicit preference. In order to solve these problems, we formulate them as Mixed Integer Linear Programming models and develop heuristics to solve them.

My contributions are focused on four problems:

1. Service Placement (Chapter 4): A Mixed Integer Linear Programming formulation and a heuristic were developed to maximise resource usage and fairness of energy between devices. The goal is to use these models when a user requires offloading to the network.
2. Reconfiguration (Chapter 5): A Mixed Integer Linear Programming formulation and a heuristic were developed to maximise fairness of energy and latency between devices. The goal is to use these models when a user is disconnected from the network due to unreachability or overloading.
3. Virtual Network Embedding (Chapter 6): A Mixed Integer Linear Programming formulation and a heuristic were developed to maximise fairness of energy and interference of resources/network slices in different time-windows. The goal is to allow users to obtain resources in the form of a Virtual Network.
4. Hybrid (Chapter 7): Several Mixed Integer Linear Programming formulations and several heuristics were developed to maximise fairness of energy, latency, and bandwidth between devices. The goal is to provide a framework that optimises Virtual Network Embedding, reconfiguration of Virtual Networks in case of failures, and Service Placement optimisation in each individual Virtual Network, such that the whole process is transparent to users.

For each of the four problems, a simulation and an experimental analysis of the proposed solution methods were conducted. The empirical results suggest that the proposed models are able to decrease the overusage of devices while providing compromise solutions. When evaluating the heuristics, the solutions obtained are very close to the optimal, while taking a few milliseconds of running time.

1.3 Scientific Outcomes

The contributions of this thesis resulted in several publications. The main journals and conferences are as follows:

- N. Godinho, H. Silva, M. Curado and L. Paquete, "A reconfigurable resource management framework for fog environments," Future Generation Computer Systems, 2022, 133: 124-140 (Q1).

- N. Godinho, H. Silva, M. Curado and L. Paquete, "Energy and Latency-aware Resource Reconfiguration in Fog Environments," IEEE 19th International Symposium on Network Computing and Applications (NCA), 2020, pp. 1-8 (CORE A).
- N. Godinho, M. Curado and L. Paquete, "Optimization of Service Placement with Fairness," IEEE Symposium on Computers and Communications (ISCC), 2019, pp. 1-6 (CORE B).

The following posters have also been published:

- N. Godinho, M. Curado and L. Paquete, "Migration and Routing of Resources in Fog Computing", Encontro Ciência '22, 2022 (Unranked).
- N. Godinho, M. Curado and L. Paquete, "Optimizing Resource Allocation in Fog Computing", Encontro Ciência '21, 2021 (Unranked).
- N. Godinho, M. Curado and L. Paquete, "Optimization of Service Placement with Fairness," COST Action 15140 Training School, 2019 (Unranked).

Moreover, the following collaborations applied to practical and realistic scenarios were performed:

- N. Godinho, P. Duarte, P. Martins, D. P. Abreu, R. F. D. Barbosa, B. M. F. Mendes, J. Fonseca, M. Silva, M. Araujo, J. D. Silva, K. Velasquez, B. M. Sousa, M. Curado and A. A. Goes, "OREOS: Demonstrating E2E Orchestration in 5G Networks with Open-Source Components," IEEE International Conference on Computer Communications (INFOCOM), 2023, pp. 1-2 (CORE A*).
- H. M. S. Silva, N. Godinho and B. M. Sousa, "SALEM: Service Fairness in Wireless Mesh Environments," IEEE Consumer Communications & Networking Conference (CCNC), 2023, pp. 1-6 (CORE B).
- B. M. Sousa, H. M. S. Silva, N. Godinho and M. Curado, "Service Function Chaining in Wildfire Scenarios," IEEE 7th International Conference on Network Softwarization (NetSoft), 2021, pp. 388-394 (CORE B).

Finally, educational and scientific contributions were also made out of the scope of this thesis, such as teaching, workshop organisation, attending summer schools and an M.Sc. thesis co-supervision of: H. M. S. Silva, "Fairness Centred Framework for the ONOS SDN Controller," Master Thesis dissertation, Faculty of Science and Technology of the University of Coimbra (2022).

1.4 Projects and Grants

The work detailed in this thesis was accomplished at the Networks, Communications and Security group of the Center for Informatics and Systems of the University of Coimbra (NCS-CISUC), within the context of the following projects and grants:

- **Ph.D. Grant**: Foundation for Science and Technology (FCT) grant DFA/BD/7293/2020 (duration: 01/2021 - 02/2023, doi: 10.54499/2020.07293.BD).
- **MobiWise**: From mobile sensing to mobility advising. MobiWise aimed to enhance mobility in the cities, both for commuters and for tourists, through the development of a 5G platform that encompasses an access infrastructure filled with sensors, people and vehicles. The project aimed to connect any sensor, person and vehicle, and used all possible information to improve the user mobility, through a complete network and services platform for an Internet of Things real deployment in a smart city. MobiWise was funded by COMPETE 2020, Portugal 2020 - POCI, European Union's ERDF (P2020 SAICTPAC/0011/2015).
- **OREOS**: Orchestration and Resource optimisation for rEliable and lOw-latency Services. The objective of OREOS project was to design and implement an end-to-end orchestration platform for provisioning and managing critical services (such as vehicle communications, electrical distribution networks or emergency communications from public or private entities) in the context of the fifth generation mobile communications, in order to leverage and expand the portfolio of products and services that Capgemini offers in these domains. OREOS was funded by COMPETE 2020, Portugal 2020 - POCI, European Union's ERDF (POCI/01/0247/FEDER/049029).
- **SNOB-5G**: Scalable Network Backhauling for 5G. The objective of the SNOB-5G project was to carry out the research and development of a self-optimised, intelligent and fault-tolerant wireless backhaul solution for 5G networks that empowers cities, as neutral hosts, in the 5G business, by promoting total connectivity with a high-bandwidth, capacity and latency requirements, capable of accommodating new and innovative urban services. SNOB-5G is co-financed by the European Regional Development Fund (ERDF) through the Operational Program for Competitiveness and Internationalisation (COMPETE 2020) and by the Portuguese Foundation for Science and Technology (FCT) under MIT Portugal Program (CENTRO-01-0247-FEDER-045929).
- **5G - Components and Services for 5G Networks**: With standardisation of 5G networks being promoted in the coming years by organisations like 3GPP, the opportunity arises for the Portuguese industry to align and become internationally competitive in different segments of this complex ecosystem. The implementation of this project allowed the design and integrated validation of a new set of products, able to become part and give services within the future 5G networks, collecting efforts from different technology companies in the telecommunications segment, addressing both B2B and B2C models. 5G - Components and Services for 5G Networks was funded by COMPETE 2020, Portugal 2020 - POCI, European Union's ERDF (POCI/01/0247/FEDER/024539).
- **MH-SDVanet**: Multihomed Software Defined Vehicular Networks. The

> MH-SDVanet project proposes and implements an SDN-based architecture to enable rapid network innovation for multihoming 5G vehicular communications. The SDN-based architecture will include the support to optimise the placement and migration of service(s)/function(s) between OBUs, RSUs and Cloud elements. The optimisation mechanisms will consider fairness on the network access with the specified requirements, velocity of vehicles, and the dynamics of VANETs like intermittent connectivity and the possibility of cooperation between vehicles. MH-SDVanet is funded by COMPETE 2020, Portugal 2020 - POCI, European Union's ERDF (PTDC/EEI-COM/5284/2020).

Currently, a formulation based on the knowledge obtained in Chapter 6 of this thesis is being developed for the SNOB-5G project. The goal is to distribute the resource allocated in Network Slices (instead of individual users) in a fair manner, such that the network is not overused and the Quality of Service is guaranteed.

1.5 Outline of the Document

This thesis is structured as follows.

Chapter 2 details the most important concepts used throughout this document. It starts by defining the Cloud-Fog Continuum, which specifies the network architecture used in this work, the associated technologies and the advantages of using them. Next, some optimisation problems that emerge in the Cloud-Fog Continuum are presented, followed by a section that introduces formulations and techniques used to solve the mentioned problems.

Chapter 3 describes several works from the literature for each of the problems considered on this thesis. Afterwards, an analysis focusing on different aspects, such as metrics of interest and scenarios, is performed, followed by the detailing of open questions.

Chapter 4 details a Mixed Integer Linear Programming and a heuristic to tackle Service Placement while considering several metrics of interest and fairness between devices, when a set of users requests computational resources to the network. An experimental analysis of the formulation and heuristic with different synthetic data is provided.

Chapter 5 deals with reconfiguration in mobility environments to decrease network failures, specially when the user disconnects from the network due to unreachability or when overloading happens. A Mixed Integer Linear Programming formulation and heuristic are developed, with various objectives to optimise and implemented into a simulator. Moreover, an experimental analysis is presented.

Chapter 6 goal is to allocate resources within Network Slicing-fog environments with different time requests. A Virtual Network is associated to each user and applied into a given slice with the aim to optimise several metrics. An experimental analysis with realistic request values is provided.

Chapter 7 takes into account several ideas from the previous chapters and presents a framework containing several techniques. It aims to provide resources to users, reconfigure to avoid network failures and offload as efficient as possible the users' services into their resources. A formulation and heuristic is provided for each problem, while taking into account both static and dynamic scenarios. Finally, an experimental analysis is developed to show the benefits of considering a joint framework.

Chapter 8 concludes this thesis and describes several directions for future work.

Chapter 2.

Concepts Definition

Contents

2.1. Cloud-Fog Continuum 10
2.1.1. Fog Computing 10
2.1.2. Network Virtualisation 11
2.1.3. Network Slicing 12
2.2. Optimisation Problems in Fog Computing 12
2.2.1. Metrics 13
2.2.2. Routing and Flows 13
2.2.3. Service Placement 14
2.2.4. Virtual Network Embedding 15
2.2.5. Reconfiguration and Migration 16
2.3. Fairness 17
2.4. Optimisation Formulation and Techniques 17
2.4.1. Multi-objective Optimisation 18
2.4.2. Integer Linear Programming 21
2.4.3. Heuristics 22
2.5. Summary 23

This chapter aims to detail the necessary concepts used throughout this thesis. Section 2.1, Cloud-Fog Continuum, focuses on the network architecture of this work, the associated technologies and the advantages of using them. Section 2.2, Optimisation Problems in Fog Computing, describes some problems that naturally arise in the Cloud-Fog Continuum, its related technologies and why they are relevant, followed by the concept of Fairness in communications (Section 2.3). Next, Section 2.4, Optimisation Formulation and Techniques, aims to provide an overview of different techniques to deal with the described optimisation problems and how they can be solved. Finally, a summary of this chapter is provided in Section 2.5.

2.1 Cloud-Fog Continuum

In this section, fog computing and its connection to cloud computing and network virtualisation are described. Moreover, details regarding the benefits of using a Cloud-Fog Continuum, and an emerging technology named *Network Slicing (NS)*, which enables a better control of the network resources, are presented.

2.1.1 Fog Computing

Fog computing has several accepted definitions (Mukherjee et al. [2018]). In this thesis, fog computing is described as a set of heterogeneous devices that have communication and resource capabilities that improve the quality of connection and to decrease cloud usage by acting as a layer between the cloud and users. It can also mitigate issues that arise in connections between data centers and users' devices. Thus, it is possible to ensure Quality of Service (QoS) requirements of the users by decreasing latency (Hu et al. [2017]; Yi et al. [2015]). QoS is used to describe the overall performance of a given service, where it is measured quantitatively by observing some network aspects, such as availability, latency, and throughput, to allow services with different priorities to meet the required level of performance. For example, voice communications have different latency requirements than streaming communications.

A basic network architecture that represents the Cloud-Fog Continuum is described by Hu et al. [2017]. This architecture is composed by cloud, fog, and users. First, the cloud layer contains high-performance servers, effective for large computational services and storage capacity. In this layer, all the information is saved permanently, contrarily to the fog layer, where it is only saved temporarily and in low quantities. Considering its large computational power, it makes possible the realisation of data analytics, relevant for discovering useful information and supporting decision-making. It is also distributed by using several clusters of data centers throughout the planet. Next, the fog layer extends the cloud layer in a decentralised manner to provide more coverage, making it useful to devices that need constant geo-localisation. It can be seen as a network of gateways covering a larger area, where these gateways redirect traffic, process data, and store temporarily data. Finally, the user layer is composed of Internet of Things (IoT) devices, mobile devices, such as smartphones, and other resource-constrained devices. Some of these devices are real-time, which need a time-window answer, and offloading of services is also desired. A few of the essential features of fog computing are highlighted as follows: to partition geographical areas, to provide location awareness, to support end-device mobility, to provide real-time applications, to support different types of IoT devices, to provide access to the set of end-devices and computing nodes through the wireless mode, and to collaboratively process a massive number of user services requests.

In this thesis, the focus is on *resource management* in the Cloud-Fog Continuum. There is a need to allocate resources efficiently according to a set of metrics of

interest within the different network layers. These resource problems also need to take into account routing in the network and reconfiguration and migration when failures occur. The next sections describe technologies that can be integrated within the Cloud-Fog Continuum to provide a robust solution for 5th Generation (5G) and beyond.

2.1.2 Network Virtualisation

In order to allow usage of the same physical resource across several users, Network Virtualisation (NV) has been proposed (Chowdhury and Boutaba [2010]). The goal is to combine network hardware and network resources into a network software module, a Virtual Network (VN). Thus, the main objectives are to decouple software and hardware and to allow flexible network function deployment. It can be considered as a whole network, with virtual links and nodes, or it can be separated into chunks according to their usage. In the latter case, there is a set of Virtual Network Functions (VNFs) that need to be allocated, and a routing algorithm must provide a path between them.

By using NV, it is possible to better manage the different devices on the network, and to improve the distribution of resources. In order to make this possible, there is a separation between the substrate network (the physical network) and the VN (the logical network). Each link in a substrate network may be mapped to several VN links, such that the sum of bandwidth capacity of these virtual links does not exceed the capacity of the substrate link. The same applies to the substrate and virtual nodes. By using NV, it is possible to ensure that each VN has only access to its own resources, since it is only aware of its virtual components.

While the usage of NV has increased in the last years, the development is still at an early stage, with many open questions. As Internet Service Providers look at the details of implementing NV and accomplishing its foreseen goals, there are concerns about the usage of some of these goals and whether implementation translates to the benefits initially expected. There are important unexplored research challenges such as testing and validation, resource management, interoperability, instantiation, performance of VNFs, and others that should be addressed.

One possible approach to implement NV is Software-Defined Networking (SDN) (Bizanis and Kuipers [2016]; Xia et al. [2015]). With SDN it is possible to separate the routing process from the forwarding process. The objective is to turn the routing process programmable, in order to allow communication. A simple reference model is used in SDN, which consists of *infrastructure layer, control layer* and *application layer*. The infrastructure layer consists of switches, routers and other similar devices that are responsible to monitor the network status, and to process network packets over a set of rules provided by the controller. The control layer acts as an intermediate layer for the application layer, where a series of functions for the upper layer are defined to obtain the network status, change the forwarding rules or tune the decision associated to the system. The application layer accesses the control layer by an Application Programming Interface

and contains the applications that accomplish the user requirements. Hence, it is possible to access different devices by the same Application Programming Interface to provide easier configuration, to improve performance globally, and to encourage the integration of new modules.

Since there are different network devices in the Cloud-Fog Continuum, it is necessary to ensure that these devices communicate correctly, in order to provide a seamless experience.

2.1.3 Network Slicing

Since services have their own set of requirements and priorities, Network Slicing emerged naturally after Network Virtualisation was proposed (Afolabi et al. [2018]). NS has been recently gaining momentum among an ever-growing community of researchers from both academia and industry. It has been also the focus of different standardisation bodies (e.g., 3GPP (3rd Generation Partnership Project), IETF (Internet Engineering Task Force) and ITU-T (International Telecommunication Union - Telecommunication Standardisation Sector)). The possibility to create, on demand and in a programmable fashion, cost-efficient end-to-end network slices and dedicate them for the dynamic provisioning of diverse services is seen as an important feature of 5G. In this vein, efforts are ongoing towards developing a 5G mobile system capable of deploying network slices of varying sizes and structures.

NS is a managed group of subsets of resources, network functions, control, management/orchestration planes and services at a given time, divided in *slices*. It consists of 3 layers: 1) Service Instance Layer, 2) Network Slice Instance Layer, and 3) Resource layer. Each slice acts as a logical network that may contain fog and cloud devices, depending on the requirements of the services to be allocated in the slice. This allows a higher level of heterogeneity, and the Internet Service Provider has only access to its own layer, increasing security. In several 5G standardisation initiatives, such as 3GPP, ITU, 5G PPP and NGMN, three different types of slices are considered (Barakabitze et al. [2020]): *enhanced Mobile Broadband (eMBB)*, where there is a need to ensure large bandwidth requirements, *Ultra-Reliable Low-Latency Communications (URLLC)*, where the focus is to ensure low latency and reliability, and *massive Machine Type Communications (mMTC)*, where there is a high number of users, yet their requirements are lower than the other two types of slices.

By using NS, future generation networks are able to provide a better distribution of resources, due to the ease of creating policies that allow a higher control of the existing services according to their requirements.

2.2 Optimisation Problems in Fog Computing

In this section, several optimisation problems that arise within the Cloud-Fog Continuum are introduced. First, the most common objectives found in fog environments are discussed. Next, several optimisation problems that will be

considered in this thesis are described: the routing problem, the Service Placement (SP) problem and the Virtual Network Embedding (VNE) problem. Finally, Reconfiguration and Migration problems emerge, which are usually found in unstable networks or when users regularly move.

2.2.1 Metrics

In the context of fog computing, different objectives (or metrics) and requirements are of interest, for the problems of Service Placement (SP), Virtual Network Embedding (VNE), and reconfiguration and migration. We focus on the following objectives:

- *Energy consumption:* Minimisation of energy consumption of the users that have resource-constrained devices or the fog nodes themselves, since the fog layer may be composed of small devices without the possibility of charging.
- *Latency/delay:* Minimisation of communication delay. Users may have (near) real-time systems that need responses in defined intervals of time.
- *Resource usage:* Maximisation of usage of computational resources in the fog nodes, which implies minimisation of cloud usage.
- *Throughput:* Maximisation of throughput. Usually, this is formalised as the data-rate of a link or the used bandwidth by a given service.
- *Interference:* Minimisation of resource interference between devices. If too many virtual resources are used in the same physical device, degradation may occur. In the case of NS, slice interference is also considered.

Note that some of these objectives may be considered as constraints. For example, in NS environments, some slices need to ensure an upper bound on the communication delay.

2.2.2 Routing and Flows

One of the main problems in computer communications is routing (Medhi and Ramasamy [2007]). Considering a set of generic devices and links connecting them, this problem aims to minimise the cost of communication between two different devices, usually called source and destination.

A network can be modelled as a graph, where each device corresponds to a vertex (or node) and the connection between each pair of devices (be it wired or wireless) is modelled as an arc (or edge) (see Ravindra K. Ahuja [1993]). The traversal of each arc may have a cost (or weight) and the set of arcs connecting a source and destination vertices corresponds to a path. When the goal is to find the path that minimises the cost between two vertices, we have the single-source shortest path problem. However, this problem cannot ensure feasible solutions when there are demand constraints, i.e., when there is a need to send an amount of information between the source and destination, yet the capacity of each arc

cannot be surpassed. In the context of network communications, that capacity is defined as bandwidth.

The category of problems that include capacity arcs are network flow problems (Ravindra K. Ahuja [1993]). The simplest and most well-known problem is the maximum flow, where the aim is to maximise the amount of flow (or information) sent through the arcs while ensuring that the capacity at each arc is not surpassed. A visual example is to imagine a set of water pipes. Each pipe has a certain diameter, the capacity, and anywhere where the pipes meet, the total amount of water arriving must be equal to the amount going out. The entrance of water corresponds to the source and the faucet corresponds to the destination. An extension to this problem is the minimum cost flow, where the goal is to minimise the cost of sending all the required flow from source to destination. In communication networks, this may be visualised as a certain device sending a set of data packets that need to reach the destination device, but each link has a limit on the data packets that are accepted in any given time.

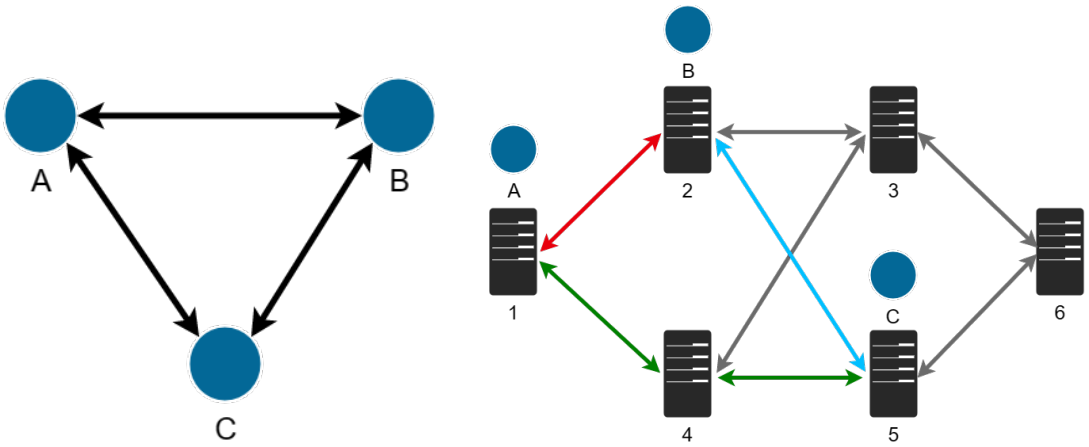
In realistic environments, it is necessary to ensure several services at any time, with multiple sources and destinations. This can easily be understood by considering a set of users with different requirements, such as streaming of a video and data communications. Each user may be located in a different part of the network and the necessary contents may be in different servers. This problem is called the multicommodity flow problem (Ravindra K. Ahuja [1993]; Pióro and Medhi [2004]). The multicommodity flow problem is a generalisation of both maximum flow and minimum cost flow problems. While the maximum flow and minimum cost flow problems can be solved in polynomial time, the multicommodity flow problem is NP-hard (Ravindra K. Ahuja [1993]).

Finally, there are two types of flows: splittable and non-splittable (Pióro and Medhi [2004]). In the splittable case, the flow can be divided through different links. An example would be to send data packets through different routes and join them in the destination device when all packets reach it (multi-path). In the non-splittable case, the required flow for each source-destination pair must be ensured by all used links. Thus, it only follows a single path to reach the destination device.

In this thesis, we consider minimum cost multicommodity flows with non-splittable flows. As it will be possible to see in the next sections, this problem can be further extended to consider node and other link constraints, on top of the link bandwidth constraints.

2.2.3 Service Placement

The problem of Service Placement (SP) aims to allocate (or offload) a set of resources into the network, with the goal of minimising some metric of interest, such as latency and cost (Huang et al. [2009]). This problem can be partitioned in two parts: allocate the resources in a given device of the network (task offloading) and define which path is used from the user to that device (routing). While the simplest version of the problem with no resource constraints is easy to solve,



(a) Virtual Network Request (b) Substrate Network

Figure 2.1.: Virtual Network request on a Substrate Network

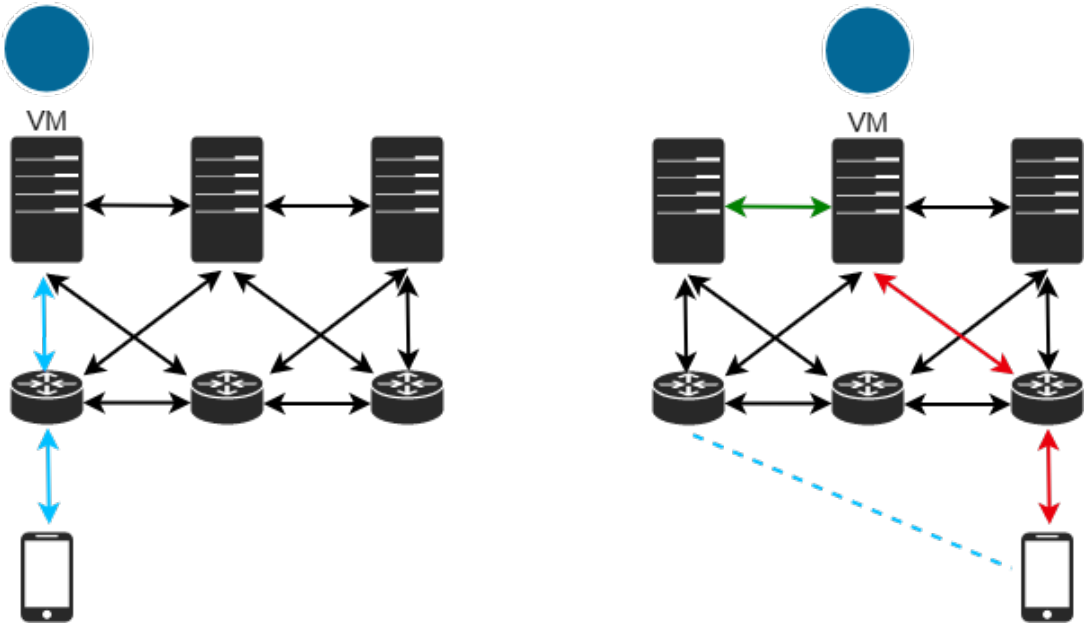
the version that includes constraints is proven to be NP-complete (Huang et al. [2009]). Developing efficient methods to solve the SP problem is particularly relevant when considering the fact that some devices have limited capabilities and battery, such in fog environments.

In this thesis, we model the Service Placement (SP) problem with multiple users and extra limitations, such as deadline and scheduling constraints.

2.2.4 Virtual Network Embedding

A similar resource allocation problem is the Virtual Network Embedding (VNE) problem (Fischer et al. [2013]; Laghrissi and Taleb [2018]). Instead of aiming to offload a set of services for each user, the goal is to allocate a previously defined Virtual Network (VN) into the physical/substrate network. Here, instead of having to ensure a path, it is necessary to provide a set of paths to allow the user to have the necessary capabilities to perform any type of offloading, while being isolated from the rest of the network. Similarly to the SP problem, it is usually divided into two parts: virtual link embedding and virtual node embedding. The virtual network embedding problem can be formulated as a multicommodity flow problem with dynamic transit nodes, i.e., each node may be a source, target or transit node, whereas the virtual node embedding can be formulated as the task offloading problem. The main different between VNE and SP is the fact that a topology must be allocated instead of a set of services associated to a user. This means that while in the SP problem, the source and target nodes are defined a priori, in the VNE problem they are usually not fixed.

Figure 2.1(a) illustrates a possible VN request from a user, with a set of virtual nodes and links, while Figure 2.1(b) represents the substrate network and the corresponding mapping of the VN onto the network. For each link on the VN it is necessary to decide which link(s) on the substrate network are chosen. For this, we need to take into account the resources of the nodes and links, since we assume that not only the link is mapped, but the associated nodes as well.



(a) Before Mobility Occurs (b) After Mobility Occurs

Figure 2.2.: Network with one user

Moreover, we need to take into account connectivity, i.e., to ensure that the existing paths on the VN are mapped correctly on the substrate network. In the example of Figure 2.1(b), $A-B$ is mapped into $1-2$ (red), $A-C$ is mapped into $1-4-5$ (green) and $B-C$ is mapped into $2-5$ (blue). The latency of $A-C$ is the sum of the latency of both links on the substrate network and both links have to ensure that the requirements of the corresponding link on the VN are met.

2.2.5 Reconfiguration and Migration

In the problems above, it is assumed that the state of the network does not change. For example, in a static network, the existing resources, devices and connections are always the same. It occurs in some scenarios, such as large events where dedicated devices are used to ensure more reliable connections, to stream and communicate. Yet, in most applications, the state of the network changes. Errors may occur, devices may crash, even connections may be temporarily unavailable. Thus, there is a need to consider dynamic networks. When dealing with realistic environments, there is a need to ensure reconfiguration and/or migration of resources. Therefore, the goal is to reallocate a set of resources while satisfying the required constraints (Pozza et al. [2020]; Gao and Rouskas [2018]), i.e., to identify a set of links/nodes that ensure the QoS requirements and optimise one or more metrics of interest. Usually, the reconfiguration term is used to describe movement of required resources to a new location, while the migration term is used to describe the need to move a service (and the data) to a new location. Thus, in this thesis, reconfiguration will be used in the context

of VNs, while migration will be applied to SP[1].

One approach to deal with network failures is to create disjoint backup resources. This allows fast reconfiguration/migration when the failure occurs. However it is not efficient due to the need to allocate twice the resources for each VN, which causes underusage of the network. Thus, there is a need for fast reconfiguration/migration approaches to decide where the necessary resources will be allocated and which path and devices are used to reconfigure/migrate the data. Figure 2.2 shows a migration example. We assume that each user (US) has its own VN previously allocated with some VN mapping algorithm and is connected to the closest device (GW) that routes the data (Figure 2.2(a), blue lines). In this case, the user has moved to a new location and cannot connect to the previous device or the latency is too high, and the connection cannot be established. In Figure 2.2(b), the blue dotted line indicates the previous connection. After connecting to a new GW, the reconfiguration of data communication (green lines) and migration (red lines) is performed.

2.3 Fairness

Fairness arises as a topic of interest in many research areas, such as optimisation and decision-making. Fairness, in a communications perspective, means to provide an equal sharing of resources with respect to devices/services (SHI et al. [2014]).

There are several issues that emerge in a networking scenario. Each device should be able to get a fair chance to access resources, bandwidth should be fairly distributed, QoS requirements should be fairly satisfied, and energy consumption should be proportionally distributed. In networking, several techniques have been used to deal with fairness (Nace and Pioro [2008]; SHI et al. [2014]).

Researchers have proposed fairness metrics to help identify underused communication channels or underused resources in devices (Jain et al. [1984]). Fairness can be adapted to consider different metrics, such as latency or energy consumption. We are particularly interested on fairness optimisation models which allow to distribute the resources as equally as possible.

In the context of this thesis, the goal is to improve the resources of the devices that get the least amount of resource or to avoid overusage of devices.

2.4 Optimisation Formulation and Techniques

This section aims to provide a formal and mathematical description of methods to solve the problems described in Section 2.2. It starts by introducing the main concepts of multi-objective optimisation and different techniques used to tackle this problem, followed by the min-max criterion, which is used throughout this thesis. Next, a description of the basics of Integer Linear Programming (ILP)

[1]Sometimes, in the state-of-the-art, migration is also used when reconfiguring the whole VN, yet we will still describe that case as reconfiguration to avoid confusion.

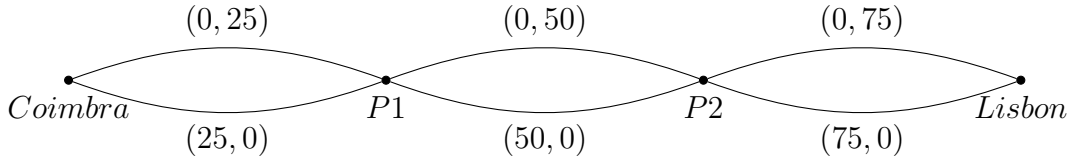


Figure 2.3.: Graph Coimbra-Lisbon (based on Ehrgott [2005]). Each vertex corresponds to a city, each edge has the value of waiting time and cost, respectively.

formulation is performed. Finally, the concept of heuristics is presented, which obtains near-optimal solutions in realistic scenarios.

2.4.1 Multi-objective Optimisation

An optimisation problem consists of a set of feasible solutions, an objective function and a set of constraints. An objective function f is a function to be minimised or maximised over the set X of feasible solutions, and X is defined by the set of constraints of the problem. A combinatorial optimisation problem is an optimisation problem with discrete variables and a finite set of feasible solutions (H. Papadimitrou and Steiglitz [1998]).

In Multi-objective Optimisation (MOO) problems, there are two or more objectives functions to be optimised simultaneously (Ehrgott [2005]; Miettinen [1998]). These problems arise in many applications, such as the shortest path problem with two or more objectives (described in Section 2.2.2). The solution to these problems depends of the preferences, or lack of, of a Decision Maker (DM). This section describes the main techniques used in MOO problems.

Hereinafter, the following formulation of MOO problems is considered:

$$\min_{x \in X} f(x) = (f_1(x), \ldots, f_k(x)) \tag{2.1}$$

where X denotes the set of feasible solutions and $f_1, \ldots, f_k$ represent k objective functions.

To understand a real application of MOO, imagine the following scenario. There is a need to travel from Coimbra to Lisbon by car and the following question arises: *which route should we take?* Without loss of generalisation, we can assume two types of roads: free and paid. The free road has a long waiting time until it can be traversed, while the paid has no waiting time, yet it is expensive. The reader can notice, then, that using only one type of road will be the ideal in one objective but the worst case in the other objective.

Let us model this problem as a graph, with the goal of finding the shortest path with two objectives to optimise, waiting time and cost. The goal is to find the lowest value path for both objectives simultaneously. Figure 2.3 shows a simplified graph of this scenario, where each vertex corresponds to a city and each edge has the value of time and cost. Table 2.1 provides the possible combinations of routes and their objective values. In most techniques, there is an

Table 2.1.: All route combinations from Figure 2.3 and their respective objective values

Route	**Objective values (waiting time, cost)**
free - free - free	(150, 0)
free - free - paid	(75, 75)
free - paid - free	(100, 50)
free - paid - paid	(25, 125)
paid - free - free	(125, 25)
paid - free - paid	(50, 100)
paid - paid - free	(75, 75)
paid - paid - paid	(0, 150)

assumption that a DM is available and that is able to express their preferences regarding the objectives.

When the DM is able to express their preferences beforehand, an *a priori* technique is used. For instance, the DM may be able to express the preferences beforehand based on weights (each objective has a weight), lexicographic ordering (the problem is solved for each objective, where each solution serves as a constraint) or thresholds (where the less preferred objectives are transformed into constraints) as goals to attain in the objective space. Thus, the MOO problem is said to be transformed into a single objective problem.

To exemplify, a weighted-sum technique is used by extending Equation 2.1. The following formulation is, then, obtained:

$$\min_{x \in X} f(x) = w_1 f_1(x) + \ldots + w_k f_k(x) \tag{2.2}$$

where w_k represents a weight for each objective and the sum of weights is, usually, 1.

Let us assume weights $w_1 = 0.6$ and $w_2 = 0.4$. The optimal solution, using this technique, is the route paid-paid-paid ($0.6 \times 0 + 0.4 \times 150 = 60$). If we invert the weights, then the optimal solution is free-free-free ($0.4 \times 150 + 0.6 \times 0 = 60$). Figure 2.4 shows the optimal solutions for each combination of weights. By doing several combinations of weights, it is possible to observe when $w_1 < 0.5$, the optimal solution will always be the one that uses only free roads, while $w_2 < 0.5$ always obtains paid roads only. If both weights are 0.5, then all solutions have the same objective values (75).

In the case of the problems tackled in this thesis, the scenario differs significantly, since the environment varies constantly. One combination of weights may not be the best in all scenarios and a DM may not be always available to express their preferences.

If the DM is not able to express their preferences, *a posteriori* technique is used. In this case, the goal is to generate the set of Pareto optimal solutions, also called Pareto set, or its image on the objective space, Pareto front. A solution $x \in X$

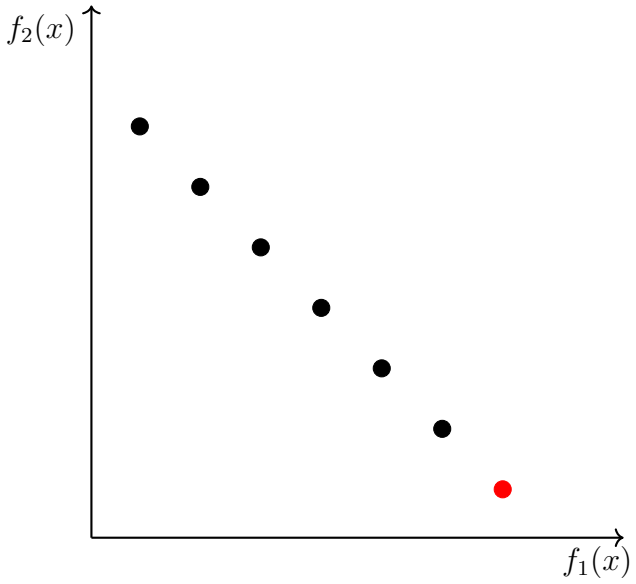


Figure 2.4.: Weighted-sum solution ($w_1 = 0.4, w_2 = 0.6$)

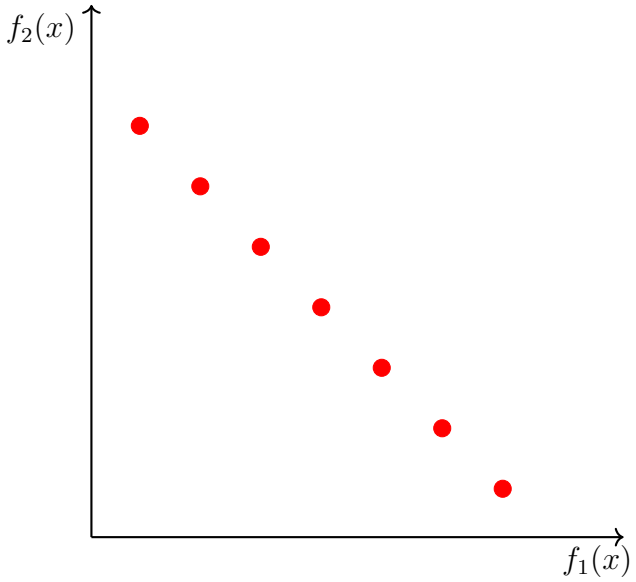


Figure 2.5.: A posteriori (all Pareto optimal solutions are considered)

is *Pareto optimal* if there is no other vector $x^* \in X$ such that $f_i(x^*) \leq f_i(x)$ for all $i = 1, \ldots, k$ and $f_j(x^*) < f_j(x)$ for at least one $j \in \{1, \ldots, k\}$. The Pareto set contains solutions that are equally relevant until the DM decides which solution better represents the preferences. Usually, the DM takes a decision by inspecting the Pareto set (or the Pareto front).

In our problem, all solutions are Pareto optimal. Figure 2.5 shows the Pareto optimal solutions. However, there might be too many Pareto optimal solutions to be found and there might not be a DM able to select the one to implement, under tight time requirements.

An *interactive* technique uses information provided by the DM during the optim-

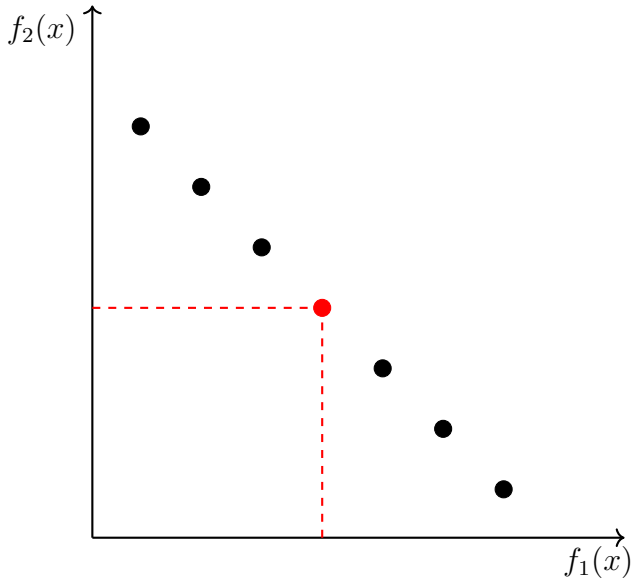


Figure 2.6.: Min-max solution (trade-off between objectives)

isation process. In this case, the DM has several interactions with the optimisation process, by defining desirable points in the objective space to be achieved. However, this technique implies several iterations, which is not feasible in a communications environment and therefore, it is out of the scope for this thesis. For further information, the reader can refer to Miettinen [1998].

A *min-max* formulation may be used, when the DM does not have any explicit preference. In this approach, the goal is to minimise the maximum value of all objectives (Aissi et al. [2009]). This allows us to obtain a compromise (or trade-off) solution for all objectives, making it preferable when the goal is to obtain a conservative decision to protect against worst-case scenarios. By extending Equation 2.1, the min-max formulation of an optimisation problem is formulated as follows:

$$\min_{x \in X} f(x) = \max(f_1(x), \ldots, f_k(x)) \tag{2.3}$$

In the shortest path example, the solution obtained by a min-max formulation is the free-free-paid route (or paid-paid-free, since they have the same objective values). Despite not being the best on any objective, is the one that minimises the maximum function value, with value 75; see red circle in Figure 2.6.

2.4.2 Integer Linear Programming

Mathematical formulations allow us to describe a problem in a mathematical language that can be implemented into a solver to generate an optimal solution. Linear Programming (LP) aims to minimise a linear cost function subject to one or more linear constraints (Bertsimas and Tsitsiklis [1997]). Equation 2.4 shows

the generic form of a LP formulation:

$$\begin{aligned} \min & \sum_{i=1}^{n} c_i x_i \\ s.t. & \sum_{i=1}^{n} A_i x_i = b_i \\ & x \geq 0 \end{aligned} \tag{2.4}$$

where $x = \{x_1, \ldots, x_n\}$ is the decision vector, $c = \{c_1, \ldots, c_n\}$ is the cost vector, $A = \{A_1, \ldots, A_n\}$ is the available resource vector and $b = \{b_1, \ldots, b_n\}$ is the target vector.

LP models are easier to solve than other models due to their linear equations and real variables (Williams [2013]). Unfortunately, many problems cannot be represented that way. One example is Integer Linear Programming (ILP) formulations. Equation 2.4 can represent an ILP formulation if instead of having real variables, integer variables are used. There exist several exact algorithms to solve these formulations, such as Branch-and-Bound, yet the execution time is not feasible in tight constraint environments. Moreover, rounding the solution of a LP model to an integer value may not give the optimal solution (sometimes not even a good solution) (Williams [2013]).

In some scenarios, we may be interested in having both real and integer variables, known as Mixed Integer Linear Programming (MILP) formulations. Nevertheless, these models continue to be hard to solve.

In many problems, obtaining an optimal solution would be the ideal. Yet, due to the complexity of most of these problems, it is not possible due to time constraints. Even a small instance on a problem with integer or non-linear variables can take hours to solve, since ILP formulations belong to the NP-hard class of problems. Specially, in dynamic and real-time scenarios, time is of most importance. However, modelling the problems into mathematical formulations may be useful to provide a baseline comparison to heuristic approaches and to provide bounds on what can be achieved.

2.4.3 Heuristics

Since there are several relevant problems belonging to the NP-hard class, we need strategies to obtain approximations to the optimal solutions in a short amount of time. Heuristic approaches have been widely used in optimisation and showed great performance with well-known NP-hard problems, such as the travelling salesman problem (Applegate et al. [2011]), from a set of cities, and return to the original city. The idea behind these approaches is the one of "trial and error". For example, we may use an initial solution, such as a randomly generated solution, and the algorithm tries to find better solutions for the problem by employing several perturbations. Usually, this is performed until defined execution time ends or the best solution does not improve after a defined number of iterations.

There are several strategies to deal with these problems (Vazirani [2004]; Cormen et al. [2009]):

1. *Approximation algorithms* are able to obtain solutions within an approximation ratio, calculated from the solution obtained, and a calculated upper bound over the optimal value. Moreover, they perform quickly. However, it is difficult to derive algorithms with a given approximation ratio for more complex problems.

2. *Greedy algorithms* are able to find solutions that are not guaranteed to be optimal, yet sufficient to obtain a satisfactory for the problem to be solved (Müller-Merbach [1981]). Their aim is to incrementally obtain a solution according to the current best possible candidate solution. In this case, the solution obtained will be deterministic, where each input always provides the same output. One example of this approach is Kruskal's algorithm to solve the minimum spanning tree problem, where the goal is to obtain a connected graph that connects all the vertices without cycles and minimises the total edge weight. This thesis will focus on this type of approach.

In the realm of MOO problems, metaheuristics are widely used. Metaheuristics refer to a set of approaches, such as genetic algorithms and simulated annealing, where the goal is to explore the search space to find near-optimal solutions. However, they are usually non-deterministic, i.e., the same input may provide different output.

In the case of fog computing, the devices used have less computational capabilities than the cloud, and the solutions need to be generated quickly. Therefore, greedy approaches are useful to solve MOO problems in fog environments and will be considered in this thesis.

2.5 Summary

This chapter described the most important concepts to be used throughout the document. It started by defining the Cloud-Fog Continuum, which presented fog computing acting as a middle layer between cloud and the users, the associated technologies, Network Virtualisation to provide heterogeneity and virtualisation of hardware, and Network Slicing to allow better configuration of services by their requirements with support of NV. Next, some optimisation problems that emerge in the Cloud-Fog Continuum were presented, namely routing to ensure connectivity, Service Placement to allow offloading of services, Virtual Network Embedding to provide a flow of resources to users and Reconfiguration and Migration when there are network failures or overusage of devices. Afterwards, a section to introduce formulations and techniques that are used to solve the mentioned problems was detailed. This included multi-objective optimisation and some existing techniques, the min-max criterion to obtain trade-off solutions between objectives, Integer Linear Programming formulations to model different problems and obtain optimal solutions, and heuristics to generate sub-optimal solutions in large environments.

Chapter 3.

Related Work

Contents

3.1. Service Placement **25**
3.2. Virtual Network Embedding **33**
3.3. Reconfiguration and Migration **39**
3.4. Analysis and Open Issues **47**

In this chapter, a literature review of some proposed approaches to solve Service Placement (SP) (Section 3.1), Virtual Network Embedding (VNE) (Section 3.2), and Reconfiguration and Migration (Section 3.3) in Fog Computing and related architectures is provided. Furthermore, an analysis and identification of open issues derived from the state-of-the-art works is performed (Section 3.4).

3.1 Service Placement

The Service Placement (SP) problem aims to allocate a service or Virtual Machine (VM) requested resources in the network. There are several variants of this problem, which may take into account different constraints, such as deadlines and scheduling. In this section, related works containing single and multi-objective versions of the problem are described.

Nishio et al. [2013] aimed to address a significant challenge in fog computing environments, specifically focusing on resource allocation within service-oriented applications. In their research, the authors concentrated on optimising two critical objectives: energy consumption and latency. To achieve this optimisation, the authors introduced a utility function that encapsulates both objectives, allowing them to be balanced according to the specific requirements of the application.

Jia et al. [2014] tackled the problem of task offloading in fog computing by developing a heuristic aimed at minimising the completion time of user services. The primary objective of their heuristic was to achieve a balance in the computing load across different services, which is essential for maintaining service

quality and avoiding bottlenecks. The authors conducted a thorough analysis of their proposed heuristic by testing it against two benchmarks, which provided a reference for evaluating its performance. Furthermore, they explored how various system parameters, such as network bandwidth and computational capacity, impact the performance of their heuristic, offering insights into its robustness under different conditions. However, despite these strengths, their work does not address other significant factors, such as energy consumption, which is a critical consideration in fog computing due to the energy constraints of edge devices. Additionally, their study lacks a formal model to rigorously analyse the behaviour and performance of the heuristic, which could have provided a deeper theoretical understanding and potential areas for improvement.

Ma and Zhang [2015] proposed a novel algorithm designed to offload services in cloud environments with the primary objectives of minimising energy consumption and resource usage while maximising Quality of Service (QoS). Their approach is grounded in a weighted-sum method, which allows for the simultaneous optimisation of multiple objectives by assigning different weights to each factor based on its importance. To evaluate the effectiveness of their algorithm, the authors employed a simulator, conducting tests with three distinct single-objective versions of their heuristic. However, it is important to note that their study has some limitations: the simulation does not use realistic data, which may affect the generalisability of the results.

Kattepur et al. [2016] aimed to optimise service offloading in fog environments by developing an Integer Linear Programming (ILP) formulation alongside a heuristic approach. Their primary objective was to minimise energy consumption, while also incorporating latency as a key constraint. To assess the effectiveness of their proposed solutions, the authors conducted a numerical analysis using synthetic data. However, the study has some limitations: it employs a relatively simplistic model and does not utilise the ILP formulation in the evaluation phase.

Deng et al. [2016] focused on addressing a resource allocation problem within a fog computing environment, with the primary objective of minimising energy consumption while maintaining latency as a critical constraint. The authors developed a heuristic algorithm designed to allocate resources dynamically. This algorithm aims to minimise energy consumption by optimising the placement and operation of computational tasks across fog nodes, all while ensuring that latency requirements are met to maintain acceptable levels of service performance. To evaluate the effectiveness of their proposed solution, the authors conducted a numerical analysis under various workload scenarios, simulating different levels of demand and network conditions. This comprehensive evaluation provided insights into how their heuristic performs in real-world settings, highlighting its potential to improve energy efficiency in fog computing environments without compromising on latency.

Agarwal et al. [2016] introduced a heuristic approach designed for resource allocation within fog computing environments, focusing specifically on minimising delay and avoiding dependence on cloud resources. The heuristic was developed

to allocate resources efficiently within the fog network, ensuring that computational tasks are handled locally whenever possible to reduce the reliance on distant cloud servers. To evaluate the effectiveness of their proposed approach, the authors conducted a series of simulations using benchmarks adapted from cloud computing scenarios. These benchmarks provided a relevant comparison, allowing the authors to demonstrate how their heuristic performs in a fog computing context compared to traditional cloud-based methods. The simulation results highlighted the potential of their approach to reduce delays and improve overall service delivery within fog environments.

Skarlat et al. [2016] set out to maximise fog usage in clustered fog environments by developing an ILP formulation and a heuristic method. Their work aimed to optimise the allocation of resources within fog computing clusters. The ILP formulation provided a rigorous mathematical model to determine the optimal resource allocation that maximises the utilisation of fog nodes while meeting various constraints. Complementing this, the heuristic method was developed to offer practical, near-optimal solutions with lower computational overhead, making it more suitable for real-time applications. To assess the effectiveness of their proposed methods, the authors evaluated their approach using synthetic data within a simulated environment. This simulation allowed them to test their algorithms under controlled conditions and provided insights into their performance and feasibility in maximising fog resource usage.

An extension of this work by Skarlat et al. [2017] further refined the initial formulation by providing a more comprehensive ILP model and introducing a genetic algorithm with similar objectives. The extended research aimed to build upon the foundation laid by the previous study by incorporating additional complexities and improving the robustness of the resource allocation strategies. The updated ILP model included more detailed constraints and objectives, enhancing the capability to handle diverse and dynamic fog computing scenarios. Additionally, the genetic algorithm was introduced as an alternative approach to explore heuristic solutions that might offer good performance with less computational effort compared to exact ILP solutions. To evaluate the improvements, the extended study included another round of simulation experiments, where the performance of the genetic algorithm was compared against the optimal solutions provided by the more detailed ILP model. This comparison provided valuable insights into the trade-offs between solution quality and computational efficiency, highlighting the effectiveness of the genetic algorithm in achieving near-optimal results while dealing with complex resource allocation problems in fog environments.

Azimzadeh and Biabani [2017] proposed a heuristic method specifically designed to optimise resource allocation for services within cloud environments, with a strong emphasis on scheduling considerations. Their approach was centered on enhancing QoS by targeting two critical aspects: improving response time and increasing the reliability of network connections. The heuristic method developed aimed to streamline the scheduling of computational tasks and network resources to minimise delays and ensure stable, reliable connections. To evaluate the effectiveness of their proposed heuristic, the authors conducted a detailed

numerical analysis using synthetic data.

Chang et al. [2017] sought to develop an energy-efficient scheme for offloading services in fog computing environments. Their approach integrates energy consumption and latency as dual objectives, which are combined into a single utility function to balance these factors effectively. To address this optimisation problem, the authors proposed a Non-Linear Programming (NLP) formulation alongside a distributed algorithm designed to find optimal solutions. For evaluation purposes, they conducted simulations using synthetic data, allowing them to analyse the performance of their proposed scheme and compare it with results from related works in the field.

Minh et al. [2017] aimed to optimise SP within fog computing environments by developing a specialised algorithm. The primary objective of their algorithm was to maximise the utilisation of fog resources while ensuring that service deadlines are consistently met. The proposed algorithm focused on balancing the allocation of resources to enhance overall utilisation, thus improving the efficiency of the fog infrastructure. Simultaneously, it was designed to guarantee that all service requests are processed within their specified deadlines, which is vital for maintaining high-quality service delivery and meeting performance expectations. To thoroughly evaluate the performance of their algorithm, the authors employed a simulation environment. This approach allowed them to test the algorithm under a range of configurations and scenarios, providing insights into how it performs across different conditions and workloads.

Zhang et al. [2017] sought to develop a solution for optimal resource allocation in fog computing environments by leveraging a game-theory-based algorithm. Their approach aimed to enhance resource usage efficiency through the application of dynamic pricing mechanisms, a technique inspired by economic theories of competition and market equilibrium. The game-theory framework allows for modeling interactions between multiple agents or services competing for limited resources, thus guiding the allocation process in a way that maximises overall efficiency and utility. To thoroughly evaluate the effectiveness of their proposed algorithm, the authors conducted a detailed numerical analysis. This analysis involved simulating various scenarios and conditions to test how well the algorithm performs in different operational environments.

Feng et al. [2018] focused on optimising resource allocation in fog computing environments with an emphasis on fairness. They developed a Mixed Integer Non-Linear Programming (MINLP) formulation designed to minimise energy consumption while balancing resource use across the network. To facilitate evaluation of their model, the authors relaxed certain variables, thereby converting the MINLP formulation into a Linear Programming (LP) problem, which is simpler to solve and analyse. Afterwards, the authors performed a numerical analysis to assess the performance of their approach and compared the results with those from related studies to highlight its effectiveness and relative advantages.

Zhang et al. [2018] proposed a service offloading scheme specifically designed for fog computing environments, with a focus on optimising both energy consump-

tion and delay. In their approach, they developed an algorithm that assigns each fog node a probability value based on its historical energy consumption. This probability value helps in making informed decisions about which nodes are more or less likely to be efficient in terms of energy usage. Additionally, each fog node is assigned a priority, which is determined by its current usage levels and available capacity. This dual consideration of historical energy data and real-time capacity ensures that service offloading decisions are made to minimise both energy expenditure and service delay, enhancing the overall efficiency of the fog network. To assess the effectiveness of their proposed scheme, the authors conducted a comprehensive simulation. This simulation involved comparing their algorithm against a baseline approach, which represents standard or existing methods for service offloading. Additionally, they evaluated two modified versions of their algorithm to explore different configurations and parameters. By comparing the performance of their proposed scheme with the baseline and the variations, the authors were able to analyse its impact on reducing energy consumption and delay.

Zhao et al. [2018] aimed to address the challenge of fair resource allocation in cloud environments. They proposed an innovative approach designed to ensure that each request receives a fair share of resources. This method seeks to balance the allocation process by distributing resources equitably among users, thereby addressing issues of fairness that are crucial in multi-tenant cloud environments where resource contention can lead to unequal service levels. The primary objective of their approach was to maximise resource efficiency while adhering to fairness properties. By integrating fairness considerations into the resource allocation strategy, the authors aimed to achieve a more balanced distribution of resources that prevents any single user or application from monopolising the available capacity. To evaluate the effectiveness of their proposed approach, the authors conducted an experimental analysis. This involved running experiments to assess how well their algorithm performs in terms of maintaining fairness among users while maximising resource utilisation. The experimental analysis also included a comparison with a state-of-the-art approach, allowing the authors to benchmark their solution against existing methods.

Kayal and Liebeherr [2019] aimed to address the SP problem within fog computing environments. They proposed an approximation algorithm designed to minimise both energy consumption and cost. To achieve this, the authors employed a weighted-sum approach to balance and optimise these objectives. The effectiveness of their algorithm was evaluated through numerical analysis, which involved assessing its performance under various conditions. Despite this, their study has some limitations: it did not utilise realistic data, which could impact the applicability of their results to real-world scenarios. Additionally, the evaluation was conducted solely through numerical methods without the aid of a simulation environment.

Meskar and Liang [2020] introduced a mathematical framework aimed at optimising the fairness of resource allocation in mobile edge computing environments, where various types of resources are involved. Their approach focuses on ensuring equitable distribution of resources among different services or users. For

the experimental analysis, the authors conducted a numerical evaluation of their framework and compared its performance with two state-of-the-art approaches in the field. While this comparative analysis provides valuable insights into the framework's effectiveness relative to existing methods, the study has some limitations. Specifically, it did not incorporate realistic data, which could affect the applicability of the findings to actual operational environments. Additionally, the research did not use a simulation environment, which could have offered a more dynamic and practical assessment of the framework's performance under real-world conditions.

The goal of Panadero et al. [2021] was to solve a SP problem in a fog environment, to maximise bandwidth usage and minimise the replication of resources. To tackle the problem, the authors proposed two approaches: a deterministic heuristic and a randomised algorithm, where they used a lexicographic ordering of objectives. To evaluate the approaches, the authors used realistic data and provided a comparison between different state-of-the-art approaches. However, energy consumption was not considered nor a comparison with the optimal solution.

Zhang et al. [2022] introduced a novel approach to address the SP problem within mobile edge computing environments, focusing on maximising profit by considering varying service request rates and pricing models. To effectively tackle this challenge, the authors developed a NLP formulation to model the problem comprehensively and employed a clustering algorithm to optimise service placement decisions based on the characteristics of the edge devices and network conditions. For the experimental analysis, the study utilised multiple baseline algorithms for comparison, including standard optimisation techniques and existing heuristic approaches. The authors conducted a thorough numerical analysis to evaluate the performance of their proposed solution, demonstrating its effectiveness in achieving higher profitability under diverse service demands and cost scenarios.

Qin et al. [2023] developed a Mixed Integer Linear Programming (MILP) formulation to address the SP problem within fog computing environments, with the overarching goal of enhancing QoS. Their approach focused on achieving multiple objectives: minimising deadline violations, reducing overall costs, and maximising the utilisation of available resources. To tackle the SP problem effectively, the authors implemented a genetic algorithm, which is known for its ability to find near-optimal solutions to complex optimisation problems. The genetic algorithm was used to explore the solution space efficiently, providing a practical means of solving the MILP formulation in scenarios where exact solutions might be computationally infeasible. For the evaluation of their proposed solution, the authors conducted a comprehensive numerical analysis. This analysis involved running simulations to assess the performance of their genetic algorithm in meeting the objectives set forth in their MILP model. They compared the results obtained from their approach with those from various state-of-the-art methods.

Table 3.1 provides a comprehensive overview of various proposals addressing

the SP problem. This table systematically categorises each proposal based on several key aspects, including the metrics used, the objectives pursued, the mathematical formulations employed, and the approaches taken. The metrics column highlights the specific performance indicators evaluated by each proposal, such as resource utilisation, latency, or cost efficiency. The objectives' column outlines the primary goals of each proposal, which may include optimising service delivery, minimising operational costs, or enhancing system reliability.

The mathematical formulation section of the table details the theoretical models used to solve the SP problem. However, it is noteworthy that this section contains considerable gaps, as not all proposals provide a detailed multi-objective formulation. This lack of information limits the ability to fully understand and compare the computational approaches used. Similarly, the approaches' column describes the methodologies applied in each proposal, such as heuristic methods and optimisation algorithms.

Table 3.2 offers an organised summary of various proposals in terms of their architectural frameworks, tools employed, and practical applications. The architecture column presents an overview of the structural designs proposed for implementing SP solutions. This includes descriptions of network configurations, system components, and deployment strategies. The tools' column details the software or platforms used in each proposal, such as optimisation tools, simulation environments, or programming languages.

In addition to these elements, the applications' column describes the real-world scenarios or case studies where the proposed solutions are applied. This could range from cloud computing environments to edge computing platforms or specific industry applications. However, similar to Table 3.1, Table 3.2 also reflects significant missing data, particularly in the areas of architectural specifics and applications. Many entries lack detailed information about the architecture or tools used, and some proposals do not provide explicit examples of applications. This incomplete data makes it challenging to evaluate the practical implementation and effectiveness of the proposals in real-world settings. The analysis underscores both the breadth of proposed solutions and the need for more comprehensive details to fully understand their practical implications and operational feasibility.

In the reviewed literature, the authors typically address the problem by focusing on a single objective or by consolidating multiple objectives into a single overarching goal. This is often achieved through techniques such as utility functions or ϵ-constraints. Utility functions allow for the aggregation of various performance metrics into a single value, representing the overall effectiveness or benefit of a solution. Alternatively, ϵ-constraint methods involve transforming multi-objective problems into single-objective problems by treating one or more objectives as constraints with predefined thresholds.

However, this approach can oversimplify complex problems, as it may not fully capture the trade-offs between different objectives. For instance, optimising for one metric, such as cost, might compromise other important factors like performance or reliability. This reductionist approach can lead to solutions that

Table 3.1.: Overview of Service Placement

Proposals	**Metrics**	**Objectives**	**Formulation**	**Approaches**
Nishio et al. [2013]	Energy Consumption Latency	Minimisation	Utility Function	NLP Heuristic
Jia et al. [2014]	Completion time	Minimisation	-	Heuristic
Ma and Zhang [2015]	Energy Consumption Resource Usage	Minimisation	Weighted-sum	Heuristic
Kattepur et al. [2016]	Energy Consumption Latency	Minimisation	ϵ-constraint	ILP
Deng et al. [2016]	Energy Consumption Latency	Minimisation	ϵ-constraint	Heuristic
Agarwal et al. [2016]	Resource Usage	Minimisation	-	Heuristic
Skarlat et al. [2016]	Resource Usage	Minimisation	-	ILP
Skarlat et al. [2017]	Resource Usage	Minimisation	-	ILP Genetic Algorithm
Azimzadeh and Biabani [2017]	Latency	Resilience	-	Genetic Algorithm
Chang et al. [2017]	Energy Consumption Latency	Minimisation	ϵ-constraint	NLP Heuristic
Minh et al. [2017]	Allocated Services	Maximisation	-	ILP
Zhang et al. [2017]	Cost	Minimisation	-	ILP Heuristic
Feng et al. [2018]	Energy Consumption	Fairness	-	MINLP
Zhang et al. [2018]	Energy Consumption Latency	Fairness	ϵ-constraint	Heuristic
Zhao et al. [2018]	Resource Usage	Fairness	-	ILP Heuristic
Kayal and Liebeherr [2019]	Energy Consumption Cost	Minimisation	Weighted-sum	ILP Heuristic
Meskar and Liang [2020]	Resource Usage	Fairness	-	ILP
Panadero et al. [2021]	Bandwidth Replication	Minimisation	Lexicographic Ordering	ILP Heuristic
Zhang et al. [2022]	Profit	Maximisation	-	NLP Heuristic
Qin et al. [2023]	Deadline violations Cost Resource Usage	Minimisation	Weighted-sum	MILP Heuristic

excel in one area while potentially neglecting others that are crucial for a holistic evaluation.

Additionally, the scenarios presented in these works are often simplistic, lacking the complexity and variability inherent in real-world systems. Simplistic scenarios may not adequately represent the multifaceted nature of practical applications, leading to results that might not be as effective or applicable in more complex environments.

Furthermore, in many cases, some important objectives are not treated as primary goals but rather as constraints within the optimisation framework. This means that these objectives are only addressed to the extent that they do not exceed certain limits, rather than being actively optimised. As a result, their impact on the overall solution might be limited or overlooked.

Finally, the studies frequently rely on synthetic data for analysis and validation. Synthetic data is artificially generated and may not accurately reflect the real-world complexities and variabilities of actual/realistic data. While it can be useful for initial testing and theoretical analysis, synthetic data may not fully capture the practical challenges and dynamics, potentially limiting the applicability and robustness of the proposed solutions in real-world scenarios.

Table 3.2.: Simulation in Service Placement

Proposals	**Architecture**	**Tools**	**Applications**
Nishio et al. [2013]	Mobile Cloud Computing	Numerical Analysis	Task Sharing
Jia et al. [2014]	Mobile Cloud Computing	Custom Simulator	-
Ma and Zhang [2015]	Cloud Computing	CloudSim	-
Kattepur et al. [2016]	Fog Computing	Numerical Analysis	-
Deng et al. [2016]	Fog Computing	Numerical Analysis	-
Agarwal et al. [2016]	Fog Computing	CloudAnalyst	-
Skarlat et al. [2016]	Fog Computing	CloudSim	-
Skarlat et al. [2017]	Fog Computing	iFogSim	-
Azimzadeh and Biabani [2017]	Cloud Computing	Numerical Analysis	-
Chang et al. [2017]	Fog Computing	Numerical Analysis	-
Minh et al. [2017]	Fog Computing	Numerical Analysis	Smart Grid
Zhang et al. [2017]	Fog Computing	MATLAB	-
Feng et al. [2018]	Fog Computing	Numerical Analysis	-
Zhang et al. [2018]	Fog Computing	Numerical Analysis	-
Zhao et al. [2018]	Cloud Computing	CloudSim	-
Kayal and Liebeherr [2019]	Fog Computing	Numerical Analysis	-
Meskar and Liang [2020]	Mobile Edge Computing	Numerical Analysis	-
Panadero et al. [2021]	Fog Computing	Numerical Analysis	-
Zhang et al. [2022]	Mobile Edge Computing	MATLAB	-
Qin et al. [2023]	Fog Computing	MATLAB	-

3.2 Virtual Network Embedding

The Virtual Network Embedding (VNE) problem aims to allocate a set of Virtual Network (VN) into the substrate network. Each VN represents the necessary resources used by a service. This is particularly useful in Service Function Chaining (SFC). In this section, we describe related works containing single and multi-objective versions of the problem.

Rahman and Boutaba [2013] aimed to address the challenge of allocating VNs within fog computing environments, focusing on optimising key metrics such as energy consumption, bandwidth utilisation, and reliability. The goal was to develop a solution that balances energy efficiency with high bandwidth and reliability, ensuring that virtual networks perform optimally while meeting operational constraints. To tackle this complex problem, the authors proposed a heuristic approach based on LP. Their heuristic incorporates several configurable parameters that allow for fine-tuning based on specific requirements and constraints. Additionally, they utilised path-flow optimisation formulations to address various recovery and protection policies, which are essential for handling network failures and ensuring service continuity. The heuristic approach leverages LP techniques to provide a practical means of solving the VN allocation problem efficiently, even in dynamic and large-scale fog environments. To evaluate their proposed solution, the authors developed a comprehensive simulator designed to model the fog environment and test their heuristic approach.

Melo et al. [2013] addressed the VN embedding problem using an ILP formulation. Their approach involves defining three distinct objective functions: load balancing, minimisation of links, and minimisation of links while considering capacity constraints. This comprehensive framework aims to optimise various aspects of network embedding, offering a multifaceted solution to the problem. To validate their approach, the authors compared their ILP formulation with several existing methods in the field, providing a comparative perspective on its

effectiveness.

Guerzoni et al. [2014] developed a MILP formulation to address the VNE problem, with the primary goal of maximising revenue while meeting capacity constraints. Their approach was designed to balance financial objectives with the need to manage limited resources effectively. To assess the performance of their MILP formulation, the authors implemented it into a solver and conducted a numerical analysis, comparing results across various parameter settings to evaluate the robustness and flexibility of their solution.

Gao et al. [2015] addressed the Virtual Network Embedding (VNE) problem with a focus on incorporating a fairness metric specifically aimed at maximising the reliability for the lowest reliability request. Their approach involves two main components: a MILP formulation and a genetic algorithm, both designed to optimise resource allocation while considering the fairness of reliability across different network requests. To evaluate the effectiveness of their proposed methods, the authors conducted simulations, comparing their approach against two other strategies: a randomised version of their heuristic and a randomised mapping algorithm. This comparative analysis provided insights into the performance of their methods relative to alternative approaches. By incorporating simulations, the study was able to test the practical applicability of their solutions in a controlled environment.

Németh et al. [2016] developed a MILP formulation to tackle the VNE problem, accompanied by a greedy heuristic specifically designed for embedding service graphs. Their approach focuses on minimising resource usage while maximising the number of successfully embedded VNs. To assess the performance of their methods, the authors conducted a numerical evaluation comparing the optimal solutions derived from the MILP formulation with those obtained from the greedy heuristic. This comparative analysis allowed them to gauge the effectiveness and efficiency of the heuristic relative to the theoretically optimal MILP solutions.

Gomes et al. [2017] aimed to solve the problem of deploying VNs to allow the fulfilment of Service Level Agreement (SLA) in fog environments. An algorithm to deploy VNs with energy consumption and bandwidth as their objectives, while considering a minimum resilience on the deployed network was proposed. The authors developed an algorithm designed to generate a set of solutions that meet a minimum requirement for resilience, while forming a Pareto front. Following the generation of the Pareto front, a subsequent approach was used to select the most appropriate solution from this set. This selection process involved evaluating the solutions based on specific criteria or objectives relevant to the context, such as overall performance, cost, or resilience. The chosen solution represents the best balance among the trade-offs presented by the Pareto front, tailored to meet the particular needs of the application or scenario. To validate and assess the performance of their approach, the authors conducted a numerical analysis. This analysis involved running simulations or computations to evaluate how well the generated solutions performed under different conditions and constraints.

Leconte et al. [2018] proposed a novel approach for Network Slicing (NS) environments aimed at maximising fairness based on a utility function that takes into account both bandwidth and computational resources. The proposed approach incorporates a utility function designed to balance these two critical resources—bandwidth and computational capacity—thus addressing the needs of different network slices in a fair manner. To provide flexibility in managing the trade-offs between computing fairness and traffic fairness, the authors introduced a tunable parameter within their approach. This parameter allows for dynamic adjustment of the balance between ensuring fair allocation of computational resources and fair distribution of network traffic, depending on the specific requirements and priorities of the NS environment. By adjusting this parameter, network operators can tailor the fairness criteria to better match their operational goals and performance objectives. To demonstrate the effectiveness and performance of their proposed algorithm, the authors conducted a detailed numerical analysis. This analysis involved running simulations and evaluating how well the algorithm performs in achieving fairness across different slices and resource types.

Zhang et al. [2019] developed a heuristic specifically designed to embed Virtual Network Functions (VNFs) within 5th Generation (5G) NS environments. Their approach aimed to achieve two primary objectives: maximising the total number of accepted requests and minimising VNF interference. The heuristic developed by the authors addresses these challenges by efficiently allocating resources to accommodate as many requests as possible while reducing conflicts and interference between VNFs, which can degrade network performance and service quality. To evaluate the effectiveness of their heuristic, the authors simulated a 5G network topology incorporating slices dedicated to specific use cases, such as autonomous driving and high-definition video streaming. By modeling this scenario, the authors were able to test how well their heuristic performs in a realistic and complex 5G environment. The evaluation involved comparing their heuristic with two baseline algorithms to benchmark its performance. The baseline algorithms provided reference points for assessing the improvements achieved by the proposed approach.

Alfoudi et al. [2019] proposed a customisable mechanism designed to optimise resource allocation in Long-Term Evolution NS environments with the goal of maximising the utilisation of channel resources. The proposed mechanism focuses on allocating resources effectively to improve overall network efficiency and service quality. To address this challenge, the authors developed two distinct algorithms. The first algorithm targets intra-slice resource allocation, which deals with managing resources within a single network slice. This algorithm ensures that the resources allocated to different components or users within the same slice are optimised to meet the slice's specific needs and performance goals. The second algorithm focuses on inter-slice resource allocation, which manages the distribution of resources across multiple network slices. This approach aims to balance the allocation of resources between different slices, ensuring that each slice receives an appropriate share based on its demands and priorities. Additionally, the authors incorporated a max-min model to address user fairness

and resource isolation between slices. This model helps to maintain a balanced distribution of resources while isolating resources between slices to prevent interference and ensure reliable performance. For their experimental analysis, the authors conducted a simulation to evaluate the effectiveness of their proposed algorithms. The simulation allowed them to model various scenarios and conditions within Long-Term Evolution NS environments, providing insights into how well their algorithms perform in optimising channel resource utilisation, fairness, and resource isolation.

Mei et al. [2020] proposed a novel algorithm designed to address the VNE problem within NS environments, with a primary focus on minimising overall resource usage. The VNE problem involves efficiently mapping VNFs onto physical network infrastructure, a task that becomes more complex in NS environments where multiple virtual networks or slices share the same physical resources. A key innovation of their approach is the assumption that VNFs can be shared between different slices. By allowing VNFs to be utilised across multiple slices, the algorithm aims to enhance resource efficiency and reduce the total resource consumption. This sharing mechanism helps to optimise the use of physical resources by avoiding redundant deployments of the same VNF and leveraging existing infrastructure more effectively. To evaluate the performance and effectiveness of their algorithm, the authors conducted a series of simulations. These simulations were designed to model various scenarios within NS environments, providing a controlled setting to test how well the algorithm performs in minimising resource usage.

Fossati et al. [2020] proposed an innovative algorithm designed to manage multiple types of resources within 5G NS environments, with the primary objective of maximising resource fairness. Their algorithm is distinguished by its ability to handle a broad range of resource types. It uses an allocation matrix as a key parameter to define how resources are distributed among different slices. This matrix enables the algorithm to manage various resources simultaneously by specifying the allocation strategy for each type. Additionally, the algorithm incorporates weights for each type of resource, allowing for tailored prioritisation based on the specific needs and importance of different resources in the network slices. To validate the effectiveness of their proposed approach, the authors conducted a detailed numerical analysis. This analysis involved using real-world demand templates to simulate actual resource usage scenarios and assess how well the algorithm performs under realistic conditions.

Iwai and Nakao [2021] proposed an algorithm specifically designed for NS environments with the primary objective of maximising fairness in bandwidth allocation. To achieve this goal, the authors developed a utility function that models the usage of bandwidth across various network slices. This utility function is tailored to account for the specific characteristics and requirements of each type of slice. By differentiating these functions based on slice types, the algorithm is able to more accurately and effectively allocate bandwidth according to the unique needs and priorities of each slice. For the experimental analysis of their proposed algorithm, the authors conducted simulations using real-world trace data.

Zhang et al. [2021] presented an ILP formulation alongside a heuristic method for the placement of VNFs in smart city environments. The proposed heuristic leverages a randomised approach to accommodate the varying requirements of different types of services, ensuring flexibility and scalability in dynamic urban settings. The authors conducted an extensive numerical evaluation to assess the performance of both the ILP formulation and the heuristic, analysing key parameters such as resource allocation efficiency, latency, and overall QoS. The results highlighted the heuristic's capability to provide near-optimal solutions with reduced computational complexity, making it a viable option for real-time VNF placement in smart cities.

Yang et al. [2021] focused on addressing the VNE problem specifically within mobile edge computing environments, where efficient bandwidth usage is critical. The authors proposed an ILP formulation to rigorously define the optimisation problem and a heuristic designed to minimise bandwidth consumption while maintaining service quality. The heuristic was crafted to offer a balance between solution quality and computational feasibility. To evaluate the effectiveness of their proposed solutions, the authors performed a comprehensive numerical analysis, comparing the results of their ILP formulation and heuristic against several state-of-the-art approaches. This comparative study highlighted the advantages of their methods in terms of reduced bandwidth usage and improved embedding efficiency, especially under varying network conditions and service demand patterns.

Cao et al. [2021] developed an innovative embedding and scheduling algorithm aimed at optimising the deployment of VNs in next-generation networks. The primary objective of their algorithm was to minimise deployment time, thereby enhancing the responsiveness and efficiency of network services. The authors conducted a detailed evaluation of their algorithm through numerical analysis, focusing on key performance metrics such as the deployment time of VNFs, the acceptance ratio of VN requests, and overall resource usage. By testing different variants of their proposed approach, the authors demonstrated its potential to significantly reduce deployment times and improve network resource allocation, making it a promising solution for the next generation of mobile networks.

Martin-Perez et al. [2022] proposed a novel algorithm aimed at optimising the embedding of VNFs within 5G NS environments that leverage fog and cloud computing resources. The core objective of this algorithm is to minimise the overall resource costs associated with deploying and managing VNFs, while still ensuring that critical performance requirements are met. These performance requirements include maintaining low latency for real-time processing, high reliability to minimise service disruptions, and adequate availability to ensure continuous access to network services. To validate the effectiveness of their algorithm, the authors conducted a thorough numerical analysis using data from realistic network environments. This analysis was designed to simulate various operational scenarios and assess the algorithm's performance under different conditions. Additionally, they carried out experimental validation using a mobile robot test-bed. This test-bed allowed them to observe the algorithm's performance in a practical, real-world setting, providing insights into its effectiveness

and practicality for actual deployment.

Promwongsa et al. [2022] presented an ILP formulation along with two heuristic methods for embedding VNs within fog computing environments. The primary objectives of this work were to address scheduling and delay constraints while focusing on maximising profits. The ILP formulation provides a precise mathematical model to optimally place and schedule VNFs across the fog infrastructure, ensuring that the network's performance requirements are met while also achieving the highest possible profit. In addition to the ILP model, the authors introduced two heuristic approaches designed to offer practical and scalable solutions, potentially faster and more adaptable than the exact ILP method. To evaluate the efficacy of these approaches, the authors conducted a comprehensive comparison against several state-of-the-art techniques.

Sato et al. [2022] sought to address the VNE problem within fog computing environments, which are characterised by distributed resources and lower latency requirements. To achieve this, they proposed an ILP formulation coupled with a heuristic designed to minimise both the operational costs and the number of virtual requests that need to be serviced. Their approach aimed to optimise resource allocation while maintaining service quality across the distributed fog infrastructure. The authors conducted an extensive numerical experimental analysis to evaluate the performance of their methods, focusing on metrics such as cost efficiency, request handling capabilities, and resource utilisation. The results of their study underscored the effectiveness of their ILP and heuristic in reducing costs and improving network performance in fog environments.

Jalalian et al. [2023] developed a sophisticated ILP formulation and a complementary heuristic to tackle the VNE problem specifically in NS fog environments. The primary objectives of their approach were to minimise resource usage and reduce the number of VNF instances required for service delivery, thereby optimising the overall efficiency of the network. To assess the effectiveness of their proposed solutions, the authors conducted a focused experimental analysis, which involved a numerical comparison of various metrics of interest, such as resource consumption, VNF instance count, and the overall performance of the embedding process. This analysis provided valuable insights into the advantages and potential limitations of their ILP formulation and heuristic in managing resources within complex fog computing scenarios.

Dimolitsas et al. [2023] presented a comprehensive set of heuristics designed to address the VNE problem specifically in fog computing environments. Their work initially focused on intra-cluster allocation, where VNFs are allocated within a single cluster of fog nodes, optimising the use of local resources. This was followed by a phase of distributed embedding, which involves mapping virtual networks across multiple clusters to ensure efficient utilisation of geographically dispersed resources. The primary goal of these approaches was to minimise the latency of communications, especially in dynamic scenarios where network conditions and resource availability can change rapidly. To evaluate the effectiveness of their proposed heuristics, the authors conducted an extensive experimental analysis using a simulation framework that incorporated real-world data.

This allowed them to simulate realistic network conditions and demonstrate the ability of their heuristics to reduce latency and improve overall network performance in fog computing environments.

Table 3.3 provides a detailed examination of various proposals addressing the VNE problem, focusing on key aspects such as metrics, objectives, multi-objective formulations, and approaches. It catalogues different proposals, specifying the performance metrics used, the primary objectives pursued, and the multi-objective formulations applied to balance competing goals, although it reveals significant gaps in the mathematical formulations as many proposals lack detailed descriptions. The approaches' column outlines the methodologies employed, yet, like the formulations, it is undefined for several entries.

Table 3.4 complements this by summarising the proposals in terms of their architecture, tools, and applications. It details the structural designs and the software or platforms used, as well as the practical applications where these solutions are implemented. However, this table also reflects a considerable amount of missing data, particularly in the architecture and applications columns, limiting the understanding of how these proposals are operationalised and applied in real-world scenarios. Both tables highlight the breadth of research in VNE while indicating the need for more comprehensive data to fully evaluate and compare the proposed solutions.

In the studies reviewed, the authors typically focused on a single objective, often neglecting critical factors such as energy consumption and latency, which are particularly significant in fog computing environments. Fog computing, due to its distributed nature, involves complex interactions between multiple layers of computing and networking infrastructure. As such, both energy efficiency and latency are crucial aspects that directly impact the performance and sustainability of fog systems.

Furthermore, the scenarios examined in these works were often overly simplistic, lacking the complexity and variability that are characteristic of real-world environments. This simplicity in scenario design means that the studies might not accurately reflect the challenges and dynamics encountered in practical applications. Additionally, the majority of these studies did not employ simulation tools, which are essential for creating realistic models and testing the performance of proposed solutions under varied conditions. Instead, the studies relied predominantly on synthetic data, which, while useful for theoretical analysis, may not fully capture the nuances and unpredictabilities of actual operational data. The absence of simulation and reliance on synthetic data limit the generalisability and applicability of the findings, potentially affecting the robustness of the proposed solutions in real-world fog computing contexts.

3.3 Reconfiguration and Migration

The Reconfiguration and Migration problem aims to reallocate resources after a failure or connectivity error occurs between the network and the user. In this section, we describe related works containing single and multi-objective versions

Table 3.3.: Overview of Virtual Network Embedding

Proposals	**Metrics**	**Objectives**	**Formulation**	**Approaches**
Rahman and Boutaba [2013]	Bandwidth	Resilience	-	MILP Heuristic
Melo et al. [2013]	Bandwidth	Load Balancing	-	ILP
Guerzoni et al. [2014]	Profit	Maximise	-	MILP
Gao et al. [2015]	Allocated VNs	Fairness Resilience	-	MILP Genetic Algorithm
Németh et al. [2016]	Resource Usage Embedded VNs	Minimisation	-	MILP Heuristic
Gomes et al. [2017]	Energy Consumption Throughput	Resilience	Representative Set of Solutions	ILP Heuristic
Leconte et al. [2018]	Resource Usage	Fairness	-	ILP Heuristic
Zhang et al. [2019]	VNF Interference	Minimisation	-	ILP Heuristic
Alfoudi et al. [2019]	Bandwidth Usage	Fairness	-	ILP Heuristic
Mei et al. [2020]	Resource Usage	Minimisation	-	ILP Heuristic
Fossati et al. [2020]	Resource Usage	Fairness	-	ILP
Iwai and Nakao [2021]	Bandwidth	Fairness	-	LP
Zhang et al. [2021]	Resource Usage Delay	Maximisation	ϵ-constraint	ILP Heuristic
Yang et al. [2021]	Bandwidth	Minimisation	-	ILP Heuristic
Cao et al. [2021]	Deployment Time	Minimisation	-	Heuristic
Martin-Perez et al. [2022]	Cost	Minimisation	-	ILP Heuristic
Promwongsa et al. [2022]	Profit	Maximisation	-	ILP Heuristic
Sato et al. [2022]	Cost Virtual Requests	Minimisation	-	ILP Heuristic
Jalalian et al. [2023]	Resource Usage VNF Instances	Minimisation	Weighted-sum	ILP Heuristic
Dimolitsas et al. [2023]	Delay	Minimisation	-	Heuristic

Table 3.4.: Simulation in Virtual Network Embedding

Proposals	**Architecture**	**Tools**	**Applications**
Rahman and Boutaba [2013]	-	Numerical Analysis	-
Melo et al. [2013]	-	MATLAB	-
Guerzoni et al. [2014]	-	Numerical Analysis	-
Gao et al. [2015]	-	Numerical Analysis	-
Németh et al. [2016]	-	Numerical Analysis	-
Gomes et al. [2017]	Edge Computing	Custom Simulator	-
Leconte et al. [2018]	Cloud Computing	Numerical Analysis	-
Zhang et al. [2019]	Network Slicing	Numerical Analysis	Autonomous Driving Video Streaming
Alfoudi et al. [2019]	Network Slicing	OPNET Modeler	FTP VoIP Video
Mei et al. [2020]	Network Slicing	Numerical Analysis	-
Fossati et al. [2020]	Network Slicing	Numerical Analysis	-
Iwai and Nakao [2021]	Network Slicing	Numerical Analysis	-
Zhang et al. [2021]	Mobile Edge Computing	Numerical Analysis	Smart Cities
Yang et al. [2021]	Mobile Edge Computing	Numerical Analysis	-
Cao et al. [2021]	-	Numerical Analysis	-
Martin-Perez et al. [2022]	Network Slicing	Testbed	Factory Setting
Promwongsa et al. [2022]	-	Numerical Analysis	-
Sato et al. [2022]	-	Numerical Analysis	-
Jalalian et al. [2023]	Network Slicing	Numerical Analysis	-
Dimolitsas et al. [2023]	Edge Computing	Numerical Analysis	-

of the problem.

Tran and Timm-Giel [2013] developed a dual approach to tackle the VNE reconfiguration problem, focusing on maximising profit within VNE contexts. They designed both a heuristic method and an ILP formulation to address this complex challenge. The heuristic provides a practical, computationally efficient solution by leveraging approximations and rule-based strategies, while the ILP formulation offers a rigorous, optimal approach that mathematically models the resource allocation problem. The heuristic method is useful for quickly generating feasible solutions in real-time scenarios, whereas the ILP formulation is valuable for achieving optimal resource allocation, albeit potentially at a higher computational cost. The combination of these methods allows for a more robust and adaptable approach to VNE reconfiguration. To evaluate the effectiveness of their proposed solutions, the authors implemented both the heuristic and ILP formulations within a simulation environment.

Cerroni and Callegati [2014] introduced a model specifically designed for the live migration of VM instances within edge computing environments. This model was crafted to handle both sequential and parallel migrations of multiple VMs, optimising the process to ensure minimal service interruption or downtime. The primary objective was to enhance the efficiency and reliability of VM migrations, thereby maintaining high availability and performance standards in edge networks. To assess the effectiveness of their model, the authors conducted a thorough numerical analysis. This analysis involved varying several parameters to understand their impact on key metrics such as downtime and migration time. By simulating different scenarios and configurations, they were able to demonstrate how their approach could effectively reduce downtime and optimise the overall migration process, making it suitable for real-time applications in edge computing where service continuity is critical.

Eramo et al. [2014] focused their research on the migration of virtual routers, with the primary goal of minimising the cost associated with embedding these virtual network elements into physical networks. To address this optimisation problem, they developed both an ILP model and a heuristic algorithm. The ILP model provided a precise mathematical formulation of the problem, aiming to find the optimal solution that minimises embedding costs, while the heuristic offered a more computationally efficient approach to achieve near-optimal solutions. To validate their proposed methods, the authors conducted a numerical analysis using a realistic network topology, which allowed them to simulate the complexities of actual network environments. However, despite the thorough numerical analysis, the study did not incorporate empirical data from real-world scenarios, which could have further validated the robustness and applicability of their solutions in practical, live network conditions.

Dalvandi et al. [2015] presented a comprehensive approach for managing VM resources in dynamic data center environments, focusing on the placement, routing, and migration of these resources within a specified time frame. They explored scenarios both with and without the migration of VMs, developing four distinct heuristics to approximate optimal solutions. These heuristics were

designed to balance two key objectives: maximising the number of accepted resource requests, thereby improving data center utilisation and service availability, and minimising energy consumption to enhance operational efficiency and sustainability. To rigorously evaluate the effectiveness of their proposed methods, the authors employed a simulation-based approach, comparing the performance of the different heuristics under a variety of simulated conditions. This allowed them to demonstrate the advantages and trade-offs of each heuristic, providing valuable insights into their potential applications in real-world data center operations.

Jiang et al. [2016] introduced a routing and re-routing algorithm designed to enhance both energy efficiency and load balancing in smart city environments. Their approach aims to optimise network performance by addressing these two critical objectives simultaneously. For the simulation of their algorithm, the authors utilised a fixed network topology and evaluated its performance under various traffic requests, allowing them to assess how well the algorithm adapts to different network conditions. Additionally, they conducted a comparative analysis by considering several related works, providing a broader context for evaluating the effectiveness of their proposed solution.

Melo et al. [2016] presented two distinct strategies to address the reconfiguration challenges associated with VNE problems. The first strategy involved a clone reconfiguration approach that does not make specific assumptions about the underlying protocols or architectures. This method focuses on reconfiguring virtual networks by creating and deploying clones, which allows for flexible adjustments without being constrained by predefined network configurations or protocol requirements. The second strategy involved the development of an ILP formulation specifically designed to handle the reconfiguration of resources that are prone to failure. This approach aims to minimise both bandwidth consumption and the frequency of reconfigurations, addressing the need for efficient resource management in the face of potential failures. By focusing on these aspects, the ILP formulation seeks to optimise resource allocation while reducing operational overhead associated with frequent reconfigurations. To evaluate the effectiveness of their proposed strategies, the authors utilised an experimental testbed, which allowed them to test their approaches under various conditions and parameters.

Khai et al. [2017] proposed a multistep ILP model specifically designed for reconfiguring VNs. The model operates in two distinct phases to optimise different aspects of the reconfiguration process. In the first phase, the ILP formulation focuses on finding a target configuration that minimises the overall operational cost of the network. This involves optimising resource allocation and reducing expenses associated with network maintenance and operation. In the second phase, the goal shifts to minimising the number of hops in the network paths, which is crucial for improving data transmission efficiency and reducing latency. By optimising the number of hops, the model aims to enhance network performance, particularly in terms of speed and reliability. For the evaluation of their model, Khai et al. used realistic network values alongside randomly generated virtual networks to simulate various network conditions and configurations.

Eramo et al. [2017] introduced an innovative approach to minimising energy consumption during the reconfiguration of VNs by presenting both an ILP formulation and a heuristic method. The ILP formulation provided a mathematical model that aimed to find the optimal solution for energy-efficient reconfiguration of VNs, ensuring that the use of computational and network resources is minimised. Complementing this, the heuristic method was developed to offer a more computationally efficient alternative that could provide near-optimal solutions in a shorter amount of time. To evaluate the effectiveness of their proposed methods, the authors conducted a thorough numerical analysis. This analysis focused on verifying the energy consumption levels achieved by the new ILP formulation and heuristic, comparing these results to those obtained from a previous version of the heuristic. By doing so, the authors were able to demonstrate the improvements made in reducing energy consumption, highlighting the potential of their approach to enhance the sustainability of virtual network operations.

Gao and Rouskas [2018] focused on solving load balancing issues in VN reconfiguration problems by developing a set of heuristics and a robust mathematical model. To achieve this, they proposed a MILP formulation aimed at minimising the maximum utilisation of substrate nodes and links in the network. This approach was designed to evenly distribute the network load across available resources, preventing any single node or link from becoming a bottleneck and ensuring efficient network performance. Additionally, the MILP formulation incorporated constraints to limit the number of virtual nodes that need to be migrated during the reconfiguration process, balancing the trade-off between optimal load distribution and the overhead associated with node migration. For the evaluation of their proposed methods, the authors conducted tests using a variety of scenarios in a testbed environment. This enabled them to analyse the performance of their heuristics and MILP model under realistic conditions, demonstrating their effectiveness in achieving balanced load distribution and efficient resource utilisation during VN reconfigurations.

Yang et al. [2018] proposed an approach to deal with failures in multi-link networks in arbitrary network topologies. Their objective was to maximise the protection performance, which can be defined as the number of paths that are protected to the number of paths that include link failures. Furthermore, they proposed a packet drop approach to prevent damage when the previous approach cannot protect the routing, since the network is disconnected by a failure. The authors conducted a comprehensive numerical analysis to evaluate their proposed strategies. This analysis involved running a series of computational experiments to quantify the performance of their methods under various conditions and parameter settings.

Bi et al. [2018] focused on providing mobility support within fog computing environments by proposing an algorithm designed to find an optimal path that maximises performance and profit while minimising cost. Their approach addresses the challenge of maintaining efficient and cost-effective service delivery in dynamic and mobile fog environments. To evaluate the efficacy of their algorithm, the authors conducted simulations, which allowed them to test and refine their approach under controlled conditions.

Medeiros et al. [2019] introduced a multi-objective routing strategy using an ILP formulation and a dedicated algorithm tailored for wireless mesh networks. Their approach was designed to address three key, yet often conflicting, objectives that are crucial for optimising network performance. First, they aimed to minimise the number of hops along a route, which directly impacts latency and overall transmission time, making the network more responsive. Second, they sought to minimise bottlenecks, which are critical points in the network that can restrict data flow and lead to congestion, thereby enhancing throughput and network reliability. Third, they aimed to minimise channel interference on the path, which is essential for maintaining signal quality and reducing data transmission errors in the wireless environment. To validate their ILP model and algorithm, the authors conducted a series of simulations. They compared their proposed solutions against several existing approaches from related works by implementing all of them within a simulation framework. This comprehensive evaluation allowed them to demonstrate the effectiveness of their multi-objective approach in balancing these conflicting goals and improving overall network performance in various scenarios.

Noghani et al. [2019] focused on addressing the complex problem of VNF placement and reconfiguration by proposing a NLP formulation aimed at minimising the overall operational cost. This problem is particularly challenging in network management, as it involves optimally placing VNFs on physical infrastructure while also determining when and how to reconfigure these functions to adapt to changing network conditions and demands. The proposed NLP model was designed to reduce costs associated with resource utilisation, operational overhead, and reconfiguration activities, thereby enhancing the efficiency of network operations. To assess the effectiveness of their model, the authors performed a detailed numerical experimental analysis, examining various scenarios that included different configurations of virtual networks and parameters. This approach allowed them to explore the performance of their NLP formulation under a range of conditions and constraints.

Wang et al. [2019] developed both an ILP formulation and a heuristic method to address the task offloading and migration problem. Their approach aims to optimise the allocation and relocation of tasks within computing environments, balancing efficiency and effectiveness. For their evaluation, the authors utilised various parameter settings to test their methods and compared the results against several state-of-the-art approaches, providing a benchmark for assessing the performance of their solutions.

Pozza et al. [2020] addressed the complex challenge of reconfiguring VNs within NS environments in fog computing, particularly focusing on the needs of 5G applications. They proposed a novel algorithm specifically designed to dynamically migrate network slices, taking into account critical constraints such as bandwidth and latency. These constraints are essential for meeting the stringent performance requirements of 5G applications, which demand high-speed data transfer and low-latency responses. To evaluate the effectiveness of their proposed algorithm, the authors conducted a comparative analysis with a well-established A* algorithm, known for its efficiency in pathfinding and network

reconfiguration tasks. This comparison provided a benchmark to assess how the new algorithm performs relative to existing solutions.

Mouaci et al. [2020] addressed the challenging problem of VNF placement and routing within network infrastructures by proposing a comprehensive MILP formulation. Their approach was distinctive in that it simultaneously integrated both the placement of VNFs on physical network nodes and the routing of network traffic through these functions, all while aiming to minimise overall operational costs. This dual-objective optimisation is crucial for efficiently managing resources and ensuring optimal network performance, as it considers both where VNFs should be deployed and how network flows should be routed to meet demand while keeping costs low. To evaluate the effectiveness of their proposed MILP formulation, the authors conducted a comparative analysis against a prior version of their model that did not account for capacity constraints on network resources. This comparison allowed them to demonstrate the improvements in cost efficiency and resource utilisation brought about by incorporating capacity constraints, highlighting the enhanced capability of their new formulation to manage network resources more effectively under realistic conditions.

Padhy and Chou [2021] focused on the reconfiguration of virtualised datacenters, which involves dynamically adjusting the allocation of virtual resources to optimise performance and reduce costs. To tackle this problem, they developed both an ILP formulation and a heuristic method. The ILP formulation was designed to provide an optimal solution to the reconfiguration problem by mathematically modeling the various constraints and objectives involved, primarily focusing on cost minimisation. Meanwhile, the heuristic method offered a more computationally efficient alternative that could quickly generate near-optimal solutions, making it suitable for large-scale or time-sensitive applications. To validate the effectiveness of their approaches, the authors conducted an extensive evaluation using a realistic network topology that closely resembled actual datacenter environments. This evaluation not only demonstrated the practical applicability of their solutions but also allowed them to benchmark their performance against other state-of-the-art methods in the field. By comparing their ILP formulation and heuristic to existing approaches, they were able to show the advantages of their methods in terms of cost savings and operational efficiency, further underscoring the potential impact of their work on optimising virtualised datacenter management.

Li et al. [2022] introduced a comprehensive model and heuristic to efficiently manage the migration of computational tasks in edge computing environments, where meeting specific latency requirements is crucial. This model was designed to optimise the allocation and migration of tasks across edge devices, ensuring that latency-sensitive applications receive the necessary computational resources without excessive delays. The heuristic provided a more practical approach for handling task migration in real-time, making it particularly useful in dynamic and unpredictable edge computing scenarios. To assess the performance and practicality of their proposed solutions, the authors conducted a detailed evaluation using a simulation environment. They implemented their model in a simulator, utilising a fixed set of parameters to create a controlled testing frame-

work. Additionally, randomised transmission data to mimic the variability and unpredictability often encountered in real-world edge computing environments was introduced. This approach allowed them to effectively evaluate the model's ability to manage task migration and adhere to latency requirements under various conditions.

Hanauer et al. [2023] focused on optimising network performance in datacenters by developing a set of heuristics aimed at reconfiguring and scheduling network links to maximise throughput. Their heuristics were designed to dynamically adjust the configuration of data paths and link schedules in response to changing network demands, with the objective of improving the overall efficiency and capacity of datacenter networks. By maximising throughput, these heuristics aimed to ensure that the datacenter could handle high volumes of data traffic with minimal bottlenecks, thereby enhancing performance and resource utilisation. To evaluate their proposed heuristics, the authors conducted a numerical analysis, which involved testing the heuristics under various theoretical scenarios and configurations.

Li and Zhu [2023] addressed the challenge of reconfiguring VNs by developing an ILP formulation and a heuristic method. Their primary objective was to minimise both the operational cost and the resource usage during the reconfiguration process, ensuring that virtual networks could be efficiently adapted to changing conditions without incurring unnecessary expenses or resource waste. The ILP formulation provided an optimal solution by mathematically modeling the network constraints and objectives, while the heuristic offered a quicker, approximate solution suitable for larger or more dynamic networks where computational efficiency is a concern. To thoroughly evaluate the effectiveness of their proposed methods, the authors conducted simulations using two distinct network topologies, representing different types of VN configurations. They also analysed various weight parameters and request scenarios to understand how their formulation and heuristic would perform under different conditions and priorities. This comprehensive evaluation allowed them to demonstrate the flexibility and robustness of their approach in addressing VN reconfiguration challenges while minimising costs and resource consumption.

Table 3.5 provides an overview of various proposals for addressing the reconfiguration and migration problem, detailing aspects such as metrics, objectives, multi-objective formulations, approaches, and the problem where it is applied. It catalogues the different proposals, specifying the performance metrics assessed, the primary objectives targeted, and the multi-objective formulations employed to address competing requirements, although there is considerable missing data in the formulations, with several entries lacking detailed mathematical descriptions. The approaches' column describes the methods used, but also exhibits gaps where specific methodologies are not clearly defined.

Table 3.6 complements this by presenting information on the proposals' architecture, tools, applications, and the problem. It outlines the architectural designs and computational tools used for implementing the solutions and provides details on practical applications. However, this table also contains significant missing

data, particularly in the architecture and applications columns, which hinders a full understanding of how the proposals are structured and applied in real-world contexts. Both tables illustrate the range of research approaches while highlighting the need for more comprehensive information to fully evaluate and compare the proposed solutions.

Given the inherent complexity and dynamics of Reconfiguration and Migration problems within network systems, the primary focus tends to be on optimising resources and/or VNs in response to network errors or failures. These problems arise when there are disruptions or inefficiencies in the network, requiring adjustments to ensure continued functionality and performance. As a result, many approaches concentrate on a single objective, such as minimising downtime or maximising resource utilisation, to achieve quicker and more straightforward solutions.

This focus on single-objective optimisation allows for more immediate and practical results, as it simplifies the problem and reduces computational complexity. However, this approach also limits the scope of the solutions, potentially overlooking other important factors that might impact the overall system performance.

Moreover, a significant limitation in many of these studies is the reliance on numerical analysis rather than using simulation tools. Numerical analysis involves working with mathematical models and equations to derive results, but it may not capture the full range of real-world variables and dynamics. Simulators, on the other hand, provide a more comprehensive and realistic environment for testing and evaluating solutions under varied conditions.

Another limitation is the use of synthetic data, which is generated artificially and may not fully represent the complexities and variabilities of actual network data. While synthetic data can be useful for preliminary analysis, it may not accurately reflect real-world scenarios, leading to potential gaps in the applicability and robustness of the proposed solutions.

3.4 Analysis and Open Issues

In order to provide a comprehensive analysis of the existing works, we have separated the tables into two different categories: overview and simulation. The overview tables aim to provide a summary of the optimisation decision in the revised works, while the simulation tables aim to describe how the works are implemented and evaluated.

Tables 3.1, 3.3, and 3.5 aim to provide a summary of the most common metrics in the three described problems, what are the most common objectives, how the problems are formulated/transformed if they have multiple objectives, and what type of approaches are usually applied in Service Placement, Virtual Network Embedding, and Reconfiguration and Migration. Note that, even when the objectives are considered as constraints, we present them on the table. In case the authors do not define the objective as a main objective, i.e., only need to

Table 3.5.: Overview of Reconfiguration and Migration

Proposals	Metrics	Objectives	Formulation	Approaches
Tran and Timm-Giel [2013]	Profit	Maximisation	-	ILP Heuristic
Cerroni and Callegati [2014]	Downtime	Minimisation	-	ILP
Eramo et al. [2014]	Cost	Minimisation	-	ILP Heuristic
Dalvandi et al. [2015]	Cost	Minimisation	-	ILP Heuristic
Jiang et al. [2016]	Energy Consumption	Load Balancing	-	ILP Heuristic
Melo et al. [2016]	Bandwidth Reconfigurations	Minimisation	Weighted-sum	ILP
Khai et al. [2017]	Cost Hops	Minimisation	-	ILP
Eramo et al. [2017]	Energy Consumption	Minimisation	-	ILP Heuristic
Gao and Rouskas [2018]	Resource usage	Load Balancing	-	MILP Heuristic
Yang et al. [2018]	Resource Usage	Reliability	-	Exact Algorithm
Bi et al. [2018]	Cost	Minimisation	-	Heuristic
Medeiros et al. [2019]	Number of hops Bottleneck Channel interference	Minimisation	Weighted-sum	ILP Heuristic
Noghani et al. [2019]	Cost	Minimisation	-	NLP
Wang et al. [2019]	Profit	Maximisation	-	ILP Heuristic
Pozza et al. [2020]	Cost	Minimisation	-	Heuristic
Mouaci et al. [2020]	Cost	Minimisation	-	MILP
Padhy and Chou [2021]	Cost	Minimisation	-	ILP Heuristic
Li et al. [2022]	Energy Consumption Latency	Minimisation	Utility Function	ILP Heuristic
Hanauer et al. [2023]	Throughput	Maximisation	-	Heuristic
Li and Zhu [2023]	Resource Usage Cost	Minimisation	Weighted-sum	ILP Heuristic

Table 3.6.: Simulation in Reconfiguration and Migration

Proposals	Architecture	Tools	Applications	Problem
Tran and Timm-Giel [2013]	-	Custom Simulator	-	VNE
Cerroni and Callegati [2014]	Edge Computing	Numerical Analysis	-	SP
Eramo et al. [2014]	-	Numerical Analysis	-	VNE
Dalvandi et al. [2015]	Cloud Computing	Numerical Analysis	-	SP
Jiang et al. [2016]	-	Numerical Analysis	Smart Cities	SP
Melo et al. [2016]	-	Testbed	-	VNE
Khai et al. [2017]	-	Numerical Analysis	-	VNE
Eramo et al. [2017]	-	Numerical Analysis	-	VNE
Gao and Rouskas [2018]	-	Numerical Analysis	-	VNE
Yang et al. [2018]	-	Numerical Analysis	-	VNE
Bi et al. [2018]	Fog Computing	Mininet	Smart Cities	SP
Medeiros et al. [2019]	IoT	NS-3	-	SP
Noghani et al. [2019]	-	MATLAB	-	VNE
Wang et al. [2019]	Fog Computing	Numerical Analysis	-	SP
Pozza et al. [2020]	Network Slicing	Numerical Analysis	-	VNE
Mouaci et al. [2020]	-	Numerical Analysis	-	VNE
Padhy and Chou [2021]	Cloud Computing	Numerical Analysis	-	VNE
Li et al. [2022]	Edge Computing	EdgeSim	-	SP
Hanauer et al. [2023]	Cloud Computing	Numerical Analysis	-	SP
Li and Zhu [2023]	-	Numerical Analysis	-	VNE

meet a certain threshold, we do not present them in the tables.

Tables 3.2, 3.4, and 3.6 aim to describe the used architecture in the works, what tools are used as a simulation, and what kind of applications/services are presen-

ted in Service Placement, Virtual Network Embedding, and Reconfiguration and Migration.

It is possible to observe that in the case of Service Placement, most works consider energy consumption and latency. This usually occurs due to the fact that most devices in fog environments have limited available energy, and it is necessary to ensure that they are functional as long as possible. In more recent works, fairness is also being considered. This shows the need to avoid overusing and underusing devices.

When analysing Virtual Network Embedding works, the focus in energy consumption and latency is lower. Here, the main metric to optimise is resource usage. Each Virtual Network Embedding has different resources and may have several services to be executed in the whole Virtual Network Embedding; for example, a Service Function Chaining. Moreover, since usually there is a higher cost to migrate or reconfigure VNs, due to higher amount of needed resources, the main objectives are resilience and fairness.

Finally, in Reconfiguration and migration works, we described approaches for both Service Placement and Virtual Network Embedding. In the case of Service Placement, the focus continues to be energy consumption and latency. However, it is not common to observe fairness. Since these approaches need to be fast, to avoid interruption of connectivity, fairness is not considered due to the extra complexity. Regarding Virtual Network Embedding, the works mainly aim to maximise resource usage and take into account resilience.

In terms of simulation and applications, it is possible to observe that most works do not take into account realistic values, to help to provide a more complete evaluation, and most do numerical analysis as well. The usage of a simulator would improve the validation of the proposed approaches.

To conclude, we describe the main gaps in the previous analysed works.

- Usually, there is no comparison between optimal solution and heuristics.
- Most approaches consider some kind of transformation of the multi-objective problem into a single version.
- Hybrid approaches are usually not developed, i.e., to consider Service Placement, Virtual Network Embedding and Reconfiguration/Migration in the same framework.
- Several works in the VNE problem do not specify any type of architecture.
- There are many works that perform numerical analysis, instead of considering a simulator to validate their approaches.
- Most works do not consider realistic service values, just apply randomly generated parameters.

In the next chapters, formulations and heuristics for the Service Placement problem, the Virtual Network Embedding problem, and the Reconfiguration

and Migration problem, followed by an approach that aims to join all problems into a single framework to deal with the previously described issues are presented.

Chapter 4.

Service Placement

Contents

4.1. Description **51**
4.2. Mixed Integer Linear Programming Formulation . . **53**
4.3. A Heuristic **55**
4.4. Experimental Results **56**
4.4.1. Setup 56
4.4.2. Experimental Analysis 58
4.5. Summary **61**

In this chapter, a Mixed Integer Linear Programming (MILP) formulation and a heuristic to place services in a static two-layered network architecture are described. Two objectives are considered: maximisation of fog usage and maximisation of energy fairness among devices. The results were published in the 2019 IEEE Symposium on Computers and Communications conference (Godinho et al. [2019]).

The organisation of this chapter is as follows: Section 4.1 describes the scenario, Section 4.2 details the MILP formulation, Section 4.3 presents the proposed heuristic and Section 4.4 defines the experimental analysis and setup. Finally, Section 4.5 summarises the content of this chapter.

4.1 Description

The goal of this chapter is to provide a MILP model of the problem to solve a Service Placement (SP) problem in fog environments and a heuristic. In this work, a static two-layered network architecture, composed of a fog layer and a cloud layer was considered, as shown in Figure 4.1, based on the work of Skarlat et al. [2017]. The cloud is composed by computing units and storage, and a cloud-fog controller. This controller allows to establish a communication to the master nodes on the fog, which allows the cloud to receive the services placed there and decide which resource units are used. Note that we do not consider how the resources are placed on the cloud.

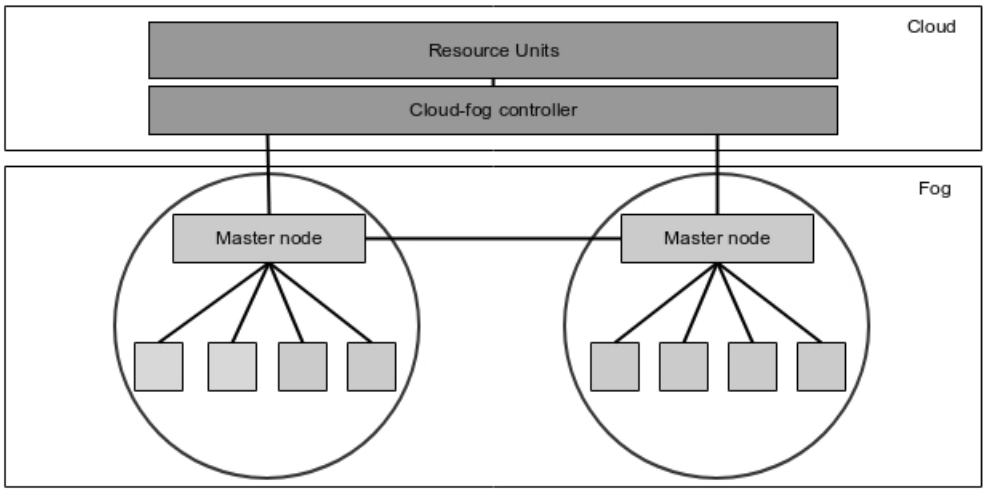


Figure 4.1.: Fog-Cloud Architecture

The fog is composed by a set of clusters, where each cluster behaves independently. This means that each cluster receives services independently and has no knowledge regarding the load of the other clusters. Furthermore, each cluster is connected to the closest cluster only, in order to avoid unnecessary delays, and is composed of a master and a set of fog nodes. The master acts as a controller, which decides the placement of the services that receives. There are four allocation possibilities: a) the service is sent to a fog node on the same cluster that is able to compute the service; b) the master computes the service; c) the service is sent to the neighbour cluster; d) the service is sent to the cloud. Each fog node can compute, at least, one of the three following types of services: sensing, processing, and/or actuating.

The following contributions of this chapter are as follows:

- A min-max Mixed Integer Linear Programming (MILP) formulation with two objectives: maximisation of resource usage and fairness of energy consumption.
- A heuristic approach to be used in online scenarios which provide results close to the optimal solution.

For the evaluation, a set of applications was generated, where each contains a set of services, with low, medium, and high deadlines. The approach was evaluated according to the distribution of services throughout the fog and cloud. The results show that the services are allocated fairly regarding their energy consumption throughout the system whereas lower priority services are sent to the cloud, while meeting the deadlines. Furthermore, the heuristic obtains solutions that are 1% away of the optimal solution, in average.

4.2 Mixed Integer Linear Programming Formulation

In this section, a MILP formulation is provided to define where the services are placed in the architecture described in the previous section (a non-linear formulation (Mixed Integer Programming (MIP)) is presented for the sake of comprehension. The linearised version can be obtained using standard techniques (Williams [2013]); see Appendix A.1). We are interested in optimising two objectives: maximise the usage of the fog and maximise fairness throughout the whole system (including cloud). A min-max formulation to obtain a compromise solution is considered, in order to deal with the possibility of conflicting objectives. Table 4.1 shows the parameters and variables used in the formulation.

Given n nodes, where the n-th is the cloud and the n-1-th is the neighbour cluster, m services, and a applications, the first objective maximises the usage of fog resources as follows (not including the cloud):

$$\max W(x) = \sum_{i=1}^{n-1} \sum_{j=1}^{m} c_{ij} x_{ij} p_j \tag{4.1}$$

where c_{ij} is a binary parameter that is 1 if a node i is able to compute a service j and 0 otherwise, p_j is a service priority, defined as $1/D_k$, where D_k is the deadline of an application k that contains service j, and x_{ij} is a binary variable that is 1 if a service j is computed in a node i and 0 otherwise. X denotes the set of feasible solutions.

In the second objective, the goal is to avoid having nodes that consume excessive energy while others do not consume any or lower energy when comparing to different devices, that is, to keep fairness in the overall system with respect to energy. We formalise this fairness objective as follows:

$$\min \max_{i=1,\dots,n} F_i(x) = \sum_{j=1}^{m} c_{ij} x_{ij} e_j \tag{4.2}$$

where e_j is the power energy required to execute a service j, which can be expressed as the necessary CPU usage of a service during its execution.

Note that the two objectives can be conflicting. The first objective intends to maximise the fog usage, while the second objective aims to distribute equally the services throughout the whole system. This implies that if a service is sent to the cloud, the first objective decreases, but if no service is sent to the cloud, the distribution may not be the most adequate and the value of the second objective increases. Thus, a min-max formulation in order to obtain a compromise solution is considered (see Section 2.4.1). We formalise the original problem as follows:

$$\min f(x) = \max_{i=1,\dots,n} (U_b - W(x), F_i(x)) \tag{4.3}$$

where U_b is an upper bound on the first objective function, obtained by solving

Table 4.1.: Parameters and variables for the Mixed Integer Linear Programming formulation

Parameter	**Description**
U_b	Upper bound on the first objective function
n	Set of nodes on the network
m	Set of services
a	Set of applications
S_k	Number of services in application k
c_{ij}	Binary parameter that is 1 if service j can be allocated in node i
p_j	Priority of service j
e_j	Energy consumption of service j
w	Waiting time on neighbor cluster
d_k	Deadline of application k
t_j	Time to execute service j
d_i	Delay between master and fog nodes
C_j	CPU needed to execute service j
M_j	Memory needed to execute service j
R_j	Storage needed to execute service j
CM_i	CPU capacity of node i
MM_i	Memory capacity of node i
RM_i	Storage capacity of node i
Variable	**Description**
x_{ij}	Binary variable that is 1 if service j is allocated in node i
y_k	Binary variable that is 1 if there is at least one service from application k allocated in the neighbor cluster

the formulation with only Equation 4.1 as a single-objective problem[1].

We consider the following constraints to describe the resources used by the services. Since the execution of those services is sequential, we are interested in ensuring that the node has enough resources to execute the services with higher consumption of CPU (C), memory (M), and storage (R). We formalise the constraint for the resources as follows:

$$\max_{j=1,\dots,m} \{c_{ij} x_{ij} C_j\} \leq CM_i, \qquad \forall i = 1, \dots, n-2 \tag{4.4}$$

$$\max_{j=1,\dots,m} \{c_{ij} x_{ij} M_j\} \leq MM_i, \qquad \forall i = 1, \dots, n-2 \tag{4.5}$$

$$\max_{j=1,\dots,m} \{c_{ij} x_{ij} R_j\} \leq RM_i, \qquad \forall i = 1, \dots, n-2 \tag{4.6}$$

where C_j, M_j, and R_j are the CPU, memory, and storage needed to execute service j and CM_i, MM_i, and RM_i are the available CPU on a node i, respectively.

[1]Note that both objective function values need to be on the same range. If the range is not the same, they are equalised (White [1982]).

Each application has a deadline to be met, which is composed by the overall time to execute its services. We express this as a constraint as follows, for each application $k = 1, \ldots, a$:

$$T_k(x) = wy_k + \sum_{j=1}^{S_k} \sum_{i=1}^{n} c_{ij} x_{ij} (2d_i + t_j) \leq D_k \tag{4.7}$$

where w is the waiting time on the neighbour cluster, S_k indicates the number of services on an application k, d_i is the delay between the master and the fog nodes, neighbour or cloud, t_j is the expected time to execute a service j, and y_k is a binary variable that is 1 if there is a least one service allocated on a neighbour and 0 otherwise.

In order to update the total waiting time correctly, we need to identify if a service from an application has been allocated to the neighbour cluster:

$$N(x) = \sum_{j=1}^{S_k} c_{n-1,j} x_{n-1,j} \tag{4.8}$$

$$y_k \leq N(x), \qquad \forall k = 1, \ldots, a \tag{4.9}$$

$$S_k y_k \geq N(x), \qquad \forall k = 1, \ldots, a \tag{4.10}$$

where $c_{n-1,j}$ is the binary parameter to indicate if the neighbour cluster is able to execute service j and $x_{n-1,j}$ is a binary variable that is 1 if the service j is executed on the neighbour cluster and 0 otherwise. Constraints 4.9 and 4.10 aim to define the value of y_k by acting as a logical OR. If at least one service of an application k is sent to the cloud, then the overall execution time of the application needs to take into account the waiting time w.

In order to ensure that the services are executed only once, and on a node capable of performing such services, we consider the following constraints:

$$\sum_{i=1}^{n} c_{ij} x_{ij} = 1, \;\; \forall j = 1, \ldots, m \tag{4.11}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \;\; \forall j = 1, \ldots, m \tag{4.12}$$

4.3 A Heuristic

In this section, we propose an heuristic approach to obtain an approximate solution as fast as possible, since the single objective SP problem is NP-complete (Huang et al. [2009]).

Algorithm 4.1 shows the proposed approach. The aim is to choose the services to place by decreasing order of priority (line 2). An empty array with the size equal to the number of nodes (line 1) is created, including the master node and the neighbour cluster. A priority queue is also created, to sort the nodes by their energy consumption (line 3). Next, the algorithm loops throughout

Algorithm 4.1: Max fog usage with fairness heuristic

Input: Set S containing the services, n nodes
Output: Set N' containing the location of the allocated services

```
1  N' ← array initialised to 0 with size n
2  Sort S by decreasing order of priority
3  Insert all nodes into an energy priority queue (PQ)
4  forall service s ∈ S do
5      Allocated ← false
6      node ← Top(PQ)
7      while Allocated = false or node.next ≠ null do
8          if node.resources ≥ s.resources and node.time ≤ s.deadline
             then
9              Allocate(node, s)
10             Allocated ← true
11         node ← node.next
12     if Allocated = false then
13         Allocate(cloud, s)
14 return N'
```

the services (lines 4-13). In this loop, a sorted list by energy consumption is generated with nodes that are capable of computing the service s (line 6). This allows the algorithm to choose the node with less energy consumption, in order to not overload the system. Since it is not ensured that the first node contains enough resources to compute the service (CPU, RAM or storage), or if the task can be completed before the deadline arrives (for example, since the neighbour has an additional deployment, the task may not be computed in time), the algorithm searches throughout the nodes until one is found (lines 7-11). Otherwise, the service is sent to the cloud (lines 12-13). Its time complexity is $\mathcal{O}(t_{|S|} + |S|(n + t_n))$, where $t_{|S|}$ and t_n indicate the complexity of sorting the set S of services and of n nodes, respectively.

4.4 Experimental Results

In this section, the experimental setup and analysis of our approaches are described. In order to implement them, the *iFogSim* toolkit (Gupta et al. [2017]) is used as a fog simulator. The linearised formulation of the MILP formulation was implemented in the *SCIP* solver (Bestuzheva et al. [2021]) with *JSCIPOpt* (JSCIPOpt [2016]).

4.4.1 Setup

The experimental work consisted on placing services from 1 to 15 applications, with three types of deadlines: low deadline (30-60s), medium deadline (120-180s), and high deadline (240-360s). Each application consists of 1-3 sensing services, 1-4 processing services, and one actuating service. The deadlines,

Table 4.2.: Type of services and their demands

Service type	**CPU**	**RAM**	**Storage**	**Expected time**
Sensing	20-50	10-30	10-50	500-1000
Processing	50-100	30-70	50-200	100-500
Actuating	20-50	10-30	10-50	500-1000

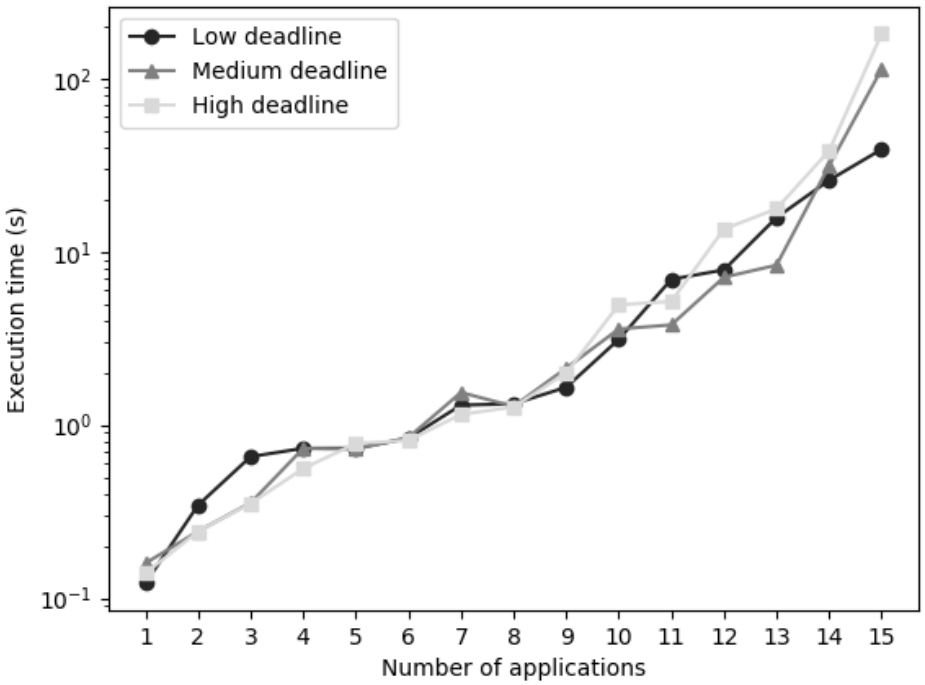


Figure 4.2.: Time needed to execute the applications with different deadlines

number of services for application, resource needs, and expected times to execute have been generated randomly according to a uniform distribution. Table 4.2 indicates the resource needs and expected times for each service type. Note that CPU is defined in MIPS, RAM and storage in Mb, and the expected time to execute in ms.

We assume that fog nodes have a single core CPU with 250 MIPS, 256Mb of memory and 4Gb of storage. The master node is composed by a single core CPU with 1000 MIPS, 512Mb of memory and 8Gb of storage (Skarlat et al. [2017]). Furthermore, 7 sensing nodes, 10 processing nodes, and 3 actuating nodes were considered. The master node, the neighbour cluster, and the cloud can only execute processing services. Moreover, we assume that 300ms is spent to communicate from the master to each fog node on the cluster, 500ms to communicate to the neighbour cluster, and 1000ms to communicate to the cloud. Finally, the neighbour cluster has a waiting time of 120s.

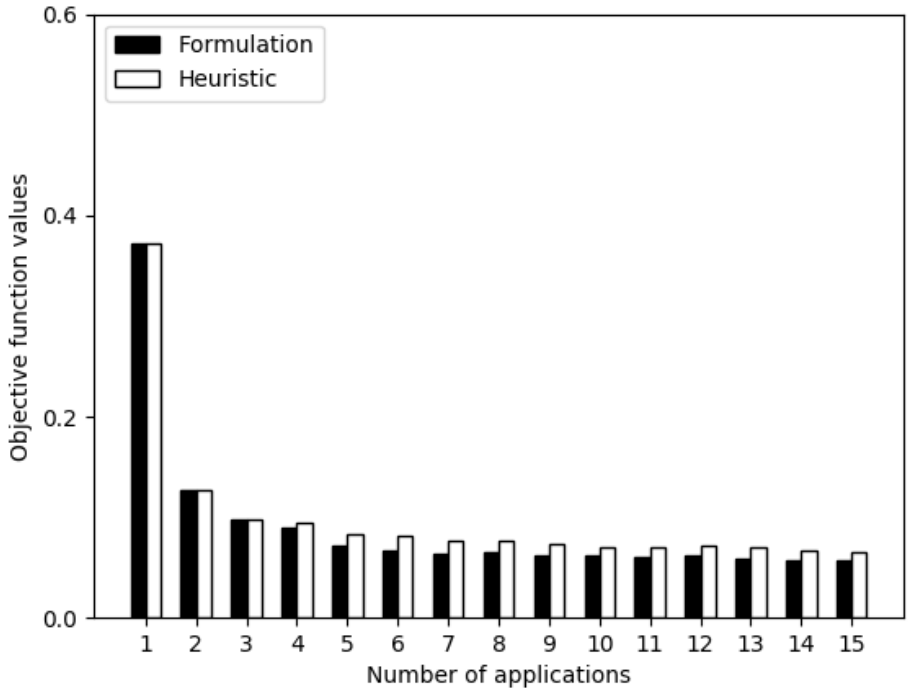


Figure 4.3.: Objective function values, formulation and heuristic, with high deadline

4.4.2 Experimental Analysis

In Figure 4.2, the time needed for the solver to solve the problem for the various deadlines and applications is provided, in log scale. As it is possible to observe, the formulation presents an exponential behaviour in terms of running time. Therefore, in order to use our work in practical scenarios, there is a need for an heuristic.

Figure 4.3 and 4.4 show the objective function values of the formulation and heuristic, and the objective function values for the lexicographic optimal solutions for each objective. The formulation and heuristic objective values increase with the decrease of the number of applications, since the values used to normalise the objectives depend on the total number of services. Therefore, with a higher number of applications, the difference between the highest value and the optimal increases. As expected, the lexicographic solutions have worse results when compared to the formulation and heuristic, since the lexicographic solutions present conflicting behaviour. Moreover, the formulation provides the lower value, indicating that it obtains the best compromise solution between the two objectives. Regarding the heuristic, it is possible to observe that it obtains the optimal solution with lower number of applications, however with a larger number of applications it presents higher objective function values than the formulation. Despite this fact, the difference between the formulation and heuristic is approximately, in average, 1%, suggesting that it obtains solutions

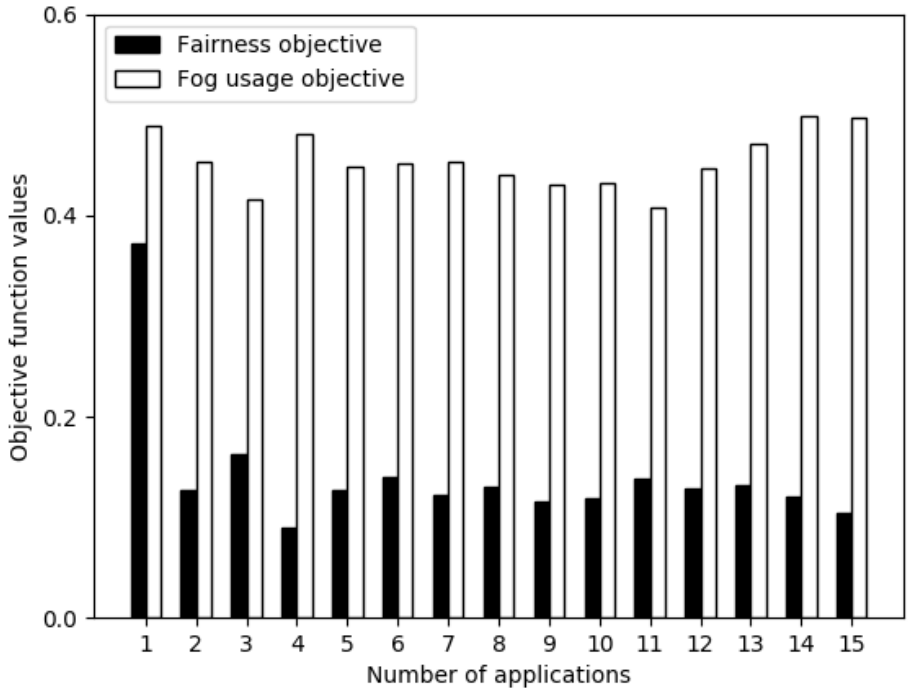


Figure 4.4.: Objective function values, lexicographic solutions, with high deadline

that are very close to the optimal. When considering low and medium deadline, the heuristic obtains results within a difference, in average, of 1% and 0.5%, respectively. By analysing the objective values of the fairness objective, we observe higher values, around 10% worse than the formulation. This happens because, despite the fact that the best required energy distribution has been obtained, the distribution regarding the priority is not optimal, which implies that higher priority services are sent to the cloud. The opposite happens when analysing the fog usage objective. All the services have been allocated on the fog, yet they have been badly distributed. Therefore, only some nodes execute those services, which results in high energy consumption and values about 50% worse than the formulation. Therefore, the fairness objective obtains worse values regarding fog usage, while the latter obtains worse values on energy consumption. By joining both objectives, we are interested on performing most services on the fog, as distributed as possible, while releasing some fewer priority services to the cloud, in order to not overload the fog.

In Tables 4.3 and 4.4, we present the distribution of energy consumption and services throughout the system, respectively, regarding the total energy required for the applications and existing services. We considered the higher demand scenario, 15 applications with 85 services, and prepared 7 different setups regarding deadlines executed on the MILP formulation (F) and heuristic (H). A setup with low deadlines (Low), medium deadlines (Medium), high deadlines (High), equally distributed deadlines (Equal), higher number of low deadlines

Table 4.3.: Energy required regarding CPU usage, 15 applications (85 services) in percentage

Deadlines	**Cloud**		**Master**		**Cluster**		**Neighbour**	
	F	***H***	***F***	***H***	***F***	***H***	***F***	***H***
Low	4.5	3.8	4.3	5.2	91.2	91.0	0.0	0.0
Medium	5.6	3.4	3.3	3.6	85.9	88.4	5.2	4.6
High	5.0	3.7	5.1	4.0	88.9	88.6	1.0	3.7
Equal	5.7	3.5	5.7	5.2	83.5	87.9	5.1	3.4
MLow	4.2	6.2	5.4	3.7	88.8	85.7	1.6	4.4
MMedium	5.6	3.7	5.5	4.1	86.5	88.4	2.4	3.8
MHigh	5.7	4.9	5.0	3.7	85.3	87.8	4.0	3.6

Table 4.4.: Service distribution for 15 applications (85 services) in percentage (Number of services)

Deadlines	**Cloud**		**Master**		**Cluster**		**Neighbour**	
	F	***H***	***F***	***H***	***F***	***H***	***F***	***H***
Low	5.9	3.5	3.6	7.1	90.5	89.4	0.0	0.0
Medium	5.9	3.5	3.5	3.5	85.9	87.1	4.7	5.9
High	5.9	2.4	5.9	5.9	85.9	88.2	2.3	3.5
Equal	9.4	4.7	9.4	4.7	77.7	87.1	3.5	3.5
MLow	8.2	7.1	8.3	4.7	82.3	83.5	1.2	4.7
MMedium	9.4	4.7	8.2	5.9	80.0	85.9	2.4	3.5
MHigh	10.6	4.7	9.4	4.7	77.6	84.7	2.4	5.9

(MLow), higher number of medium deadlines (MMedium), and higher number of high deadlines (MHigh).

4.4.2.1 Energy consumption

In Table 4.3, the formulation and heuristic present similar values regarding energy consumption for the cloud. The same is observed for the master and cluster categories. However, while observing the neighbour values in the Low and MLow setups, most (or all) applications cannot be sent to the neighbour cluster, which provides an underusage of it. When the Equal setup is observed, there is a similar distribution of both energy consumption and service distributions on the formulation and heuristic since there is an equal division of deadlines between the services. In the case of the Medium and MMedium setups, more distribution to the neighbour is observed.

4.4.2.2 Service distribution

In Table 4.4, it is possible to observe that the formulation provides better distribution of services according to their energy consumption, while the heuristic distributes better the number of services. Since the heuristic distributes services with a greedy approach, it is possible to place some services in different nodes which, considering their energy consumption, would be better placed in

the same node, in order to allow other higher energy consuming services to be placed in less used nodes. Moreover, despite the fact that the formulation sends more services to the cloud than the heuristic, the priority of those services is lower, in general, than the ones sent by the heuristic.

In order to compare the proposed formulation to the original formulation in Skarlat et al. [2017], we implemented both into the solver. The objective values obtained are similar to the fog usage objective as shown in Figure 4.4, yet there is a higher distribution of services throughout the system, while placing most services on the fog. This allows the formulation to be fair regarding the energy required in the nodes on the system, which is specially important for battery powered devices.

4.5 Summary

In this chapter, a solution to the Service Placement problem is provided. A static two-layered network architecture, composed by fog and cloud was considered. The fog consists of clusters, each containing a master node and a set of fog nodes. In order to place the services, two objectives were considered: maximisation of fog usage and maximisation of energy fairness throughout the system, using a min-max formulation. For this, a Mixed Integer Linear Programming formulation and a heuristic were proposed, that model both objectives taking into account the constraints of the devices and the requirements of the services. Both approaches were compared using a set of randomly generated applications composed by different deadlines and services. While the formulation distributes better the services regarding energy consumption, the heuristic distributes better the number of services. Moreover, both approaches provide similar usage of the fog.

Chapter 5.

Reconfiguration and Migration

Contents

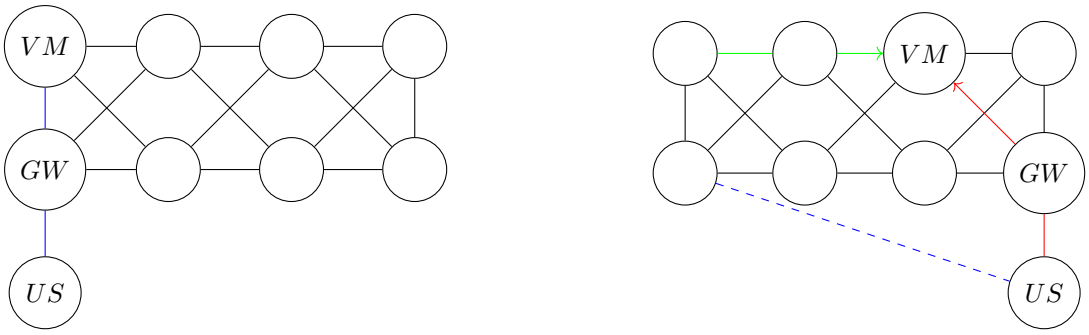
5.1. Description **63**
5.2. Mixed Integer Linear Programming Formulation . . **65**
5.3. A Heuristic **69**
5.4. Experimental Results **71**
5.4.1. Setup 71
5.4.2. Experimental Analysis 72
5.5. Summary **77**

This chapter presents a Mixed Integer Linear Programming (MILP) formulation and a heuristic to address the data communication-migration problem, focusing on two key objectives: minimising energy consumption across the network and reducing overall latency. The results were published in the 2020 IEEE International Symposium on Network Computing and Applications conference (Godinho et al. [2020]).

The chapter is structured as follows: Section 5.1 describes the environment, Section 5.2 outlines the MILP formulation, Section 5.3 introduces the proposed heuristic, and Section 5.4 details the experimental analysis and setup. Finally, Section 5.5 provides a summary of the chapter's content.

5.1 Description

This chapter aims to present a model and a fast algorithm for addressing a reconfiguration problem in fog environments. A fog scenario where users connect to the closest device is considered. Figure 5.1 shows an example of a connection before and after mobility occurs. It is assumed that each user (US) has its own Virtual Machine (VM) previously allocated with some placement algorithm and is connected to the closest device (GW) that routes the data (Figure 5.1a, blue lines). The focus is on the scenario in which a user has moved to a new location and cannot connect to the previous device or the latency is too high and the connection cannot be established. In Figure 5.1b, the blue dotted line indicates



(a) Before Mobility Occurs

(b) After Mobility Occurs

Figure 5.1.: Network with one user

the previous connection. After connecting to a new GW, the reconfiguration of data communication (green lines) and migration (red lines) is performed. In each time unit, it is assumed that there are several users that move to a different location and it may be necessary to perform reconfiguration of the connection. It is also considered that, even if it is possible to freely create new VMs, there is still the need to define where the new VMs will be created and to migrate the data from the previous VMs. Since the reconfiguration is only made after mobility occurs, it is possible to generalise for any kind of mobility pattern. Moreover, despite focusing on a general type of network, it is possible to consider a real scenario by specifying the requirements of the VMs to be used on the parameters. The new path to be obtained takes into account Quality of Service (QoS) requirements, which may deem necessary to perform some migration of the VMs to a new location, closer to the user. However, this migration may have a larger cost than only data communication. Thus, one of the following two possibilities can occur: 1) a new data path to access the resources is found that satisfies QoS requirements; 2) a migration closer to the user is performed and a path to the new location of the VM is found.

The following contributions are highlighted:

- Data communication-migration path strategy to obtain a new path. Each user has its own connection to a node on the network with its own VM in order to offload services. However, it may not be possible to use the previous connection after the user changes location due to QoS requirements or unreachability. Thus, the aim is to obtain a new path that considers both migration of the resources and data communication to the new location.
- Min-max Mixed Integer Linear Programming (MILP) formulation to avoid definition of preferences and to obtain a compromise solution that minimises the worst case. In this problem, we assume that there is no explicit definition of preferences for each objective.
- Development of a heuristic to provide approximate solutions. Since the formulation is not suitable for large scenarios, given the time requirements of these type of applications, we develop a heuristic based on the MILP formulation.

The proposed approaches are evaluated in terms of energy consumption and latency as well as the number of user rejections and number of hops. As baseline, a first-fit heuristic that optimises the number of hops while considering the formulation constraints is provided.

5.2 Mixed Integer Linear Programming Formulation

In this section, a MILP formulation to find a single path data communication and migration after a set of users have changed location is detailed (a non-linear formulation (Mixed Integer Programming (MIP)) is presented for the sake of comprehension. The linearised version can be obtained using standard techniques (Williams [2013]); see Appendix A.2). The aim is to minimise two objectives: energy consumption and latency. A min-max formulation which allows to obtain a compromise solution to deal with the possibility of conflicting objectives is used. This way, it is possible to obtain a solution that takes into consideration both objectives at the same time and that minimises the worst case, a compromise solution. Table 5.1 shows the parameters and variables used in the formulation.

Given n nodes and U users, the first objective of the MILP model aims to minimise the overall energy consumption from data communication and migration of the users:

$$\min E(x, x') = \sum_{u=1}^{U} \sum_{i=1}^{n} \sum_{j=1}^{n} e_{iju} x_{iju} + e'_{iju} x'_{iju} \tag{5.1}$$

where x_{iju} and x'_{iju} are binary variables that are 1 if a link (i, j) is used on data communication and migration links by user u, respectively, and 0 otherwise.

Parameters e_{iju} and e'_{iju} correspond to the energy consumption values of using each link by each user, calculated as follows in W/s (Lera et al. [2019]):

$$e_{iju} = d_{iju} e''_{ij} \tag{5.2}$$

$$e'_{iju} = d'_{iju} \bar{e}''_{ij} \tag{5.3}$$

where e''_{ij} and $\bar{e}''_{ij}$ are the energy consumption of link (i, j) per second when using data communication and migration, respectively, and d_{iju} and d'_{iju} are the time to communicate and to migrate on each link (i, j) by each user u, respectively. Note that these parameters are known a priori.

The second objective aims to minimise the overall latency. It is formulated as follows:

$$\min L(x, x') = \sum_{u=1}^{U} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{iju} x_{iju} + d'_{iju} x'_{iju} \tag{5.4}$$

Table 5.1.: Parameters and variables for the Mixed Integer Linear Programming formulation

Parameter	**Description**
n	Set of nodes on the network
U	Number of users on the network
e_{iju}	Energy consumed on link (i, j) by user u on data communication
e'_{iju}	Energy consumed on link (i, j) by user u on migration
d_{iju}	Time to communicate on link (i, j) by user u
d'_{iju}	Time to migrate on link (i, j) by user u
b_u	Bandwidth usage for data by user u
b'_u	Bandwidth usage for migration by user u
bw_{ij}	Bandwidth capacity of link (i, j)
PD_{ij}	Propagation delay on link (i, j)
MS_u	Message size for data communication by user u
MS'_u	Message size for migration by user u
V_{iu}	Conservation flow matrix for data
V'_{iu}	Conservation flow matrix for migration
C_u	CPU usage of user u
M_u	Memory usage of user u
S_u	Storage usage of user u
CM_i	CPU capacity of node i
MM_i	Memory capacity of node i
SM_i	Storage capacity of node i
T	Upper bound on total time for data
T'	Upper bound on total time for migration
Variable	**Description**
x_{iju}	Binary variable that indicates if a link is used for data on a user
x'_{iju}	Binary variable that indicates if a link is used for migration on a user
y_{iu}	Binary variable that indicates if a VM from user u is on node i
s_u	Continuous variable to indicate the data starting time of user u
s'_u	Continuous variable to indicate the migration starting time of user u
δ_{uv}	Slack variable to indicate if user u starts before user v in data
δ'_{uv}	Slack variable to indicate if user u starts before user v in migration

The following equation is used to calculate the delay of a link (Lera et al.

[2019]):

$$d_{iju} = \frac{MS_u}{b_u} + PD_{ij} \tag{5.5}$$

$$d'_{iju} = \frac{MS'_u}{b'_u} + PD_{ij} \tag{5.6}$$

where b_u and b'_u represent the usage of bandwidth of data communication and migration by user u respectively, PD_{ij} corresponds to the propagation delay of link (i, j) and MS_u and MS'_u are the size of the message for data communication and migration, respectively. Note that these parameters are known a priori.

The min-max formulation is formalised as follows:

$$\min f(x, x') = \max\left(E(x, x'), L(x, x')\right) \tag{5.7}$$

Equation 5.7 aims to find the minimum of the maximal value of both objectives, from the set of possible solutions, i.e, to minimise the worst case scenario[1].

It is also assumed that each user has a deadline, according to the type of profile associated to the user, such as streaming and VoIP or messaging and e-mail data. In this constraint, we aim to ensure that the deadline D_u of each user u is guaranteed. Since it is considered that the migration and data communication paths are disjoint, the migration, if exists, starts at the same time as the data from the user to the new location. It is formulated as follows:

$$\max\left(\sum_{i=1}^{n}\sum_{j=1}^{n} d_{iju}x_{iju} + s_u, \sum_{i=1}^{n}\sum_{j=1}^{n} d'_{iju}x'_{iju} + s'_u\right) \leq D_u \tag{5.8}$$

for all $u = 1, \ldots, U$, where s_u and s'_u are continuous variables and indicate the data communication and migration starting time of user u, respectively.

Moreover, the aim is to ensure that the bandwidth constraints of each link B_{ij} are not exceeded:

$$\max_{u=1,\ldots,U}(b_u x_{iju} + b'_u x'_{iju}) \leq B_{ij}, \quad \forall i, j = 1, \ldots, n \tag{5.9}$$

where b_u and b'_u represent the bandwidth usage when data communication or migration, respectively.

The goal of the following constraint is to ensure that the resources are migrated to a capable node. It is formulated as follows:

$$\sum_{u=1}^{U} C_u y_{iu} \leq CM_i, \qquad \forall i = 1, \ldots, n \tag{5.10}$$

[1]Note that both objective function values need to be on the same range. If the range is not the same, they are equalised (White [1982])

$$\sum_{u=1}^{U} M_u y_{iu} \leq MM_i, \qquad \forall i = 1, \ldots, n \tag{5.11}$$

$$\sum_{u=1}^{U} S_u y_{iu} \leq SM_i, \qquad \forall i = 1, \ldots, n \tag{5.12}$$

where y_{iu} is a binary variable that takes value 1 if the resource from user u is migrated to i and 0 otherwise. Parameters C_u, M_u and S_u represent the usage of CPU, memory and storage of user u, respectively, and CM_i, MM_i and SM_i represent the available CPU, memory and storage of node i.

The following constraint allows the definition of two different paths, one for data communication and one for migration, in order to ensure flow conservation on each of them.

$$\sum_{j=1}^{n} x_{iju} - \sum_{j=1}^{n} x_{jiu} = V_{iu}, \;\; \forall i = 1, \ldots, n, u = 1, \ldots, U \tag{5.13}$$

$$\sum_{j=1}^{n} x'_{iju} - \sum_{j=1}^{n} x'_{jiu} = V'_{iu}, \;\; \forall i = 1, \ldots, n, u = 1, \ldots, U \tag{5.14}$$

where V_{iu} takes value 1 if the node i is the user, which corresponds to the source or the value of the variable y_{iu}, which can be a transit, if its value is 0, or target node, if its value is 1 (in this case its value is multiplied by -1 due to the fact that a target node needs to have a value of -1 on the flow constraint). This allows us to have a target such that the best location for the migration of the resources is found, according to the objective function values, i.e., we have a fixed source, but the new location of the resources depends on the variable y_{iu}. Similarly, V'_{iu} is 1 if the source is the original location of the resources of the user or the value of the variable y_{iu} multiplied by -1, since the target is the same in order to have a connected path.

To guarantee that two disjoint paths are found, we consider the following constraint:

$$x_{iju} + x'_{iju} \leq 1, \;\; \forall i, j = 1, \ldots, n, u = 1, \ldots, U \tag{5.15}$$

In order to ensure that the resources are placed in a node connected on both paths, i.e., to guarantee that both paths are connected by the same target, it is necessary to have one link of each path connected to it. With this constraint, variable y_{iu} only takes value 1 if the minimum number of connected links from each path is one. We formulate it as follows:

$$y_{iu} \leq \min\left(\sum_{j=1}^{n} x_{iju} + x_{jiu}, \sum_{j=1}^{n} x'_{iju} + x'_{jiu}\right) \tag{5.16}$$

for all $i = 1, \ldots, n, u = 1, \ldots, U$.

Moreover, we assume that each link provides data communication with one user at a time. Thus, we include scheduling constraints into the MILP formulation to avoid having a link used by more than one user at any given time. The scheduling

constraints ensure that the starting time for a user is equal or greater than the ending time of the previous user. We define an index mapping $(i, j) \mapsto e$, where each e represents a link (i, j). We formulate these scheduling constraints as follows (see H. Papadimitrou and Steiglitz [1998]):

$$s_v + d_{eu} - s_u \leq T(1 - \delta_{uv} + 2 - x_{eu} - x_{ev}), \quad \forall e \in E, u \neq v, \quad u, v = 1, \ldots, U \tag{5.17}$$

$$s_v - s_u \leq T\delta_{uv}, \quad \forall u \neq v, u, v = 1, \ldots, U \tag{5.18}$$

$$\delta_{uv} + \delta_{vu} = 1, \quad \forall u < v, u, v = 1, \ldots, U \tag{5.19}$$

$$s'_v + d'_{eu} - s'_u \leq T'(1 - \delta'_{uv} + 2 - x'_{eu} - x'_{ev}), \quad \forall e \in E, u \neq v, \quad u, v = 1, \ldots, U \tag{5.20}$$

$$s'_v - s'_u \leq T'\delta'_{uv}, \quad \forall u \neq v, u, v = 1, \ldots, U \tag{5.21}$$

$$\delta'_{uv} + \delta'_{vu} = 1, \quad \forall u < v, u, v = 1, \ldots, U \tag{5.22}$$

where s_u and s'_u are integer variables to indicate the starting time of user u of data communication and migration, respectively, δ_{uv} and δ'_{uv} are slack binary variables to indicate whether the communication and migration of a user u start before user v, respectively, and T and T' are upper bound parameters.

Finally, we have a constraint to ensure that the resources for each user are placed in only one node.

$$\sum_{i=1}^{n} y_{iu} = 1, \ \ \forall u = 1, \ldots, U \tag{5.23}$$

5.3 A Heuristic

Since the presented formulation is not suitable for large scenarios, we present a heuristic to obtain a high quality sub-optimal solution as fast as possible.

Algorithms 5.1, 5.2, and 5.3 describe the heuristic to obtain a joint data communication and migration path for each user with Equation 5.7 as objective function. Procedure *Heuristic* finds a new path after the user reaches a new position, in a graph G. The required data structures are two lists: P contains the paths for each user and N contains the new nodes where the resources will be allocated (lines 2-3). First, for each user, we obtain a path from the user to the original location of the resources (line 5) with the energy and latency values of data communication, and a path backwards (line 6) with the energy and latency values of resource migration. The path is calculated by procedure *Path*. The method to obtain a path is similar to Dijkstra's algorithm, where the difference is to include a check for available bandwidth on each link. If there is no path, then the user is rejected, since it is not possible to reconfigure the user connection (lines 7-8). Otherwise, the heuristic checks whether exists, at least,

Algorithm 5.1: Single path data communication and migration heuristic

Input: Graph G, n nodes, U users
Output: List P containing paths and list N containing new nodes

```
1  P ← ∅
2  N ← ∅
3  foreach user u ∈ U do
4      p_1 ← Path(G, u.source, u.target)
5      p_2 ← Path(G, u.target, u.source)
6      if p_1 and p_2 do not exist then
7          p_f ← ∅
8      else
9          if Intersect(p_1, p_2) = true then
10             nodes ←Intersect_node(p_1, p_2)
11             p_f, node ←Join_path(p_1, p_2)
12             N.add(u, node)
13         else
14             p_f ←Min_obj_value(p_1, p_2)
15         if Deadline(u, p_f) > u.deadline then
16             p_f ← ∅
17     P.add(u, p_f)
18 return P, N
```

Algorithm 5.2: Path procedure for data communication and migration

Input: Graph n, source s, target t
Output: List N containing new nodes

```
1  N ← empty list to add nodes of the path
2  Push s into Q with distance 0
3  while Q is not empty do
4      i ← Q.pop()
5      N.append(i)
6      if i = t then
7          break
8      foreach i, j = 1, ..., n do
9          if Bandwidth(i, j) ≤ b then
10             d ←obj_value(i)+dist(i, j)
11             if d <obj_value(j) then
12                 obj_value(j)← d
13                 if v ∈ Q then
14                     Q.update(j)
15                 else
16                     Q.push(j)
17 return N
```

one common node on both paths (line 10, procedure *Intersect*).

The node found in *Intersect_node* (line 11) provides intersection points between

Algorithm 5.3: Intersect procedure for two paths

Input: Paths $p1$ and $p2$
Output: True if exists intersection, False otherwise

```
1 foreach node i ∈ p1 do
2     if i ∈ p2 then
3         return True
4 return False
```

Table 5.2.: Parameter values in the network

Link	Values
Bandwidth	[100-1000]Mbps
Propagation Delay	[100-1000]ms
Energy Consumption	[1-10]W/s

Node	Values
CPU	[5-10]MIPS
RAM	[1-5]Gb
Storage	[10-100]Gb

both paths. For example, if we have $p_1 = (1, 3, 4, 5)$ and $p_2 = (5, 2, 3, 1)$, a possible merged path would be $p_f = (1, 3, 2, 5)$ (line 12, procedure *Join_path*), where node 2 is the new location of the resources. If there exists more than one common node, then the node that provides smaller objective function value on the overall path will be chosen (Equation 5.7). If there are no common nodes, then the chosen path is the one with the smallest objective function value (line 15). Finally, the scheduling constraints are checked (line 16, procedure *Deadline*). Since each link may be used by more than one user, it is necessary to take into account the waiting time, which may overcome the QoS time requirements of the user. If that happens, then the reconfiguration is rejected for that user. Its time complexity is $\mathcal{O}(|U|(|E| + n \log n))$ since it needs to iterate through each user and find two different paths (data communication and migration) which are joined afterwards to define the new path.

5.4 Experimental Results

This section describes the experimental analysis. The approaches were implemented on the *Yet Another Fog Simulator (YAFS)* simulator (Lera et al. [2019]) and used the *SCIP* solver (Bestuzheva et al. [2021]) with *PySCIPOpt* (Maher et al. [2016]) to implement the linearised MILP formulation.

5.4.1 Setup

The experimental scenario considers a set of users on a given topology with their own VMs. It is assumed that these users moved from their original position to a different location and the original location of the resources is not reachable.

Different topology sizes for each experimental case were considered, with uniformly random generated values for the network and the users (Lera et al. [2019]), as described in Tables 5.2 and 5.3, respectively.

Moreover, we implemented a first-fit algorithm, called *baseline*, which finds a

Table 5.3.: Parameter values of the users

Link	Values
Deadline	[100-200]s
Communication Bandwidth	[1-5]Mbps
Migration Bandwidth	[10-20]Mbps
Communication Message Size	[5-10]Mb
Migration Message Size	[20-50]Mb

Node	Values
CPU	[100-500]IPS
RAM	[10-100]Mb
Storage	[100-500]Mb

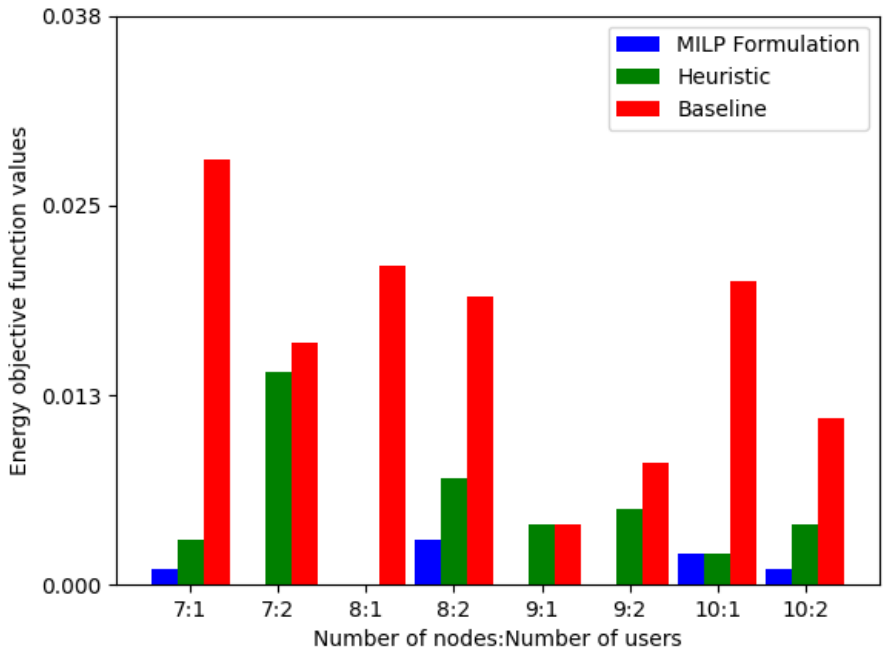


Figure 5.2.: Values of energy obtained by the Mixed Integer Linear Programming formulation, heuristic and baseline, with 7-10 nodes and 1 to 2 users

path that minimises the number of hops and aims to migrate the resources to the first available node, if possible, while taking into account the constraints defined by the MILP formulation. If there is no node with available resources on the path or the path is not found, the user is rejected.

5.4.2 Experimental Analysis

The approaches were evaluated as follows: 1) comparison of energy and latency objective function values; 2) comparison of rejected users with different topology sizes and users; 3) comparison of different energy and latency link values on data communication and migration to evaluate the number of hops with different topologies. Due to the prohibitive execution time of the MILP formulation, it was only considered in 1), which had smaller test cases.

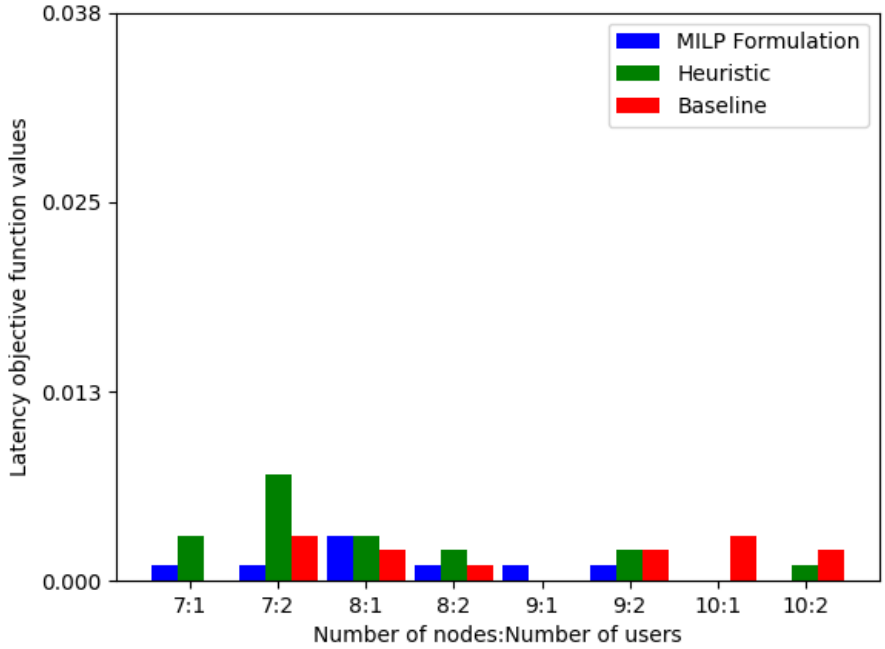


Figure 5.3.: Values of latency obtained by the Mixed Integer Linear Programming formulation, heuristic and baseline, with 7-10 nodes and 1 to 2 users

5.4.2.1 Formulation and heuristic

In order to compare the results of the formulation, heuristic and baseline, a network topology with 7-10 nodes and 1 to 2 users for each topology size are considered. Figures 5.2 and 5.3 show the energy and latency for the data communication-migration with different topology sizes and users, while Figure 5.4 shows the objective function values (maximum value between energy and latency) of the scenarios. It is important to notice that the objective function value is calculated as the worst value considering both objectives, as described in Equation 5.7. Table 5.4 shows the execution times for each objective individually (both minimisation and maximisation), as reported by the min-max formulation, the heuristic and the baseline. Note that, in order to obtain the upper and lower bounds for the min-max for each objective, it is necessary to minimise and maximise each one individually. In most scenarios, the heuristic obtains values closer to the optimal, both in terms of energy and latency, while the first-fit approach obtains the worst results. This indicates that the heuristic is able to achieve a better compromise solution than the first-fit approach in all scenarios. Moreover, the first-fit has the largest energy consumption values in all scenarios. Since this approach finds the first available path for data communication and migration for each user, this path may not be the best in terms of energy or latency.

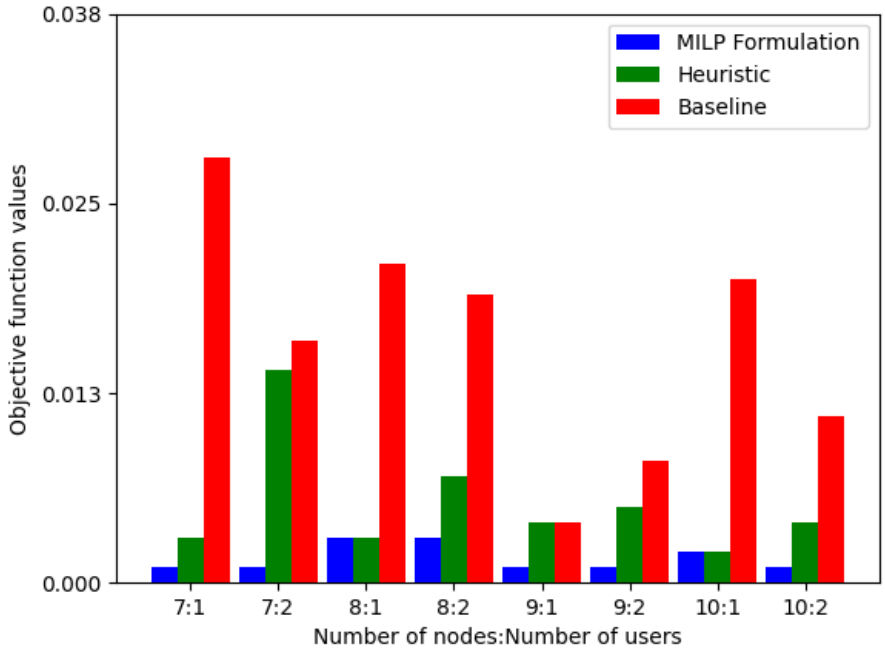


Figure 5.4.: Objective function values (Equation 5.7) obtained by the Mixed Integer Linear Programming formulation, heuristic and baseline, with 7-10 nodes and 1 to 2 users

Table 5.4.: Execution times (in seconds) for the Mixed Integer Linear Programming formulation (individual objectives (E=Energy, L=Latency) and min-max (MM)), heuristic (H) and baseline (FF), with 7-10 nodes and 1 to 2 users

N:U	**Min E**	**Max E**	**Min L**	**Max L**	**MM**	**H**	**FF**
7:1	0.098	0.082	0.055	0.082	0.152	0.002	0.001
8:1	0.077	0.259	0.065	0.782	0.182	0.003	0.001
9:1	0.120	0.192	0.123	0.095	0.110	0.003	0.001
10:1	0.105	0.916	0.086	0.632	0.132	0.004	0.001
7:2	0.175	0.969	0.153	0.893	0.194	0.004	0.002
8:2	0.177	4.404	0.152	27.498	0.471	0.005	0.002
9:2	0.297	2.887	0.228	0.906	0.418	0.006	0.002
10:2	0.297	17.867	0.255	58.932	0.445	0.021	0.003

5.4.2.2 Analysis of user rejection

This section analyses how many requests are rejected, on the heuristic and the baseline algorithm, with a different number of users (1 to 400 users) and topology sizes (15, 30 and 50 nodes).

Figure 5.5 displays the results obtained and Table 5.5 shows the execution times. We only present some execution times in fixed intervals due the large number of scenarios considered in the experimental analysis. Note that the Integer Linear Programming (ILP) solver was not able to solve the problem for larger scenarios.

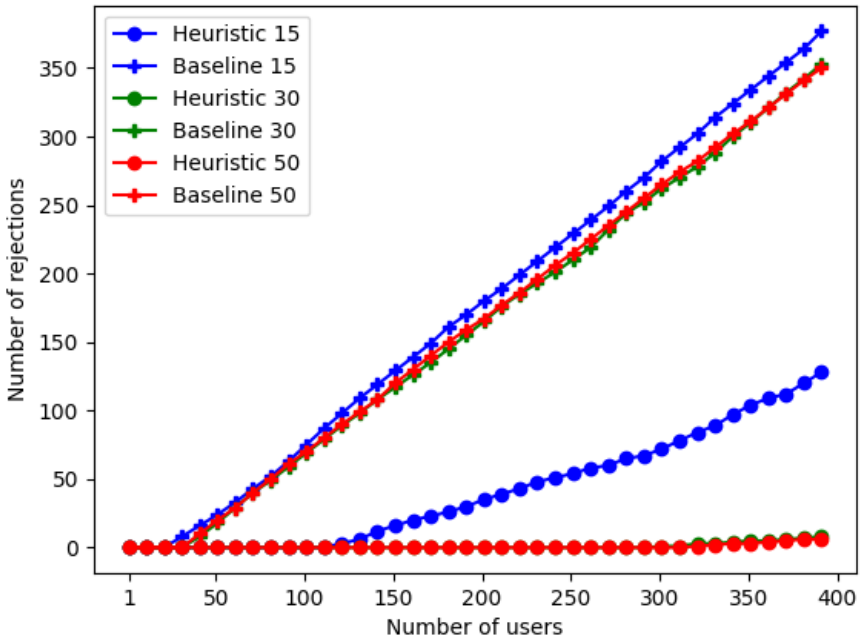


Figure 5.5.: Number of rejected users, 15, 30 and 50 nodes, 1 to 400 users

Table 5.5.: Execution times (in seconds) for the heuristic and baseline, with 15, 30 and 50 nodes and 1, 100, 200, 300 and 400 users

Nodes:Users	**Heuristic**	**First-Fit**
15:1	0.003	0.001
15:100	0.231	0.080
15:200	0.445	0.151
15:300	0.843	0.247
15:400	0.969	0.423
30:1	0.009	0.003
30:100	1.054	0.420
30:200	1.879	0.682
30:300	3.604	1.261
30:400	3.653	1.232
50:1	0.035	0.011
50:100	2.586	0.953
50:200	5.329	1.751
50:300	7.097	2.368
50:400	11.292	3.776

Thus, we approximate upper and lower bounds for each objective, which need normalisation due to the difference in the ranges. The proposed heuristic is able to allocate a larger number of users than the baseline, even in a scenario with a large number of users and a small topology size. It is possible to observe that when the number of users increases, there is an increasing number of rejections for both algorithms. Moreover, in smaller scenarios it is possible to observe that both algorithms start to reject users faster than in larger scenarios (30 and 50

Table 5.6.: Number of hops with different topology sizes and link probabilities

Number of Nodes	Link Probability	Data		Migration	
		H	B	H	B
15	0.25	2	2	2	2
	0.50	2	2	2	2
	0.75	3	3	3	3
30	0.25	2	2	2	2
	0.50	2	2	2	2
	0.75	2	2	2	2
50	0.25	2	2	2	2
	0.50	2	2	2	2
	0.75	2	2	3	2

nodes), due to the lower number of links and nodes. Since the baseline algorithm generates paths without taking into account availability of node resources on the path, that is (are) chosen on each path may not have enough resources. When comparing the scenarios with a small number of users, without rejection, it is possible to observe that the overall energy consumption and latency values obtained by the heuristic are smaller than that of the baseline. This shows that the proposed heuristic is able to migrate and find a new path efficiently and reject less users.

5.4.2.3 Different energy and latency link values

In these experiments, different values of energy and latency for each link in both data communication and migration were defined, i.e., larger values for migration than communication and larger values for communication than migration. We aim to compare the number of hops for different topology sizes and link probabilities. We define link probability as the probability of a link being present on the network while every node remaining connected. Only one user is considered in this section since, in larger scenarios, the baseline rejects more users than the heuristic and it may not be possible to provide a thorough comparison. In this scenario, we considered topology sizes of 15, 30 and 50 nodes, with three different link probabilities: 0.25, 0.5 and 0.75. The number of hops on both approaches is the same (2 and 3 hops), except in the scenario with 50 nodes, link probability of 0.75 and higher values on migration energy and latency, where the solution has 3 hops on the heuristic and 2 hops on the baseline (Table 5.6). This allows us to observe that the proposed heuristic is able to obtain better paths, according to Equation 5.7, with the same number of hops in most scenarios, independent of the number of nodes, link probabilities and energy and latency link values.

5.5 Summary

In this chapter, a Mixed Integer Linear Programming formulation and heuristic for the data communication and migration problem were proposed, to deal with changes of location of a set of users by minimising energy consumption and latency in a fog environment. These approaches, as well as a baseline heuristic for reference purposes, were implemented in a solver and simulator and tested with different topology sizes, number of users and different energy and latency values for migration and data communication. The results show that the proposed heuristic obtains results close to the optimal in most scenarios while avoiding rejection of users and obtaining paths with low number of hops.

Chapter 6.

Virtual Network Embedding

Contents

6.1. Description **79**
6.2. Mixed Integer Linear Programming Formulation . . **80**
6.3. A Heuristic **85**
6.4. Experimental Results **87**
6.4.1. Setup 87
6.4.2. Experimental Analysis 88
6.5. Summary **94**

This chapter introduces a Mixed Integer Linear Programming (MILP) formulation and a heuristic designed to tackle the Virtual Network Embedding (VNE) problem within Network Slicing (NS) environments. The findings will be submitted for journal publication.

The chapter is structured as follows: Section 6.1 provides an overview of the scenario, while Section 6.2 delves into the details of the MILP formulation. Section 6.3 outlines the proposed heuristic, and Section 6.4 discusses the experimental setup and analysis. Finally, Section 6.5 offers a summary of the chapter's key points.

6.1 Description

In a large smart city, the digital infrastructure supports various critical services, including traffic control, public safety, environmental monitoring, and energy distribution. The city's network is built on a multi-layered architecture, comprising core datacenters, edge servers, and distributed fog nodes placed near key locations. As the city grows, the network must not only meet current demands but also allocate resources for future needs driven by upcoming events, seasonal changes, and population increases.

Key areas of concern include managing increased traffic during major events, enhancing public safety measures during festivals, expanding environmental mon-

itoring in new industrial zones, and preparing for the rollout of an electric vehicle charging infrastructure. These future requests require careful planning to ensure the network can scale effectively, maintain low latency for real-time applications, and manage energy consumption efficiently while adapting to the city's evolving demands.

The main contributions of this chapter are the following:

- A MILP formulation and heuristic are proposed to tackle the VNE problem in NS fog-cloud environments by maximising fairness in the consumption of energy and slice/resource interference. A min-max formulation is developed to obtain a fair solution for each objective (energy consumption and slice/resource interference). Afterwards, the min-max formulation is also used to obtain a trade-off solution for both objectives. By applying this formulation, it is possible to avoid the definition of explicit preferences and obtain solutions that take into account both objectives equally.
- In some scenarios, future requests are known, such as requests that arise periodically. To allow a better distribution of resources, we model two types of requests into the formulation, to take into account two different time windows. Thus, it is possible to ensure enough requests for both types of requests and improve fairness.
- A heuristic is proposed to solve the VNE problem in NS environments. The heuristic is based on a greedy approach that prioritises current requests.

The proposed approaches are evaluated in terms of energy consumption and resource usage, an analysis of interference values, and an analysis of current and future requests. The presented results show that the heuristic was able to provide results close to the optimal, while allocating Virtual Networks with low energy consumption and low resource/slice interference.

6.2 Mixed Integer Linear Programming Formulation

In this section, a MILP formulation used to allocate requested VNs on various slices is detailed (a non-linear formulation (Mixed Integer Programming (MIP)) is presented for the sake of comprehension. The linearised version can be obtained using standard techniques (Williams [2013]); see Appendix A.3). Since Service Function Chaining (SFC) requests are a specific case of VNs, the proposed approaches focus on the more generalised VNs. However, both types of requests can be applied. Two different states of the network are considered: current and future. Thus, this model contains two temporal windows. The requests that arise in the first window are referred as *current requests*, which correspond to requests that emerge in a specific moment. In the second time window, the emerging requests are referred as *future requests*. These correspond to requests that are known to appear in the near future. When a current request needs resources during several time windows, it is referred as *current and future request*, or *cf request* for short. We separate these requests since they need to be taken into account in both states of the network. The focus is to obtain a

Table 6.1.: Parameters and variables for the Mixed Integer Linear Programming formulation

Parameter	**Description**
n	Set of nodes on the network
U	Set of slices
α	Parameter to indicate most relevant type of interference
V	Set of Virtual Networks (VNs)
m_v	Links in a VN v
e_i	Energy consumed by a node i
e_{ij}	Energy consumed by a link (i, j)
PD_{ij}	Propagation delay on link (i, j)
MS^v	Message size for VN v
b^{vrs}	Bandwidth required by a link (r, s) in VN v
C^{vr}	CPU requested by node r of VN v
M^{vr}	Memory requested by node r of VN v
γ_{iu}	Maximum CPU capacity of a node i in slice u
μ_{iu}	Maximum memory capacity of a node i in slice u
k_{iu}	Binary parameter that is 1 if a node i is in slice u
w_{iju}	Maximum bandwidth capacity of a link (i, j) in slice u
l^v	Maximum allowed end-to-end latency of a VN v
Variable	**Description**
y_{iu}^{vr}	Binary variable that is 1 if a node r from VN v has been allocated on a substrate node i and slice u
x_{iju}^{vrs}	Binary variable that is 1 if a link (r, s) from VN v to a is allocated on a substrate link (i, j) and slice u

trade-off solution, or compromise, between these types of requests. Moreover, we take into account two different objectives: fairness of energy consumption and fairness of slice/resource interference. Table 6.1 shows the parameters and variables used in the formulation.

Given U slices, V VNs, n nodes on the substrate network and m_v nodes on each VN, the aim is to obtain energy consumption fairness on the substrate network as the first objective. For the sake of comprehension, a node in a VN v with an index from 1 to m_v is denoted. It is formulated as follows:

$$\min \max_{\substack{i,j=1,\dots,n \\ i \neq j}} CE_{ij}(x, x', y, y') = \sum_{u=1}^{U} \left(\sum_{v=1}^{V} \sum_{r=1}^{m_v} \left(y_{iu}^{vr} e_i + \sum_{s=1}^{m_v} x_{iju}^{vrs} e_{ij} \right) + \sum_{v=1}^{V'} \sum_{r=1}^{m'_v} \left(y_{iu}^{vr\prime} e_i + \sum_{s=1}^{m'_v} x_{iju}^{vrs\prime} e_{ij} \right) \right) \tag{6.1}$$

where x_{iju}^{vrs} is a binary variable that is 1 if $r, s \in \{1, \dots, m_v\}$ from VN v is allocated to a substrate link (i, j) and both are on the same slice, 0 otherwise,

y_{iu}^{vr} is a binary variable that is 1 if node r of VN v has been allocated on a substrate node i and both are on the same slice, 0 otherwise, and e_i and e_{ij} are the energy consumed by node i and by link (i, j), respectively. We use "′" to represent cf requests.

When considering the requested VNs for a future instant, the following equation is considered:

$$\min \max_{\substack{i,j=1,\dots,n \\ i \neq j}} FE_{ij}(\bar{x}, x', \bar{y}, y') = \sum_{u=1}^{U} \left(\sum_{v=1}^{\bar{V}} \sum_{r=1}^{\bar{m}_v} \left(\bar{y}_{iu}^{vr} e_i + \sum_{s=1}^{\bar{m}_v} \bar{x}_{iju}^{vrs} e_{ij} \right) + \sum_{v=1}^{V'} \sum_{r=1}^{m'_v} \left(y_{iu}^{vr\prime} e_i + \sum_{s=1}^{m'_v} x_{iju}^{vrs\prime} e_{ij} \right) \right) \tag{6.2}$$

where "¯" denotes a parameter or variable with respect to a future instant.

Thus, the goal is to obtain a compromise solution between both states of the network in terms of energy consumption. It is formulated as follows:

$$\min E(x, \bar{x}, x', y, \bar{y}, y') = \max_{\substack{i,j=1,\dots,n \\ i \neq j}} (CE_{ij}(x, x', y, y'), FE_{ij}(\bar{x}, x', \bar{y}, y')) \tag{6.3}$$

To calculate the energy spent on executing services on each node, the following equation from the energy model described on Kaup et al. [2014] is considered:

$$e_i = 1.5778\text{W} + 0.181 \sum_{u=1}^{U} \sum_{v=1}^{V} \sum_{r=1}^{m_v} y_{iu}^{vr} \tag{6.4}$$

Moreover, to calculate the energy consumed on this link, the following equation is considered, as described in Kaup et al. [2014]:

$$e_{ij} = 0.294\text{W} + 0.006 + 1.06e^{-3} b^{vrs} \frac{\text{W}}{\text{Mbps}} \tag{6.5}$$

$$e_{ji} = 0.294\text{W} + 2.327e^{-3} b^{vrs} \frac{\text{W}}{\text{Mbps}} \tag{6.6}$$

where e_{ij} is the energy consumed by downlink, e_{ji} is the energy consumed by uplink, and b^{vrs} corresponds to the bandwidth of a virtual link (r, s) in VN v.

With the second objective we aim to minimise the slice/resource interference at the substrate network for all the nodes. Here, the aim is to avoid allocating a high number of virtual nodes in a single substrate node, since this causes degradation of throughput (Zhang et al. [2019]) and to avoid allocating a high number of virtual nodes in resources from different slices. Despite the aim of

NS to provide isolation of slices, in some scenarios it may exist slice overloading. Moreover, it may not be possible to reconfigure slices, or it may be too costly. Thus, sharing physical resources may be a feasible option. Therefore, we consider a tunable parameter to control resource interference and slice interference. This is formulated as follows:

$$\min \max_{i=1,\dots,n} CI_i(y, y') = \sum_{u=1}^{U} \left(\sum_{v=1}^{V} \sum_{r=1}^{m_v} y_{iu}^{vr} \left(\alpha(1 - k_{iu}) + (1 - \alpha) \left(\frac{C^{vr}}{\gamma_{iu}} + \frac{M^{vr}}{\mu_{iu}} \right) \right) + \sum_{v=1}^{V'} \sum_{r=1}^{m'_v} y_{iu}^{vr'} \left(\alpha(1 - k_{iu}) + (1 - \alpha) \left(\frac{C^{vr'}}{\gamma_{iu}} + \frac{M^{vr'}}{\mu_{iu}} \right) \right) \right) \tag{6.7}$$

where C^{vr} and M^{vr} indicate the CPU and memory that need to be allocated of virtual node r from VN v, respectively, and γ_{iu} and μ_{iu} indicate the maximum CPU frequency and memory of substrate node i, respectively. k_{iu} is parameter which is 1 if node i is used in slice u and 0 otherwise, and α is a parameter to indicate the most relevant type of interference in the range $[0, 1]$, where 0 focuses completely on resource interference and 1 focuses completely on slice interference.

Similarly to the previous objective, we also formulate the minimisation of the future state of the network with regard to interference:

$$\min \max_{i=1,\dots,n} FI_i(\bar{y}, y') = \sum_{u=1}^{U} \left(\sum_{v=1}^{\bar{V}} \sum_{r=1}^{\bar{m}_v} \bar{y}_{iu}^{vr} \left(\alpha(1 - k_{iu}) + (1 - \alpha) \left(\frac{\bar{C}^{vr}}{\gamma_{iu}} + \frac{\bar{M}^{vr}}{\mu_{iu}} \right) \right) + \sum_{v=1}^{V'} \sum_{r=1}^{m'_v} y_{iu}^{vr'} \left(\alpha(1 - k_{iu}) + (1 - \alpha) \left(\frac{C^{vr'}}{\gamma_{iu}} + \frac{M^{vr'}}{\mu_{iu}} \right) \right) \right) \tag{6.8}$$

Then, the aim is to obtain a compromise solution in both states of the network, focusing on interference:

$$\min I(y, \bar{y}, y') = \max_{i=1,\dots,n} (CI_i(y, y'), FI_i(\bar{y}, y')) \tag{6.9}$$

Note that the two objectives can be conflicting. The first objective intends to maximise the distribution of consumed energy per node, while the second objective aims to minimise the interference of VNs per node. To solve this, it is obtained a compromise solution for both objectives in both states of the network[1].

$$\min f(x, \bar{x}, x', y, \bar{y}, y') =$$

[1]Note that both objective function values need to be on the same range. If the range is not the same, they are equalised (White [1982])

$$\max(E(x, \bar{x}, x', y, \bar{y}, y'), I(y, \bar{y}, y')) \tag{6.10}$$

We consider the following constraints to describe the resources used by the VNs. First, it is necessary to ensure that the CPU frequency (C) and memory (M) are sufficient to allocate the virtual nodes on the substrate nodes. Moreover, each substrate link must have enough available bandwidth to allocate the virtual link. We formalise the CPU, memory, and bandwidth constraints as follows:

$$\sum_{u=1}^{U}\sum_{v=1}^{V}\sum_{r=1}^{m_v}(y_{iu}^{vr}C^{vr} + y_{iu}^{vr\prime}C^{vr\prime}) \leq \gamma_{iu}, \qquad \forall i = 1, \ldots, n \tag{6.11}$$

$$\sum_{u=1}^{U}\sum_{v=1}^{\bar{V}}\sum_{r=1}^{\bar{m}_v}(\bar{y}_{iu}^{vr}\bar{C}^{vr} + y_{iu}^{vr\prime}C^{vr\prime}) \leq \gamma_{iu}, \qquad \forall i = 1, \ldots, n \tag{6.12}$$

$$\sum_{u=1}^{U}\sum_{v=1}^{V}\sum_{r=1}^{m_v}(y_{iu}^{vr}M^{vr} + y_{iu}^{vr\prime}M^{vr\prime}) \leq \mu_{iu}, \qquad \forall i = 1, \ldots, n \tag{6.13}$$

$$\sum_{u=1}^{U}\sum_{v=1}^{\bar{V}}\sum_{r=1}^{\bar{m}_v}(\bar{y}_{iu}^{vr}\bar{M}^{vr} + y_{iu}^{vr\prime}M^{vr\prime}) \leq \mu_{iu}, \qquad \forall i = 1, \ldots, n \tag{6.14}$$

$$\sum_{u=1}^{U}\sum_{v=1}^{V}\sum_{r,s=1}^{m_v}(x_{iju}^{vrs}b^{vrs} + x_{iju}^{vrs\prime}b^{vrs\prime}) \leq w_{iju}, \qquad \forall i, j = 1, \ldots, n \tag{6.15}$$

$$\sum_{u=1}^{U}\sum_{v=1}^{\bar{V}}\sum_{r,s=1}^{\bar{m}_v}(\bar{x}_{iju}^{vrs}\bar{b}^{vrs} + x_{iju}^{vrs\prime}b^{vrs\prime}) \leq w_{iju}, \qquad \forall i, j = 1, \ldots, n \tag{6.16}$$

where CM_{iu} indicates the maximum CPU frequency on substrate node i for slice u, b^{vrs} indicates the requested bandwidth for VN v and link (r, s) and w_{iju} indicates the maximum bandwidth of substrate link (i, j) in slice u.

Moreover, it is necessary to ensure that the latency requirements are met. For example, in a Ultra-Reliable Low-Latency Communications (URLLC) slice, latency is of most importance. This latency is modelled as follows:

$$\sum_{u=1}^{U}\sum_{i,j=1}^{n}\sum_{r,s=1}^{m_v}\left(\frac{MS^v}{w_{iju}} + PD_{ij}\right)x_{iju}^{vrs} \leq l^v, \qquad \forall v = 1, \ldots, V \tag{6.17}$$

$$\sum_{u=1}^{U}\sum_{i,j=1}^{n}\sum_{r,s=1}^{\bar{m}_v}\left(\frac{\overline{MS}^v}{w_{iju}} + PD_{ij}\right)\bar{x}_{iju}^{vrs} \leq \bar{l}^v, \qquad \forall v = 1, \ldots, \bar{V} \tag{6.18}$$

$$\sum_{u=1}^{U}\sum_{i,j=1}^{n}\sum_{r,s=1}^{m_{v'}}\left(\frac{MS^{v\prime}}{w_{iju}} + PD_{ij}\right)x_{iju}^{vrs\prime} \leq l^{v\prime}, \qquad \forall v = 1, \ldots, V' \tag{6.19}$$

where MS^v corresponds to the message size of VN v, PD_{ij} corresponds to the propagation delay, and l^v corresponds to the maximum allowed latency for the end-to-end communication in v.

In the following, we describe the flow conservation between the different source/target nodes. Each node on the VN may be considered as a source, target, or transit node of different paths. The constraint is formulated as fol-

lows:

$$\sum_{j=1}^{n}\sum_{u=1}^{U}(x_{iju}^{vrs} - x_{jiu}^{vrs}) = \sum_{u=1}^{U}(y_{iu}^{vr} - y_{iu}^{vs}), \quad \forall i = 1,\ldots,n, v = 1,\ldots,V, \\ r,s = 1,\ldots,m_v, r < s \tag{6.20}$$

$$\sum_{j=1}^{n}\sum_{u=1}^{U}(\bar{x}_{iju}^{vrs} - \bar{x}_{jiu}^{vrs}) = \sum_{u=1}^{U}(\bar{y}_{iu}^{vr} - \bar{y}_{iu}^{vs}), \quad \forall i = 1,\ldots,n, v = 1,\ldots,\bar{V}, \\ r,s = 1,\ldots,\bar{m}_v, r < s \tag{6.21}$$

$$\sum_{j=1}^{n}\sum_{u=1}^{U}(x_{iju}^{vrs\prime} - x_{jiu}^{vrs\prime}) = \sum_{u=1}^{U}(y_{iu}^{vr\prime} - y_{iu}^{vs\prime}), \quad \forall i = 1,\ldots,n, v = 1,\ldots,V', \\ r,s = 1,\ldots,m_v{}', r < s \tag{6.22}$$

To ensure that all virtual nodes are allocated and on, at most, one substrate node, we consider the following constraint:

$$\sum_{i=1}^{n}\sum_{u=1}^{U} y_{iu}^{vr} = 1, \quad \forall v = 1,\ldots,V, r = 1,\ldots,m_v \tag{6.23}$$

$$\sum_{i=1}^{n}\sum_{u=1}^{U} \bar{y}_{iu}^{vr} = 1, \quad \forall v = 1,\ldots,\bar{V}, r = 1,\ldots,\bar{m}_v \tag{6.24}$$

$$\sum_{i=1}^{n}\sum_{u=1}^{U} y_{iu}^{vr\prime} = 1, \quad \forall v = 1,\ldots,V', r = 1,\ldots,m_v{}' \tag{6.25}$$

6.3 A Heuristic

In this section, the Current + Future (CF) and VN Slicing heuristics developed from the formulation on the previous section are presented.

Algorithm 6.1 is executed at each time-window and receives a set of current and future requests to be mapped. Thus, for each VN request, a call to the Virtual Network Slicing heuristic is made. Since the embedding is performed in a greedy manner, we prioritise the current requests.

Algorithm 6.2 aims to map VNs (or SFCs) in a specific network slice. As data structures, it requires a list L, where each entry contains the mapping of a VN into the substrate network, a min-priority-queue PQ containing the substrate nodes with value according to the min-max formulation (see Equation 6.10) and a queue Q with candidate nodes. For each VN request, queue Q is filled up with the candidate nodes (line 2). Then, the algorithm finds on PQ, the first virtual node r with smallest objective value according to the min-max formulation and node degree equal or lower than the substrate node i, i.e., selects the node at the top of the priority queue that satisfies the previous requirements. This ensures that all links of r can be mapped (line 3). Next, the first virtual node of the actual VN v is removed from the queue (line 4) to be mapped onto the previous obtained substrate node (line 5). The priority queue is updated with the new

Algorithm 6.1: Current+Future heuristic

Input: Graph G, set VC of current Virtual Networks and set of VF future VNs requests for each service
Output: List L containing all the mappings of Virtual Networks to the substrate network

```
1  Create empty list L
2  for v = 1, ..., VC do
3      L.append(VNS(G, v))
4  for v = 1, ..., VF do
5      L.append(VNS(G, v))
6  return L
```

Algorithm 6.2: Virtual Network Slicing heuristic

Input: Graph G, Virtual Network request v
Output: List L containing all the mappings of Virtual Networks to the slice

```
1  Create priority queue PQ containing all substrate nodes with value 0
2  Create queue Q containing VN m_v nodes
3  s_s ← PQ.top()
4  s_l ← Q.pop()
5  Map s_l into s_s and add to list L
6  PQ.update(s_s)
7  while Q is not empty do
8      s_l ← Q.pop()
9      path ← FindPath(s_s, s_l)
10     s_s ← path.lastNode
11     Map VN path into substrate path and add to list L
12     PQ.update(path.nodes)
13 return L
```

value of node i after having virtual node r mapped onto it (line 6). Then, the algorithm traverses each link of the substrate network from a given node (lines 7-12).

FindPath implements a single-source single-target shortest path algorithm, such as Dijkstra algorithm, that ensures the requested upper bound on bandwidth for each candidate link. It finds the best link according to the min-max formulation (lines 9-10). If the link has enough available bandwidth, the virtual link is mapped into this substrate link (lines 11-12). Similarly, if the new node has enough capacity, the virtual node is mapped into the substrate node i or j, where j is the current target. Otherwise, the algorithm compares the min-max value of a different link (i to different node l), adds an extra node on the actual path (i to j to new node k) and chooses the one with lower value.

If there is a loop path on the VN, the algorithm will check the next mapped node, in order to ensure that the loop is also embedded on the substrate net-

Table 6.2.: Parameter values for fog nodes

	Network parameters	Values
Link	Bandwidth	5Gbps
Node	CPU	1.5Ghz
	RAM	4Gb
	Storage	32Gb

Table 6.3.: Service Function Chaining requirements

Requests	CPU	RAM	Bandwidth	Latency
8k Video (eMBB)	150 Mhz	256 Mb	50 Mbps	100ms
VR Space (URLLC)	100 Mhz	128 Mb	25 Mbps	40ms

work. If it has been mapped, then the shortest path between those two nodes is found. The algorithm iterates until the VN has been mapped or there not exist enough resources to do so. The time complexity of both algorithms is $\mathcal{O}((|VC| + |VF|)|N|(t_F + t_{|PQ|}))$, where t_F corresponds to the complexity of finding a path for each pair of virtual nodes on the substrate network and $t_{|PQ|}$ corresponds to the complexity of updating PQ ($\mathcal{O}(n \log(n))$), where n is the number of nodes.

6.4 Experimental Results

In this section, we describe the network setup, the service and SFC requirements, and the experimental analysis and obtained results. We implemented the approaches in *Yet Another Fog Simulator (YAFS)* (Lera et al. [2019]) and used the *SCIP* solver (Bestuzheva et al. [2021]) with *PySCIPOpt* (Maher et al. [2016]) to implement the MILP formulation, with all the objective functions linearised.

6.4.1 Setup

In terms of network setup, the Raspberry Pi device (Foundation [2021]) for the fog nodes and two different slices (enhanced Mobile Broadband (eMBB), URLLC) are considered. Table 6.2 shows the values used to model the Raspberry Pi devices.

Regarding requests, a linear SFC topology with 2 nodes is considered, where each virtual node has a given service to execute in a specific order, such as compressing data and encoding. It is assumed that each virtual node is mapped only on one substrate node, but each substrate node may have more than one virtual node. However, one virtual link may be mapped into one or more substrate link.

Each slice has already been allocated in the substrate network and the aim is to embed the services modelled as SFC requests into the corresponding slice. Table 6.3 shows the slices considered in this work, their respective requests and their requirements.

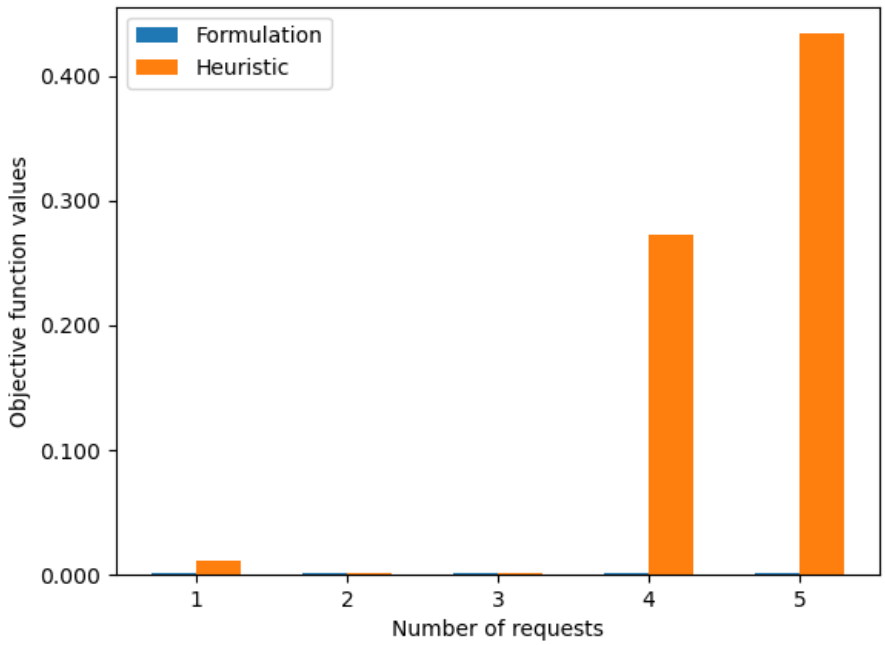


Figure 6.1.: Mixed Integer Linear Programming formulation and heuristic objective function values

6.4.2 Experimental Analysis

In the following, an experimental analysis with the goal of analysing the performance of the MILP formulation and Virtual Network Slicing heuristic (Algorithm 6.1) is accomplished as follows: 1) a solution quality analysis of the heuristic regarding the optimal solution in a small scenario is performed. This allows to analyse how close the heuristic is to the optimal; 2) a study with different α values (see Equations 6.7 and 6.8) to analyse their effect in the embedding of VNs is detailed; 3) a set of current and future services is considered. Due to the prohibitive execution time of the MILP formulation, it was only considered in 1).

6.4.2.1 Formulation and heuristic

To effectively compare the performance of the proposed formulation and heuristic approach, experiments in a controlled, static scenario are conducted. In this scenario, the physical network contains 5 nodes as a complete graph. Each node handles various types of requests, with the number of SFCs from 1 to 5, all within a 2-slice configuration. By varying the number of requests, the goal is to assess how well each method scales and how closely the heuristic approximates the optimal solution in environments of increasing complexity.

Figure 6.1 shows the solution quality obtained from 1 to 5 requests. It is possible to observe that, for few requests, the heuristic is able to provide results close to the optimal, while ensuring a trade-off between energy consumption and

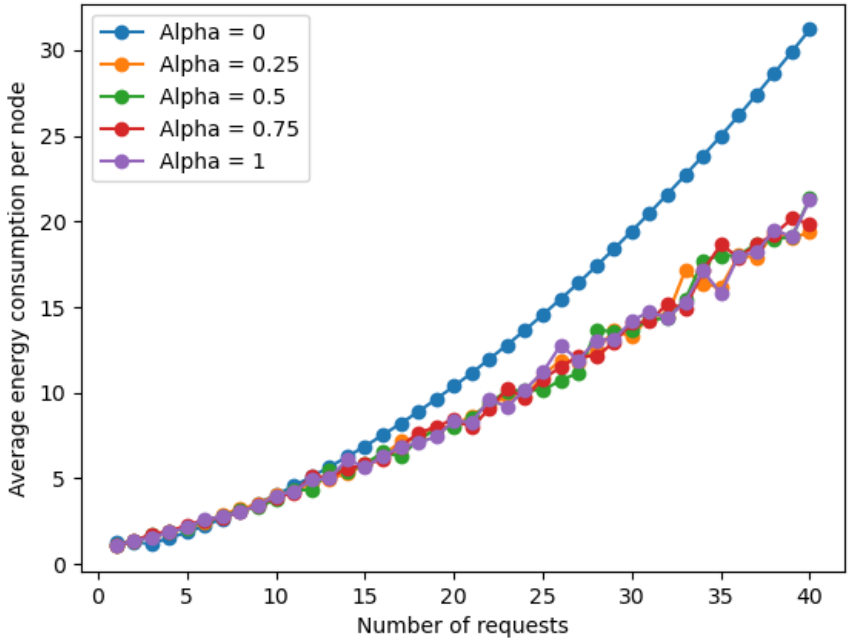


Figure 6.2.: Average energy consumption per node for different α values from 1 to 40 requests

interference on the slices. With 4 and 5 services, there is a larger difference between the heuristic and the formulation. This is due to the fact that the heuristic only obtains suboptimal solutions.

6.4.2.2 Analysis of α values

To explore how different interference values impact resource allocation, the α parameter in the resource and slice interference objective functions, as defined in Equations 6.7 and 6.8, is adjusted to reflect varying levels of emphasis on resource interference (with α values approaching 0) versus slice interference (with α values approaching 1). Five specific α values (0, 0.25, 0.5, 0.75, and 1) are used to evaluate their effects on average energy consumption and resource usage in a network containing 30 devices.

Figures 6.2 and 6.3 show the results of this analysis. In terms of energy consumption, the results indicate that, except for $\alpha = 0$, the average values are relatively consistent across all α values. This suggests that the model is able to maintain energy efficiency regardless of the emphasis placed on resource or slice interference. In terms of resource usage, the results show that the model is able to maintain a consistent level of average resource usage per node across all α values, except for $\alpha = 0$, where the average is lower. This indicates that the model is able to effectively balance resource usage and interference, ensuring that resources are allocated efficiently while minimising interference on the slices.

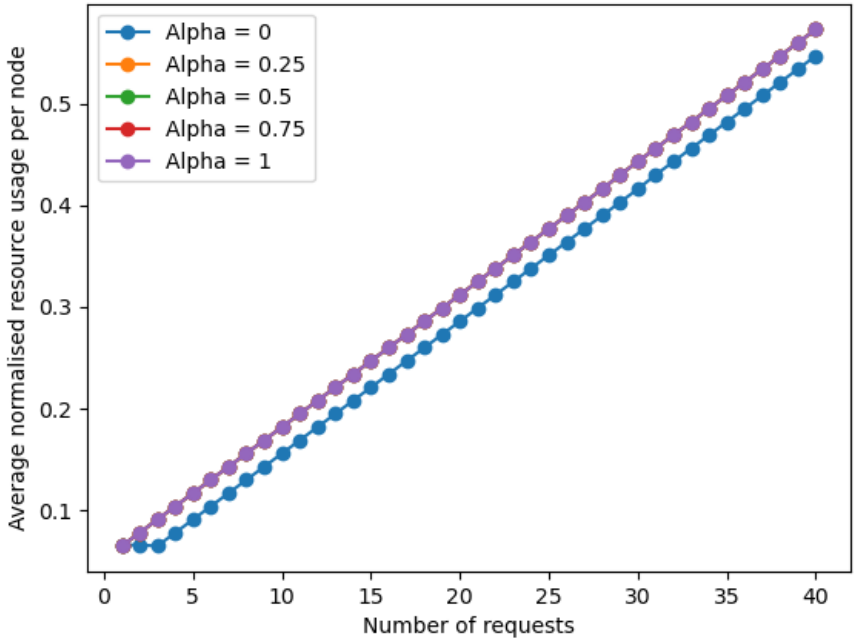


Figure 6.3.: Average normalised resource usage per node for different α values from 1 to 40 requests

Tables 6.4 and 6.5 show the standard deviation of the results for energy consumption and resource usage. The results indicate that the model is able to maintain a consistent level of energy consumption and resource usage across all α values, except for $\alpha = 0$, where the standard deviation is higher in both objectives. This suggests that the model is able to effectively balance energy consumption and resource usage, in general, ensuring that resources are allocated efficiently while minimising interference on the slices.

6.4.2.3 Analysis of current and future services

One of the key contributions of the proposed model is its ability to proactively allocate resources for future known requests, thereby enhancing overall network efficiency. To evaluate this capability, a two-phase analysis is performed. In the first phase, we assess the model's performance by processing only the current requests. After an interval, the model's performance is re-evaluated by treating future requests as if they were current. In the second phase, a scenario where the model has prior knowledge of some future requests during specific steps is introduced. For this purpose, we generate a set of hypothetical future requests based on the average characteristics of two existing requests. This approach allows to gauge the effectiveness of the model in reserving resources in advance, ensuring that future demands are met without compromising the network's current operations. In this scenario, the network contains 30 devices.

Figures 6.4 and 6.5 show the results of this analysis. In terms of energy consumption, the results indicate that the model is able to maintain a consistent level of

Table 6.4.: Standard deviation of the average energy consumption for different values of α for the 40 requests

Requests	$\boldsymbol{\alpha = 0}$	$\boldsymbol{\alpha = 0.25}$	$\boldsymbol{\alpha = 0.5}$	$\boldsymbol{\alpha = 0.75}$	$\boldsymbol{\alpha = 1}$
1	3.640	2.258	2.379	2.459	2.379
2	4.645	2.849	3.110	3.102	3.185
3	5.659	3.461	4.130	4.657	3.193
4	6.997	4.116	4.057	4.399	4.164
5	8.308	4.544	3.970	5.309	4.178
6	9.809	4.755	5.450	5.690	6.217
7	11.303	6.459	5.753	5.572	5.697
8	12.805	7.258	6.345	6.650	6.353
9	14.488	7.499	6.192	7.322	6.784
10	16.294	8.793	7.401	8.715	8.825
11	17.993	8.481	9.318	8.632	8.483
12	19.976	10.352	7.840	12.609	10.982
13	21.833	10.083	13.976	10.793	9.991
14	23.988	10.151	11.348	11.903	15.818
15	26.003	11.386	11.797	11.915	10.656
16	28.165	12.945	14.326	11.745	12.927
17	30.484	15.833	11.093	13.970	14.873
18	32.802	15.491	13.844	17.908	14.485
19	35.151	16.616	17.052	17.398	14.432
20	37.789	16.136	15.500	18.397	18.265
21	40.297	17.546	17.228	13.936	15.750
22	42.944	19.243	19.912	17.549	19.181
23	45.758	18.510	23.179	22.716	16.794
24	48.562	17.693	20.674	17.704	21.251
25	51.403	22.113	18.654	22.541	23.362
26	54.525	25.190	20.197	25.561	29.461
27	57.668	23.439	21.772	24.191	24.695
28	60.952	23.937	30.906	23.699	26.387
29	64.259	29.063	27.445	25.373	26.730
30	67.706	27.797	29.056	29.336	30.570
31	71.177	28.559	28.897	31.469	29.670
32	74.630	27.850	27.933	31.075	26.700
33	78.125	37.137	31.794	28.237	29.460
34	81.741	33.042	38.883	35.536	36.434
35	85.545	29.517	39.739	40.770	28.398
36	89.472	35.910	39.130	36.504	36.199
37	93.293	34.092	37.577	40.630	35.715
38	97.234	41.690	37.450	40.475	40.479
39	101.365	37.786	37.392	40.254	35.802
40	105.618	35.390	48.036	36.882	44.895

average energy consumption per node across both scenarios. This suggests that the model is able to effectively balance energy consumption and resource usage,

Table 6.5.: Standard deviation of the average normalised resource usage for different values of α for the 40 requests

Requests	$\boldsymbol{\alpha = 0}$	$\boldsymbol{\alpha = 0.25}$	$\boldsymbol{\alpha = 0.5}$	$\boldsymbol{\alpha = 0.75}$	$\boldsymbol{\alpha = 1}$
1	0.156	0.104	0.112	0.112	0.112
2	0.189	0.132	0.144	0.138	0.144
3	0.221	0.159	0.179	0.197	0.148
4	0.255	0.181	0.177	0.181	0.172
5	0.287	0.192	0.174	0.217	0.183
6	0.321	0.201	0.225	0.232	0.240
7	0.354	0.253	0.236	0.221	0.225
8	0.387	0.271	0.255	0.245	0.249
9	0.420	0.279	0.243	0.267	0.264
10	0.454	0.325	0.286	0.306	0.320
11	0.486	0.317	0.336	0.307	0.320
12	0.520	0.355	0.295	0.404	0.378
13	0.552	0.359	0.420	0.366	0.361
14	0.586	0.362	0.383	0.399	0.458
15	0.619	0.398	0.399	0.399	0.381
16	0.652	0.425	0.452	0.405	0.425
17	0.685	0.489	0.394	0.460	0.456
18	0.718	0.482	0.472	0.508	0.454
19	0.751	0.507	0.507	0.521	0.465
20	0.784	0.501	0.489	0.534	0.526
21	0.817	0.529	0.518	0.464	0.494
22	0.850	0.559	0.571	0.543	0.585
23	0.883	0.576	0.601	0.613	0.527
24	0.916	0.553	0.586	0.548	0.587
25	0.949	0.629	0.560	0.614	0.644
26	0.983	0.672	0.586	0.644	0.731
27	1.016	0.642	0.594	0.670	0.650
28	1.050	0.672	0.748	0.645	0.711
29	1.083	0.728	0.725	0.683	0.695
30	1.117	0.683	0.711	0.736	0.744
31	1.150	0.744	0.729	0.723	0.758
32	1.183	0.725	0.714	0.769	0.715
33	1.216	0.861	0.764	0.732	0.756
34	1.249	0.801	0.876	0.844	0.843
35	1.282	0.768	0.874	0.907	0.746
36	1.316	0.858	0.856	0.851	0.855
37	1.348	0.829	0.877	0.873	0.848
38	1.381	0.898	0.873	0.888	0.900
39	1.415	0.858	0.862	0.918	0.863
40	1.448	0.858	0.961	0.883	0.955

ensuring that resources are allocated efficiently while minimising interference on the slices. In terms of resource usage, the results show that the model is able to

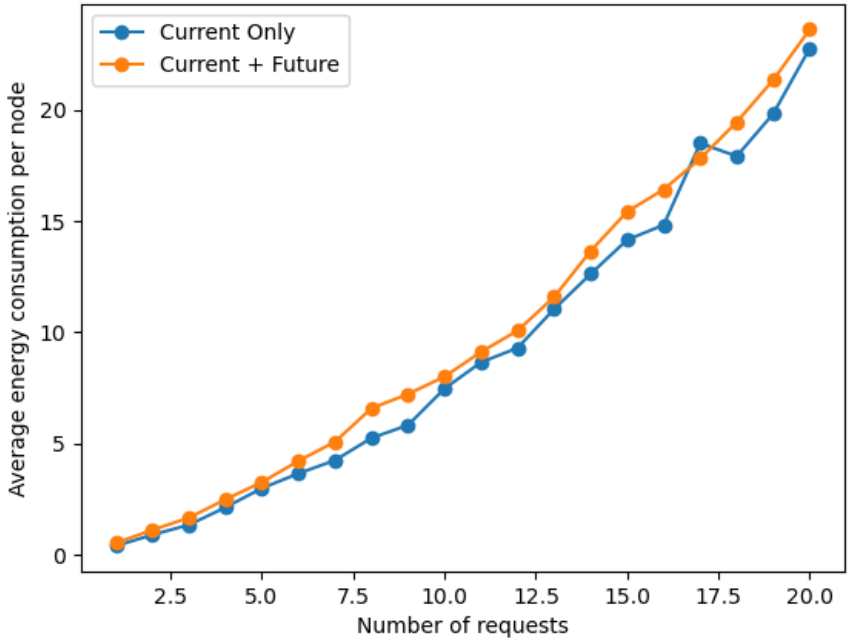


Figure 6.4.: Average energy consumption per node for current and future services from 3 to 60 requests

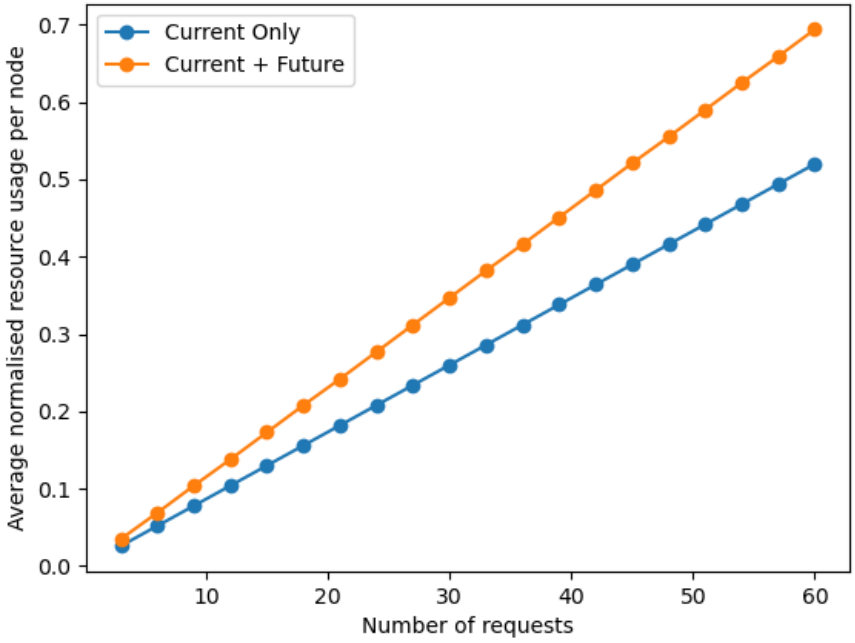


Figure 6.5.: Average normalised resource usage per node for current and future services from 3 to 60 requests

maintain a consistent level of average resource usage per node across all scen-

Table 6.6.: Standard deviation of the average energy consumption for Current Only and Current + Future requests

Requests	**Current Only**	**Current + Future**
3	1.246	2.060
6	2.927	3.652
9	3.407	4.198
12	6.353	7.145
15	8.906	8.959
18	10.274	11.506
21	10.249	13.921
24	13.179	20.119
27	13.679	20.365
30	19.220	20.236
33	23.041	22.613
36	22.594	24.124
39	27.510	27.611
42	32.580	35.708
45	37.176	41.822
48	34.969	42.610
51	51.289	44.755
54	42.436	48.965
57	46.582	54.022
60	60.699	62.118

arios. This indicates that the model is able to effectively balance resource usage and interference, ensuring that resources are allocated efficiently while minimising interference on the slices. It is important to notice that while current only requests are allocated instantly, future requests are not actually allocated, i.e., they are only used as a guideline to help prepare the requested resources when necessary. This means that the algorithm is able to distribute the resources in a more efficient way, ensuring that future requests are met without compromising the network's current operations.

Tables 6.6 and 6.7 show the standard deviation of the results for energy consumption and resource usage. The results indicate that the model is able to maintain a consistent level of energy consumption and resource usage across all scenarios. This suggests that the model is able to effectively balance energy consumption and resource usage, ensuring that resources are allocated efficiently while minimising interference on the slices.

6.5 Summary

In this chapter, a Mixed Integer Linear Programming formulation and heuristic were proposed to solve the Virtual Network Embedding problem in fog environments with Network Slicing architectures. The aim is to maximise fairness of energy consumption and resource/slice interference in two different settings.

Table 6.7.: Standard deviation of the average resource consumption for Current Only and Current + Future requests

Requests	**Current Only**	**Current + Future**
3	0.062	0.100
6	0.128	0.172
9	0.152	0.207
12	0.243	0.317
15	0.318	0.386
18	0.351	0.475
21	0.371	0.536
24	0.435	0.681
27	0.446	0.690
30	0.554	0.716
33	0.609	0.780
36	0.619	0.820
39	0.705	0.916
42	0.771	1.052
45	0.828	1.148
48	0.833	1.174
51	0.992	1.227
54	0.934	1.299
57	0.997	1.382
60	1.100	1.485

First, we assume a set of current requests that need resources and, secondly, there exist a set of future known requests that will need resources in a given time window. A range of scenarios has been considered. The presented results show that the heuristic is able to provide results close to the optimal, while ensuring a trade-off between energy consumption and interference on the slices.

Chapter 7.

Hybrid Approach

Contents

7.1. Description **98**
7.2. Mixed Integer Linear Programming Formulations . **100**
7.2.1. Virtual Network Embedding 103
7.2.2. Service Placement 104
7.3. Heuristics **107**
7.3.1. Virtual Network Embedding (Offline) 108
7.3.2. Virtual Network Embedding (Online) 109
7.3.3. Virtual Network Reconfiguration 110
7.3.4. Service Placement (Offline) 110
7.3.5. Service Placement (Online) 112
7.4. Experimental Results **112**
7.4.1. Setup 113
7.4.2. Experimental Analysis 115
7.5. Summary **126**

This chapter presents several key contributions to the field, including Mixed Integer Linear Programming (MILP) formulations and heuristics for addressing both the Virtual Network Embedding (VNE) and Service Placement (SP) problems. In addition, a dynamic heuristic is developed to tackle the Virtual Network (VN) reconfiguration problem, offering a more adaptable solution for evolving network environments. These research efforts culminated in the publication of the results in the Future Generation Computer Systems journal in 2022 (Godinho et al. [2022]).

The chapter is organised as follows: Section 7.1 sets the stage by describing the scenario in which these problems arise, providing the necessary context for understanding their significance. Section 7.2 delves into the MILP formulations, offering a detailed explanation of the mathematical models used to optimise network resource allocation. In Section 7.3, the proposed heuristics are introduced,

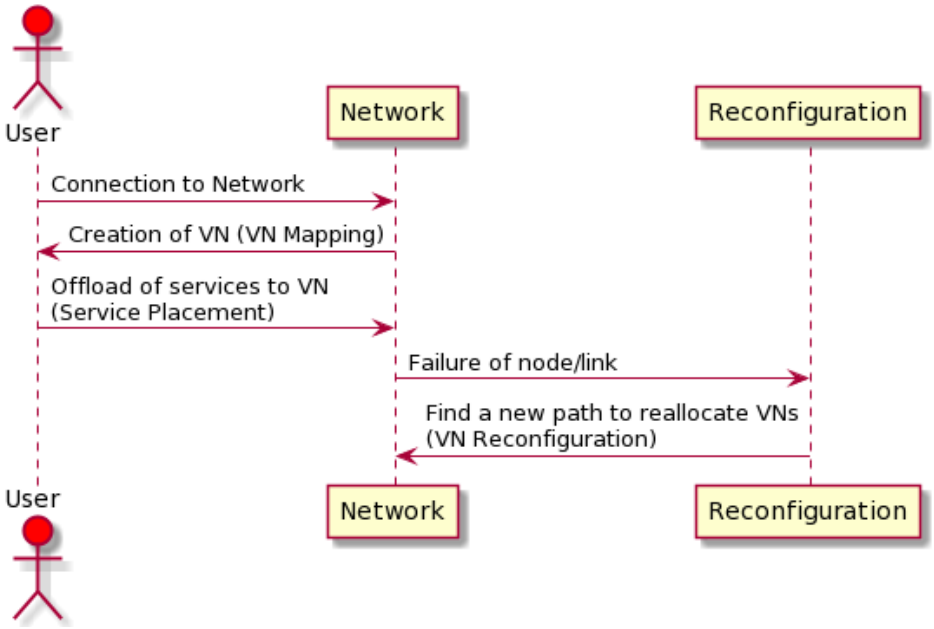


Figure 7.1.: Conceptualisation of the framework

highlighting their role in providing efficient, near-optimal solutions to the VNE, SP, and VN reconfiguration problems. Section 7.4 outlines the experimental analysis and setup, detailing the methodologies used to evaluate the performance of the proposed approaches. Finally, Section 7.5 offers a comprehensive summary of the chapter's key contributions and findings, setting the stage for future research in this domain.

7.1 Description

In a smart event environment, with a conglomeration of users, there is a need for computational and communication resources (Afolabi et al. [2017]). One example of this environment is the Olympics Games, which is composed of different areas, such as marathon, cycling, martial arts, and others. It is assumed that users in an area have a set of applications with their own requirements (DOCOMO 5G White Paper [2014]; Frascolla et al. [2017]). To avoid degradation of Quality of Service (QoS), it is necessary to adequately distribute resources, since each user needs to have its own slice of the network and resources, while ensuring latency requirements to meet deadlines of the users. These applications may be video sharing, health sensors or reception of information concerning results of other events. Each venue may also have its own set of applications and requirements, such as video monitoring and disaster detection.

Figure 7.1 shows a diagram of the proposed framework. It works as follows: first, there is a user that needs a connection to a network with a set of services to offload. The request is in the form of a Virtual Network (VN), which provides the necessary resources for the user. In order to take a decision, the solution method obtains a (sub)optimal solution that indicates where this VN will be allocated in the physical network. If it is not possible to allocate, then the user

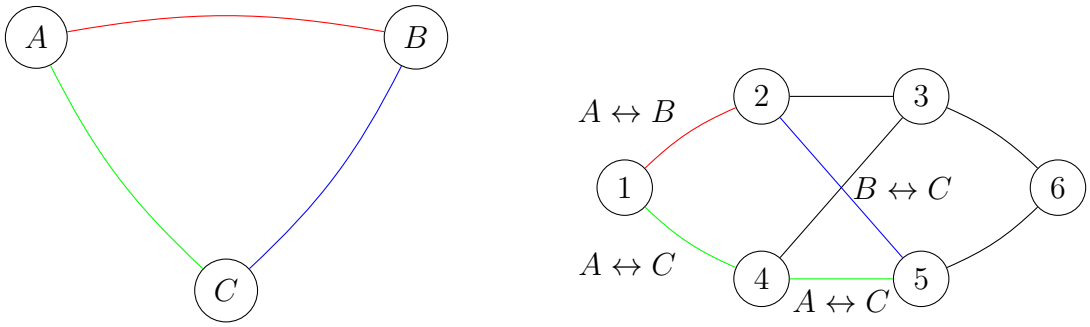
is rejected. Next, the solution method for the SP on the users' VN is used to efficiently decide where the services are offloaded in the VN. If it is not possible to offload, then the services are sent to the cloud. In the case of a failure on a node/link on the physical network, the reconfiguration heuristic is used, where the users' VN will be migrated to a new place to be usable again as fast as possible. Otherwise, it is not necessary to perform any action.

Here, the topology considered in this work is described. The topology is split into a set of clusters. Each cluster has its own substrate network to which users in that area connect. Every time a user needs an Internet connection, it requests a connection to the closest node and the system provides a unique VN for communication and computing needs. Each node on this VN will need a set of resources, such as CPU, memory and storage, which may be represented as a Virtual Machine (VM) or container on a given node and each link will need to ensure the requested bandwidth. Then, each user has access to a set of services that may only perform communication or execute some processing along the path, for example, video encoding before reaching the target device.

Figure 7.2a illustrates a possible VN request from a user, with a set of virtual nodes and links, while Figure 7.2b represents a single cluster substrate network and the corresponding mapping of the VN onto the network. For each link on the VN it is necessary to decide which link(s) on the substrate network are chosen. For this, we need to take into account the resources of the nodes and links, since we assume that not only the link is mapped, but the associated nodes as well. Moreover, we need to take into account connectivity, i.e., to ensure that the existing paths on the VN are mapped correctly on the substrate network. In the example of Figure 7.2a, $A-B$ is mapped into $1-2$ (red), $A-C$ is mapped into $1-4-5$ (green) and $B-C$ is mapped into $2-5$ (blue). The latency of $A-C$ is the sum of both links latency on the substrate network and both links have to ensure that the requirements of the corresponding link on the VN are met. On this work, it is assumed that each virtual link may be mapped into more than one physical link. In some scenarios, it may be necessary to map one virtual link in more than one physical link ($A-C$ on the VN and $1-4-5$ on the substrate network), in order to ensure that the VN request has the same topology on the substrate network or due to lack of resources. We also assume that each node is mapped only on one node on the substrate network, i.e., all the necessary resources for a user must be on a single node.

The following contributions are highlighted:

- To integrate VNE, VN reconfiguration, and SP into a single framework that allocates the necessary resources to a user and deals with failures when they occur.
- To reconfigure the network when failures occur.
- To avoid degradation of QoS by allocating resources efficiently in terms of fairness.
- To provide min-max MILP formulations for the VNE and SP problems.



(a) Virtual Network request (b) Substrate Network

Figure 7.2.: Virtual Network request on a Substrate Network

- To develop heuristics that provide results close to the optimal solutions.

For the experimental analysis, offline, online and dynamic settings are considered. The offline setting is composed of randomly generated values and the online setting is composed of randomly generated values with an average of 10 runs and a realistic scenario with referenced parameters. The results obtained with the proposed framework show that the heuristics are able to obtain results close to the optimal value in most offline settings. In the online versions, the VNE approach attains better distribution of resources when comparing to an approach without the usage of fairness. Moreover, the dynamic reconfiguration approach is able to reject less users than an approach based on allocating a node/link disjoint VN for each user. Finally, the SP approach is able to lower the energy consumption and latency when comparing to a state-of-the-art approach.

7.2 Mixed Integer Linear Programming Formulations

In this section, Mixed Integer Linear Programming formulations for the Virtual Network Embedding and Service Placement problems for static scenarios are detailed (non-linear formulations (Mixed Integer Programming (MIP)) are presented for the sake of comprehension. The linearised versions can be obtained using standard techniques (Williams [2013]); see Appendix A.4). Figure 7.3 shows preliminary experimental results obtained with a MILP solver on formulation and a heuristic (Section 7.3). The results show that the MILP formulation scales badly with a small substrate network (5 nodes) and small sized VNs (1 to 4 nodes), which makes it unsuitable for realistic scenarios.

It is considered that each user has a VN with its own set of requirements, such as bandwidth, CPU, RAM, and storage to be mapped on the substrate network. In order to avoid unbalanced energy consumption and usage on some nodes/links, we consider fairness, i.e., distribute the resources on the physical network according to some objectives of interest.

The goal is to optimise two objectives in the VNE: to maximise fairness of energy and to maximise fairness of bandwidth. The VNE problem can be formulated as a bi-objective multicommodity flow problem with dynamic transit nodes, i.e.,

Table 7.1.: Parameters and variables for the Mixed Integer Linear Programming formulations

Parameter	**Description**
V, n, K	Set of Virtual Networks, nodes on the network, and commodities
q_k	Number of services on each commodity
m_v	Links in a VN v
e^{vr}	Energy consumed by a VN v in node r
$\bar{e}^{vrs}$	Energy consumed by a VN v in link (r, s)
b^{vrs}	Bandwidth requested by a VN v in to a link (r, s)
U_b	Upper bound of the second objective function
C^{vr}	CPU needed to be allocated to virtual node r
M^{vr}	Memory needed to be allocated to virtual node r
S^{vr}	Storage needed to be allocated to virtual node r
CM_i	Maximum available CPU of substrate node i
MM_i	Maximum available memory of substrate node i
SM_i	Maximum available storage of substrate node i
$\bar{b}_{ij}$	Maximum bandwidth of substrate link (i, j)
PD_{ij}	Propagation delay on link (i, j)
MS_k	Message size for commodity k
$\bar{e}_{ijk}$	Energy consumed by commodity k in link (i, j)
e_{ikp}	Energy consumed by services of a commodity k in node i
d_{ijk}	Delay to traverse a link (i, j) of a commodity k
t_{ikp}	Time to execute service p from commodity k in node i
C_{kp}	CPU needed to execute a service of a commodity
M_{kp}	Memory needed to execute a service of a commodity
S_{kp}	Storage needed to execute a service of a commodity
T	Sum of all the execution times t_{ih} (Upper bound)
D_k	Deadline of commodity k
H_k	Set of indices containing commodity k
b_k	Bandwidth needed for the commodity k
$\bar{b}_{ij}$	Available bandwidth on arc (i, j)
U_{ik}	Conservation matrix with value 1 on the source node rows, -1 on the target node rows, and 0 on the others
Variable	**Description**
y_i^{vr}	Binary variable that is 1 if a VN v corresponding to a node r has been allocated on a substrate node i
x_{ij}^{vrs}	Binary variable that is 1 if a VN v corresponding to a link (r, s) is allocated on a substrate link (i, j)
z_{ijk}	Binary variable that is 1 if the commodity k traverses link (i, j)
w_{ikp}	Binary variable that is 1 if service p of a commodity k is computed on node i
s_h	Integer variable that indicates the start of service h
δ_{hg}	Slack binary variable that is 1 if the start time of service h is before service g

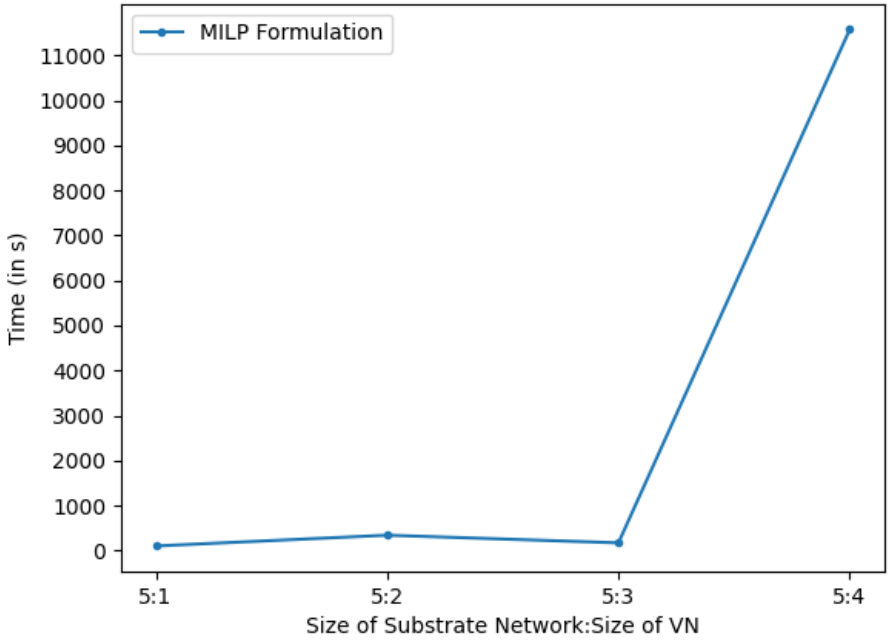


Figure 7.3.: Execution time (in s) of the Mixed Integer Linear Programming formulation

each node may be a source, target or transit node.

In the SP problem, the focus is on two objectives: to minimise both energy consumption and latency. The VN has already been mapped for each user and, therefore, each VN has its own set of K commodities (applications) to apply on the network. Thus, the problem can be recast as a bi-objective multicommodity flow problem with capacities at the nodes, where each weight corresponds to the available resources (CPU, RAM and storage). This corresponds to services to be computed on nodes along the path. Moreover, each node may process one or more services, or none.

In order to be used on the same framework, both problems are joined as follows: first, the user connects to the network with a set of VN requirements and the VNE problem embeds the users' VN. Then, each user offloads the services on its own VN already mapped on the substrate network. Any change on the substrate network will impact the performance on the user.

A commodity (or application) $k \in \{1, \ldots, K\}$ has a source and target node, a set of services to be computed[1], demands for each service, and a deadline to meet. Each VN is represented by an undirected graph $G = (N, A)$, where N indicates the set of nodes and A the set of links. The nodes represent the candidates to offload each service, whereas the links represent the links to be mapped on the substrate network. Each link $(i, j) \in A$ has an available bandwidth and each

[1]Note that we also assume that an application may only use communication, which in this case has zero services to compute.

node $i \in N$ has a set of available resources (CPU, memory and storage).

Table 7.1 shows the parameters and variables used in the formulations.

7.2.1 Virtual Network Embedding

Given V VNs, n nodes on the substrate network and m_v nodes on each VN, the goal is to obtain energy consumption fairness on the substrate network as the first objective. For the sake of comprehension, it is denoted a node in a VN v with an index from 1 to m_v. It is formulated as follows:

$$\min \max_{\substack{i,j=1,\dots,n \\ i \neq j}} FE_{ij}(x,y) = \sum_{v=1}^{V} \sum_{r=1}^{m_v} \left(y_i^{vr} e^{vr} + \sum_{s=1}^{m_v} x_{ij}^{vrs} \bar{e}^{vrs} \right) \tag{7.1}$$

where x_{ij}^{vrs} is a binary variable that is 1 if $r, s \in \{1, \dots, m_v\}$ from VN v is allocated to a substrate link (i, j), 0 otherwise, y_i^{vr} is a binary variable that is 1 if node r of VN v has been allocated on a substrate node i, 0 otherwise, and e^{vr} and $\bar{e}^{vrs}$ are the energy consumed by a node r in VN v and by a link (r, s) in v, respectively.

The aim in the second objective is to obtain fairness on the bandwidth usage at the substrate network for all the links, i.e., to distribute equally the usage of bandwidth throughout the network. It is formulated as follows:

$$\max \min_{\substack{i,j=1,\dots,n \\ i \neq j}} T_{ij}(x) = \sum_{v=1}^{V} \sum_{r=1}^{m_v} \sum_{s=1}^{m_v} x_{ij}^{vrs} b^{vrs} \tag{7.2}$$

where b^{vrs} is the bandwidth requested by link (r, s) from VN v.

Note that the two objectives can be conflicting. This implies that a commodity may go through a path that spends less energy, yet can only send smaller messages or viceversa. The following equation is used to obtain a compromise solution:

$$\min f(x,y) = \max_{\substack{i,j=1,\dots,n \\ i \neq j}} (FE_{ij}(x,y), U_b - T_{ij}(x)) \tag{7.3}$$

where U_b is an upper bound on the second objective function (see Equation 7.2)[2].

The following constraints are used to describe the resources used by the VNs. First, it is necessary to ensure that the CPU instructions per second (C), memory (M) and storage (S) are sufficient to allocate the virtual nodes on the substrate nodes. Moreover, each substrate link must have enough available bandwidth to allocate the virtual link. We formalise CPU, memory, storage and bandwidth

[2]Note that both objective function values need to be on the same range. If the range is not the same, they are equalised (White [1982])

constraints as follows:

$$\sum_{v=1}^{V}\sum_{r=1}^{m_v} y_i^{vr} C^{vr} \leq CM_i, \qquad \forall i = 1,\ldots,n \tag{7.4}$$

$$\sum_{v=1}^{V}\sum_{r=1}^{m_v} y_i^{vr} M^{vr} \leq MM_i, \qquad \forall i = 1,\ldots,n \tag{7.5}$$

$$\sum_{v=1}^{V}\sum_{r=1}^{m_v} y_i^{vr} S^{vr} \leq SM_i, \qquad \forall i = 1,\ldots,n \tag{7.6}$$

$$\sum_{v=1}^{V}\sum_{r=1}^{m_v}\sum_{s=1}^{m_v} x_{ij}^{vrs} b^{vrs} \leq \bar{b}_{ij}, \qquad \forall i,j = 1,\ldots,n \tag{7.7}$$

where C^{vr}, M^{vr}, and S^{vr} indicate the CPU, memory, and storage that need to be allocated to virtual node r from VN v, CM_i, MM_i, SM_i indicate the maximum available CPU, memory, and storage of substrate node i, respectively, and $\bar{b}_{ij}$ indicates the maximum bandwidth of substrate link (i,j).

In the following, the flow conservation between the different source/target nodes is described. Each node on the VN may be considered as a source, target, or transit node of different paths. It is formulated as follows:

$$\sum_{j=1}^{n} x_{ij}^{vrs} - \sum_{j=1}^{n} x_{ji}^{vrs} = y_i^{vr} - y_i^{vs}, \quad \forall i = 1,\ldots,n, v = 1,\ldots,V, \\ r,s = 1,\ldots,m_v, r < s \tag{7.8}$$

Finally, to ensure that all virtual nodes are allocated and on, at most, one physical node, we consider the following constraints:

$$\sum_{i=1}^{n} y_i^{vr} = 1, \qquad \forall v = 1,\ldots,V, r = 1,\ldots,m_v \tag{7.9}$$

$$\sum_{r=1}^{m_v} y_i^{vr} \leq 1, \qquad \forall i = 1,\ldots,n, v = 1,\ldots,V \tag{7.10}$$

7.2.2 Service Placement

Given n nodes, where the n-th node corresponds to the cloud, and K commodities, each of which representing a flow from a source node to a target node, we aim to minimise energy consumption. Thus, we have the first objective as follows:

$$\min E(w,z) = \sum_{k=1}^{K}\sum_{i=1}^{n}\left(\sum_{p=1}^{q_k} w_{ikp} e_{ikp} + \sum_{j=1}^{n} z_{ijk}\bar{e}_{ijk}\right) \tag{7.11}$$

where in each commodity k, q_k indicates the number of services to be computed, z_{ijk} is a binary variable that is 1 if the commodity k traverses link (i,j), 0 otherwise, w_{ikp} is a binary variable that is 1 if service p of a commodity k is computed on node i, 0 otherwise, $\bar{e}_{ijk}$ is the energy consumed by commodity k

in arc (i, j), e_{ikp} is the energy consumed to compute services $p = 1, \ldots, q_k$ of a commodity k in node i.

In the second objective, the goal is to minimise the overall latency, that is, to minimise both communication time from source to target on each commodity, and the execution time of the corresponding service. We formalise this latency objective as follows:

$$\min L(w, z) = \sum_{k=1}^{K} \sum_{i=1}^{n} \left(\sum_{p=1}^{q_k} w_{ikp} t_{ikp} + \sum_{j=1}^{n} z_{ijk} d_{ijk} \right) \tag{7.12}$$

where d_{ijk} is the delay to traverse an arc (i, j) of a commodity k and t_{ikp} corresponds to the time to execute service p from a commodity k in a node i.

In order to calculate the execution time of a service on a given application and a node i, we use the following equation:

$$t_{ikp} = \frac{C_{kp}}{CM_i} \tag{7.13}$$

where C_{kp} is the CPU needed to execute a service of a commodity and CM_i is the available CPU on a node i.

To calculate the energy spent on executing this service, the following equation is considered from the energy model described by Kaup et al. [2014]:

$$e_{ikp} = 1.5778\text{W} + 0.181 t_{ikp} \tag{7.14}$$

Regarding the calculation of the delay, we use the following equation:

$$d_{iju} = \frac{MS_k}{b_k} + PD_{ij} \tag{7.15}$$

where PD_{ij} corresponds to the propagation delay of link (i, j) and MS_k corresponds to the message size of application k.

Moreover, to calculate the energy consumed on this link, in Joule, we consider the following equation, described in Kaup et al. [2014]:

$$\bar{e}_{ijk} = 0.294\text{W} + 0.006 + 1.06e^{-3} b_k \frac{\text{W}}{\text{Mbps}} \tag{7.16}$$

$$\bar{e}_{jik} = 0.294\text{W} + 2.327e^{-3} b_k \frac{\text{W}}{\text{Mbps}} \tag{7.17}$$

where $\bar{e}_{ijk}$ is the energy consumed by downlink and $\bar{e}_{jik}$ is the energy consumed by uplink.

Similarly, as described in Section 7.2.1, the described objectives may be conflicting. Therefore, it is considered a min-max formulation as follows:

$$\min f(w, z) = \max(E(w, z), L(w, z)) \tag{7.18}$$

We consider the following constraints to describe the resources used by the services represented in the commodities. Since the execution of those services is sequential, we are interested in ensuring that each node has enough resources to execute the services with higher consumption of CPU (C), memory (M), and storage (S). We formalise the constraint for the CPU, yet the same applies for the memory and storage constraints:

$$\max_{\substack{k=1,\dots,K \\ p=1,\dots,q_k}}(w_{ikp}C_{kp}) \leq CM_i, \quad \forall i = 1,\dots,n-1 \tag{7.19}$$

$$\max_{\substack{k=1,\dots,K \\ p=1,\dots,q_k}}(w_{ikp}M_{kp}) \leq MM_i, \quad \forall i = 1,\dots,n-1 \tag{7.20}$$

$$\max_{\substack{k=1,\dots,K \\ p=1,\dots,q_k}}(w_{ikp}S_{kp}) \leq SM_i, \quad \forall i = 1,\dots,n-1 \tag{7.21}$$

Since we are assuming that each execution is sequential, it is necessary to ensure that each service has a waiting time if it is not the first to execute on a node. For the sake of comprehension, we define $(k,p) \mapsto h$, where each k represents a commodity and each p represents a service of that commodity, into an index $h = 1,\dots,H$. We express this scheduling constraint with mapped indices as follows (H. Papadimitrou and Steiglitz [1998]):

$$s_g + t_{ih} - s_h \leq T(1 - \delta_{hg} + 2 - y_{ih} - y_{ig}), \quad \forall i = 1,\dots,n, h \neq g, h, g = 1,\dots,H \tag{7.22}$$

$$s_g - s_h \leq T\delta_{hg}, \quad \forall h \neq g, h, g = 1,\dots,H \tag{7.23}$$

$$\delta_{hg} + \delta_{gh} = 1, \quad \forall h <_{lex} g, h, g = 1,\dots,H \tag{7.24}$$

where s_h is an integer variable that indicates the start (or waiting) of service h, T is the sum of all the execution times t_{ih}, and δ_{hg} is a slack binary variable that is 1 if the start time of service h is before service g and 0 otherwise. $<_{lex}$ defines the lexicographic ordering of the pair (k,p). If both services are on the same node, it is necessary to ensure that they do not occur at the same time. If both services are allocated on the same node, then y_{ih} and y_{ig} will be 1, which negates that part of the equation. Afterwards, if δ_{hg} is also 1, it indicates that service associated with index h must be executed before the one in index g. In that case, the slack variable T is negated and the constraint cannot be satisfied. These constraints ensure that two services that are assigned to a node i cannot be executed at the same time.

Each application has a deadline to be met, which is composed of the overall time to execute the corresponding service of each commodity, the time to reach the target node and the waiting time. We express this as a constraint as follows:

$$TD(w,z) = \sum_{i=1}^{n}\left(\sum_{p=1}^{q_k} w_{ikp}t_{ikp} + \sum_{j=1}^{n} z_{ijk}d_{ijk}\right) + \sum_{h=1}^{H_k} s_h \leq D_k, \quad \forall k = 1,\dots,K \tag{7.25}$$

where D_k is the deadline of commodity k and H_k corresponds to the set of indexes (k, p) with a fixed commodity k, in order to obtain the starting time of the services p of its commodity.

In order to ensure that each arc has enough bandwidth for a commodity and to ensure that, for each commodity, there is flow conservation, i.e., the information sent from the source to the target is the same, we define the following constraints:

$$\sum_{k=1}^{K} b_k z_{ijk} \leq \bar{b}_{ij}, \quad \forall i, j = 1, \ldots, n \tag{7.26}$$

$$\sum_{j=1}^{n} z_{ijk} - \sum_{j=1}^{n} z_{jik} = U_{ik}, \quad \forall i = 1, \ldots, n, k = 1, \ldots, K \tag{7.27}$$

where b_k is the bandwidth needed for the commodity k, $\bar{b}_{ij}$ is the available bandwidth on arc (i, j), and U_{ij} is 1 if index i corresponds to a source node, -1 if i corresponds to a target node and 0 otherwise. This ensures that there is conservation of flow: the sources have negative flow, the targets have positive flow and the transit nodes have neutral flow.

There is a need to link the variables z_{ijk} and w_{ikp}, i.e., to ensure that the services are computed on nodes belonging to the $s - t$ path (source-target). A way to ensure this constraint is to force the execution of a service on a node where it has, at least, one incident link used on the $s - t$ path for the commodity k. We formulate this constraint as follows:

$$w_{ikp} \leq \sum_{j=1}^{n} (z_{ijk} + z_{jik}), \quad \forall i = 1, \ldots, n, \\ k = 1, \ldots, K, p = 1, \ldots, q_k \tag{7.28}$$

To ensure that all the services of each commodity are executed only once, according to the number of services necessary to compute (q_k), we consider the following constraint:

$$\sum_{i=1}^{n} w_{ikp} = 1, \quad \forall k = 1, \ldots, K, p = 1, \ldots, q_k \tag{7.29}$$

7.3 Heuristics

In this section, heuristics to tackle the same problems on the previous section are described. Two settings are considered: offline and online. The first are used in offline scenarios, where there is only one execution of the heuristic. The latter are used in online scenarios, where there are users entering and leaving the area, and it is necessary to take into account the resources previously used. Moreover, we describe a dynamic reconfiguration approach to be used in realistic environments, where failures of links and/or nodes may occur on the network.

Algorithm 7.1: Static Virtual Network Embedding heuristic

Input: Graph G, set V of Virtual Network requests
Output: List L containing all the embedding of Virtual Networks to the substrate network

```
1  Create priority queue PQ containing all substrate nodes with value 0
2  for v = 1, ..., V do
3      Create queue Q containing VN m_v nodes
4      s_s ← PQ.top()
5      s_l ← Q.pop()
6      Map s_l into s_s and add to list L
7      PQ.update(s_s)
8      while Q is not empty do
9          s_l ← Q.pop()
10         path ← FindPath(s_s, s_l)
11         s_s ← path.lastNode
12         Map VN path into substrate path and add to list L
13         PQ.update(path.nodes)
14 return L
```

7.3.1 Virtual Network Embedding (Offline)

In order to have a faster execution, a heuristic to solve the VNE problem is provided. Algorithm 7.1, called SVNE (Static VN Embedding) aims to map VNs in an offline environment where all the requests are known a priori. As data structures, it requires a list L, where each entry contains the mapping of a VN into the substrate network, a min-priority-queue PQ containing the substrate nodes with value according to the min-max formulation (see Equation 7.3) and a queue Q with candidate nodes. For each VN request (line 2), queue Q is filled up with the candidate nodes (line 3). Then, the algorithm finds on PQ, the first virtual node r with smallest objective value according to the min-max formulation and node degree equal or lower than the substrate node i, i.e., selects the node at the top of the priority queue that satisfies the previous requirements. This ensures that all links of r can be mapped (line 4). Next, the first virtual node of the actual VN v is removed from the queue (line 5) to be mapped onto the previous obtained substrate node (line 6). The priority queue is updated with the new value of node i after having virtual node r mapped onto it (line 7).

Then, the algorithm traverses each link of the substrate network from a given node (lines 8-13). It starts at node i and finds the best link according to the min-max formulation (lines 9-10). FindPath() implements a single-source single-target shortest path algorithm, such as Dijkstra algorithm, that ensures the requested upper bound on bandwidth for each candidate link. If the link has enough available bandwidth, the virtual link is mapped into this substrate link (lines 11-12). Similarly, if the new node has enough capacity, the virtual node is mapped into the substrate node i or j, where j is the current target. Otherwise, the algorithm compares the min-max value of a different link (i to different node

Algorithm 7.2: Dynamic Virtual Network Embedding heuristic

Input: Graph G, set V of Virtual Network requests
Output: List F containing the main and secondary embeddings of the Virtual Networks into the substrate network

1 Create priority queue PQ containing all substrate nodes with value 0
2 **for** $v = 1, \ldots, V$ **do**
3 $L \leftarrow SVNM(G, v)$
4 $G' \leftarrow$ Copy of G, where the nodes and links obtained in L have min-max values set to ∞
5 $L' \leftarrow SVNM(G', v)$
6 Add L and L' to list F
7 **return** F

l), adds an extra node on the actual path (i to j to new node k) and chooses the one with lower value. This verification is done on the *FindPath* procedure on line 10. Moreover, if there is a loop path on the VN, as seen in Figure 7.2a, the algorithm will check the next mapped node, in order to ensure that the loop is also embedded on the substrate network. If it has been mapped, then a shortest path between those two nodes is found. The algorithm iterates until the VN has been mapped. Its time complexity is $\mathcal{O}(|V||N|(t_F + t_{|PQ|}))$, where t_F corresponds to the complexity of finding a path ($\mathcal{O}(|E| + n\log(n))$) for each pair of VN nodes on the substrate network and $t_{|PQ|}$ corresponds to the complexity of updating PQ ($\mathcal{O}(\log(n))$), where n is the number of nodes.

7.3.2 Virtual Network Embedding (Online)

Here, we consider two extended versions of the heuristic (Algorithm 7.1) and assume that each VN has a lifespan, i.e., after some time, the resources are released. First, we execute it in time-windows, i.e., SVNE is executed every instant t_w with the existing VN requests. If some resources are to be released before the time limit of the new request is reached, the *FindPath* method considers those resources. Otherwise, they cannot be used. The time complexity of the dynamic version is the same as the static version, since the only difference is the execution in time-windows, which does not change the performance.

Another extended version considers resilience, as described in Algorithm 7.2, called DVNE (Dynamic VN Embedding). For each VN request, DVNE generates a full link/node disjoint VN according to the fairness objectives. In DVNE (Algorithm 7.2) we generate a two disjoint VNs for each request. First, SVNE generates a VNE for a request $v \in V$. Next, the min-max values of nodes and links used on the first mapping are set to a very large value, in order to only be used if there is no other link or node available (line 3). Finally, SVNE is used again to generate a new VN with the updated values on the already obtained links/nodes. This way, we are able to generate a fully disjoint VN.

7.3.3 Virtual Network Reconfiguration

In this section, we describe the reoptimisation techniques to be used when a link or node fails. This implies that, when a failure occurs, it is necessary to reoptimise the VNs that used those links/nodes. This is useful when there are resource constraints on the substrate network and it may not be possible to have a secondary path allocated for each VN or the secondary path cannot be used. Despite being possible to use this approach to migrate partially or fully VNs when there is overload of resources in a specific area, we focus only on reoptimising the links/nodes where a failure occurs, since the proposed VNE formulation/heuristic already provides a distribution of resources taking fairness into account. If it is not possible to reoptimise the VN, the user is rejected and the heuristic will reallocate resources on the next time window. For this, we consider the following failures: *Link failure* and *Node failure*.

First, we focus on a single link failure. For this, we consider the following approaches:

- If a substrate link (i, j) fails and one or more VN are allocated on that link, then a new path is found to reconnect node i to node j.
- If a substrate link (i, j) fails and there is no VN using that link, then no action is performed.
- If a substrate link (i, j) fails and one or more substrate nodes become disconnected from the network, then the VNs associated are removed and the heuristic reallocates them in the next time window.

Next, we focus on a single node failure. This implies multiple link failures. For this, we have the following approaches:

- If a substrate node i fails and one or more VN are allocated on the node or the incident links, then a new path is found to reconnect the nodes. For example, if there is a VN using path $j \rightarrow i \rightarrow k$, then a new path from j to k is found.
- If a substrate node i fails and there is no VN that is using that node or the incident links, then no action is performed.
- If a substrate node i fails and the network becomes disconnected, then the VNs associated are removed and sent to the heuristic to be reallocated.

This allows us to have a fast reoptimisation on the network, i.e., obtain a good approximate solution without the need to full reconfiguration. The time complexity of the VN reconfiguration is $\mathcal{O}(t_F)$, where t_F corresponds to the complexity of finding a path.

7.3.4 Service Placement (Offline)

In the following, we propose a greedy heuristic to solve the problem of SP on a static scenario. Algorithm 7.3 presents the pseudocode of our approach.

Algorithm 7.3: Static Service Placement heuristic

Input: Set K of commodities, $n-1$ nodes
Output: Path st_k for each commodity and set N' with the services placed

$st_k \leftarrow$ empty list for each commodity $k \in K$
$N' \leftarrow$ array initialised to 0 with size $n-1$
Sort K by increasing order of deadline
foreach *commodity* $k \in K$ **do**
    $N'' \leftarrow$ empty list for the nodes on each path st_k
    Set all nodes in V with distance ∞
    Push s_k into Q with distance 0
    while Q *is not empty* **do**
        $u \leftarrow Q.pop()$
        $N''.append(u)$
        if $u = t_k$ **then**
            break
        foreach *link* $\{u, v\} \in G$ **do**
            if $bandwidth(u, v) \geq b'_k$ **then**
                $d \leftarrow accumulated_distance(u) + dist(u, v)$
                if $d < accumulated_distance(v)$ **then**
                    $accumulated_distance(v) \leftarrow d$
                    if $v \in Q$ **then**
                        $Q.update(v)$
                    else
                        $Q.push(v)$
    $latency \leftarrow$ latency of the path obtained
    Copy N'' into st_k
    Sort N'' according to the min-max formulation (Eq. 7.18)
    $node \leftarrow N''.top()$
    $Allocated \leftarrow false$
    foreach *service* $p \in q_k$ **do**
        while $Allocated = false$ ***or*** *node not null* **do**
            if $resources(node) \geq resources(p)$
            and $time(p) + latency \leq deadline(k)$ **then**
                $Allocate(N', node, p)$
                $Allocated \leftarrow true$
                $latency \leftarrow latency + time(p)$
            $node \leftarrow next(node)$
        if $Allocated = false$ **then**
            $Allocate(N', cloud, p)$
return st_k, N'

The algorithm obtains paths for each commodity while executing the corresponding services on a node, inspired on Dijkstra's algorithm. As data structures, we consider a list s_{tk}, which contains the path for each commodity $k \in K$, array N' with the services placed on each node i and a min-priority-queue Q,

which contains the candidate nodes for each link (i, j) in each iteration. Function *accumulated_distance*(u) obtains the sum of the min-max values of the existing links with node u, *bandwidth*(u, v) obtains the bandwidth of link u, v, and the available resources of node u are obtained by *resources*(u). Similarly, *resources*(p) obtains the necessary resources of service p, *Allocate*(N', u, p) allocates service p on node u (which can be the cloud) and updates the u entry on array N', *time*(p) obtains the time to execute of service p and *deadline*(k) obtains the deadline associated to commodity k.

First, a path is obtained for each commodity (lines 8-21). *dist*(u, v) corresponds to the energy consumption and bandwidth usage of link (u, v) by request k. Then, the algorithm chooses one of the transit nodes to execute the services (lines 27-35). The commodities need to be sorted in increasing order of deadline (line 3). Next, all nodes are set to an infinite distance from the source (line 6), the source node is inserted into a min-priority queue (line 7) and the algorithm iterates throughout the queue (line 8) until the target is found (lines 11-12). For each node obtained from the queue (line 9), the algorithm updates the distances of each link according to the min-max formulation (see Equation 7.18, line 15), if the link has enough bandwidth (line 14). Finally, when the algorithm reaches the target, a feasible path is found.

After the path has been defined, it is necessary to place the services. First, the node list is sorted according to the min-max formulation (see Equation 7.18, line 24) and the algorithm iterates through this list until the service is placed or the list ends (lines 27-33). If it is not possible to place the service, then it is sent to the cloud (lines 34-35). When a service is placed by the function *Allocate*$(\cdot, \cdot)$, N' is updated with the node placement of the service. Its time complexity is $\mathcal{O}(|K|(t_F + q_k))$, where t_F corresponds to the complexity of finding a path.

7.3.5 Service Placement (Online)

Here, we extend the previous heuristic to be able to use it in a dynamic scenario. We consider two extensions of the previous algorithm. First, we execute it in time-windows, i.e., the heuristic is executed every time-window t_w with the existing users requests. Second, we consider the resources being used as it is considered on the scheduling constraints on the formulation, but with the addition of bandwidth. For example, if the bandwidth necessary to use a link is superior than the available bandwidth, we also consider the time to wait until it has enough free bandwidth if the time to go to another link is higher than the waiting time until the resources are freed. The time complexity of the dynamic version is the same as the static version, since the only difference is the execution in time-windows and the increased time of waiting, which does not change the performance.

7.4 Experimental Results

In this section, we describe the experimental setup and analysis of our approaches. In order to implement them, we used the *Yet Another Fog Simu-*

Table 7.2.: Parameter values in the network (Foundation [2021])

	Network parameters	Values
Link	Bandwidth	5Gbps
	Propagation Delay	100ms
Node	CPU	9600 MIPS
	RAM	4Gb
	Storage	32Gb

Table 7.3.: Parameter values of the users' Virtual Networks (randomized)

	Users parameters	Values
Link	Bandwidth	[10-50]Mbps
Node	CPU	[50-100]MIPS
	RAM	[50-100]Mb
	Storage	[100-500]Mb

Table 7.4.: Parameter values of the users' services (randomized)

	Services parameters	Values
Link	Deadline	[10-20]s
Node	CPU	[5-10]MIPS
	RAM	[5-10]Mb
	Storage	[50-100]Mb

lator (YAFS) toolkit (Lera et al. [2019]) as a fog simulator. The linearised formulations of the MILP formulations were implemented in the *SCIP* solver (Bestuzheva et al. [2021]) with *PySCIPOpt* (Maher et al. [2016]). Every parameter with interval values was generated according to a random uniform distribution. Moreover, we implemented one related work approach (FTO, Zhang et al. [2018]) to compare to our service placement problem solution.

7.4.1 Setup

The experimental work consisted on having a set of users with different VN size topologies and different request values. For the network, we model the nodes as a Raspberry Pi device (Foundation [2021]), as seen in Table 7.2. It is considered uniformly random generated values for the network, users' VNs and services (see Tables 7.3 and 7.4 for the range of values) used in the randomised scenario. The realistic scenario parameters (Skarlat et al. [2017]; Navarro-Ortiz et al. [2020]) are described in Tables 7.5 and 7.6.

The YAFS simulator was configured as follows: first, the VN class was implemented to create VNs on the simulation since YAFS does not support VN. Afterwards, the formulations and heuristics were implemented and prepared to be used by the simulator. Next, the settings were modelled and the simulation was executed to create the VNs according to the results from our approaches and then obtaining the paths for the service placement to provide to the simu-

Table 7.5.: Parameter values of the users' Virtual Networks (real scenario (Skarlat et al. [2017]; Navarro-Ortiz et al. [2020]))

	Users parameters	**Values**
Link	Bandwidth	1.5Mbps
Node	CPU	250MIPS
	RAM	256Mb
	Storage	4Gb

Table 7.6.: Parameter values of the users' services (real scenario (Skarlat et al. [2017]))

	Services parameters	**Values**
Link	Deadline	120s
Node	CPU	100MIPS
	RAM	30Mb
	Storage	30Mb

lator. In the case of failures, we defined a random node/link each time window and the simulator indicated which users tried to use a failed node/link. In that case, we executed the reconfiguration approach and observed if it was possible to reconfigure the VNs which had failures associated to them.

On this experimental analysis, it is assumed that the substrate network and the VN requests have a random topology. Since it is considered a topology with several clusters, the inter-cluster connections have a higher energy consumption and higher latency.

In the VNE situation, we consider two versions: offline and online. In the first, we aim to understand how close the heuristic is to the formulation and which objective has the maximum value on each test. For the latter, we aim to analyse the improvement of having fairness and neighbour clusters.

After the VNE occurs, each user has its own set of applications that needs to be placed on the network. Moreover, each of these applications may have a set of services to compute, such as encoding on streaming services. In this case, we assume that each application has a source (the user) and a target, which may be an endpoint to a neighbour cluster that has a different user, or even to the cloud.

Similarly to the VNE experimental analysis, an offline version is considered to compare the formulation and the heuristic. Afterwards, it is also analysed an online version of the heuristic, where there is a set of application requests in each time window. For the latter, the aim is to analyse the cloud usage, the energy consumption and the latency.

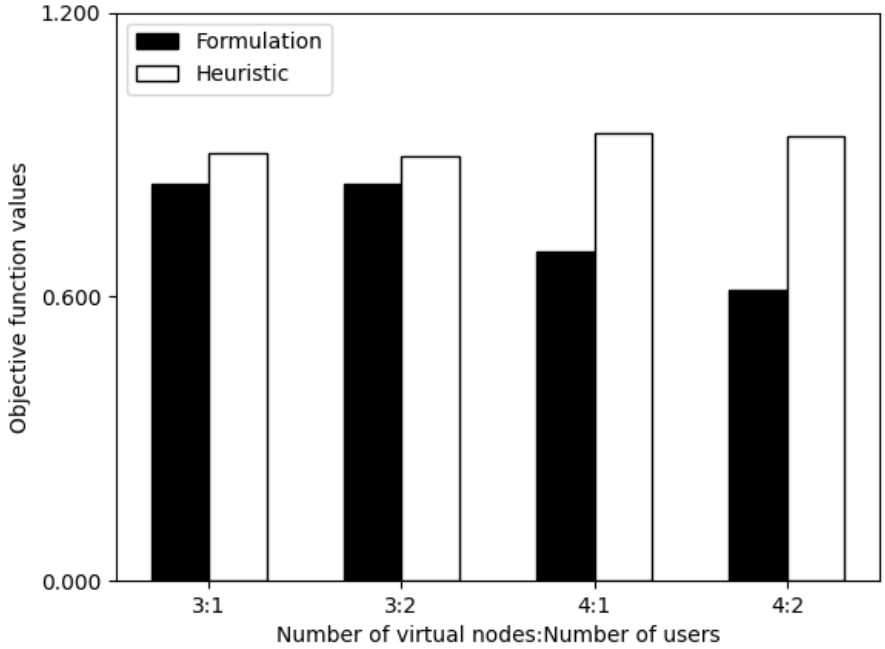


Figure 7.4.: Offline Virtual Network Embedding with topology size and number of users

7.4.2 Experimental Analysis

The approaches were evaluated as follows: 1) comparison of energy and latency objective function values on static scenarios; 2) comparison of average energy consumption and bandwidth usage used without reconfiguration; 3) comparison of rejected users with reconfiguration and VNE with fully disjoint node/link; 4) comparison of overall energy consumption and latency values and services offloaded to the cloud.

Due to the prohibitive execution time of the MILP formulation, we only considered it in 1), for which small test cases were considered. In the dynamic version, we consider the setting with randomised parameters and 10 runs for each combination of parameters, and the referenced setting.

7.4.2.1 Virtual Network Embedding (Offline)

In order to compare the results of the formulation and heuristic, it is considered an offline version with substrate network of 5 nodes and two sizes of users requests: 3 and 4 nodes. Moreover, 1 and 2 users in each configuration are used.

Figure 7.4 shows the objective function values for the VNE formulation and heuristic with 3 and 4 nodes and 1 to 2 users. As it is possible to observe, both formulation and heuristic perform better with less users, since there are less resources to allocate. However, when we have two users, the difference between

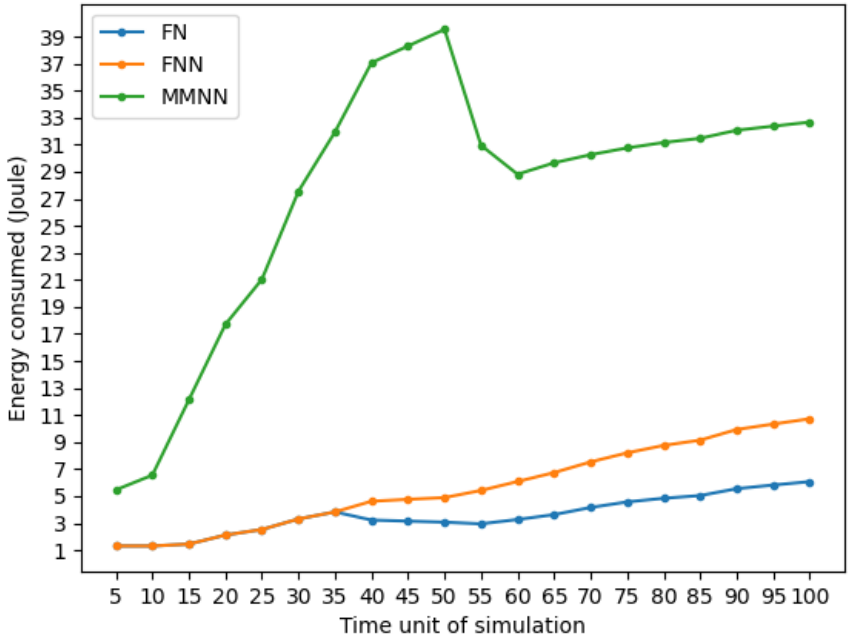


Figure 7.5.: Online Virtual Network Embedding (randomised setting), average energy consumption per node

them is less than when only one user exists. This may be due to the fact that the heuristic maps links and nodes according to the best solution at the time, which implies that it may not be the best mapping overall. Yet, with the increase of virtual resources to map, it has a better distribution of services since it chooses the node/link with lowest value between both objectives.

7.4.2.2 Virtual Network Embedding (Online)

In the online setting, the aim is to compare three different versions: usage of fairness without a neighbour cluster (FNN); usage of fairness with a neighbour cluster (FN); and minimisation of energy and maximisation of bandwidth without fairness and without a neighbour cluster (MMNN). For this, the approaches were evaluated in terms of the overall energy consumption and bandwidth and the number of neighbour nodes used on each time unit of simulation. In this scenario, we considered that each cluster is a complete graph with 50 nodes. Every 5 time units, there are 1 to 5 VN requests randomly generated, with different topology size (5, 10 and 20 nodes).

Figures 7.5 and 7.6 show the average energy consumption and bandwidth usage, respectively, for the randomised setting. Tables 7.7 and 7.8 show the standard deviation of energy consumption and bandwidth usage, respectively. The approaches focusing on fairness (FN and FNN) have the lowest energy consumption and bandwidth in average. This happens because the algorithm aims to distribute the resources along the network. Moreover, when using a neighbour cluster, the energy consumption and bandwidth are even lower because it uses

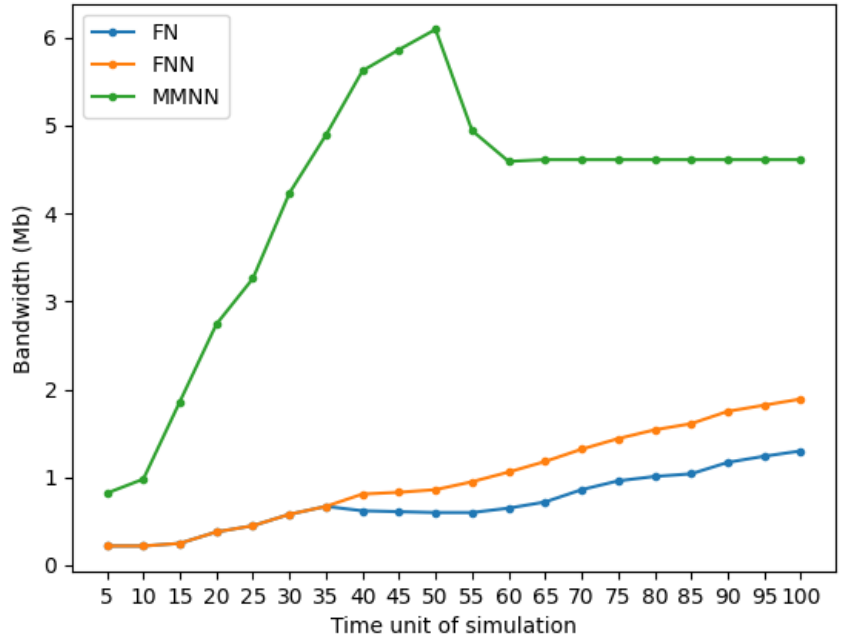


Figure 7.6.: Online Virtual Network Embedding (randomised setting), average bandwidth usage per link

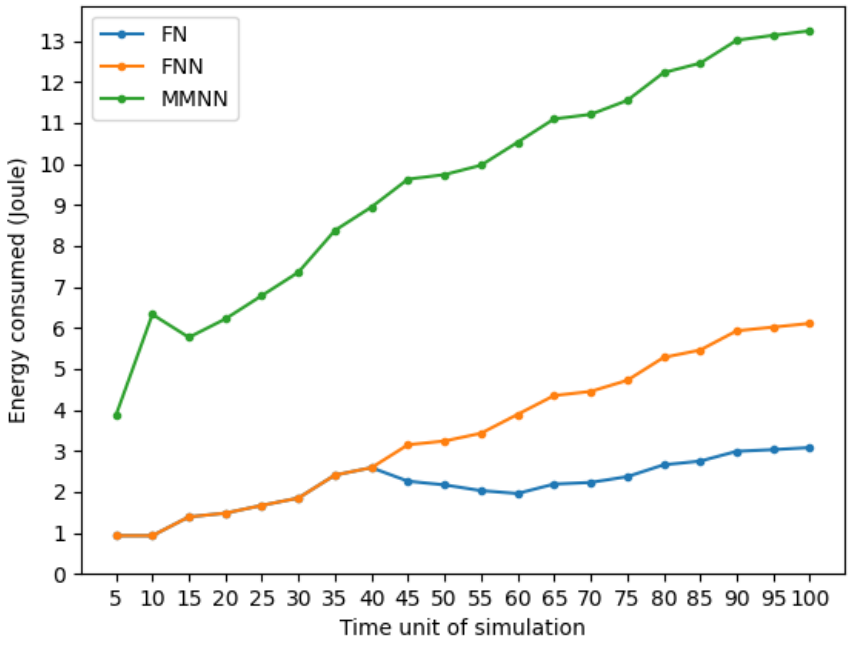


Figure 7.7.: Online Virtual Network Embedding (referenced setting), average energy consumption per node

more nodes. In the first moments of the simulation, both approaches allocate

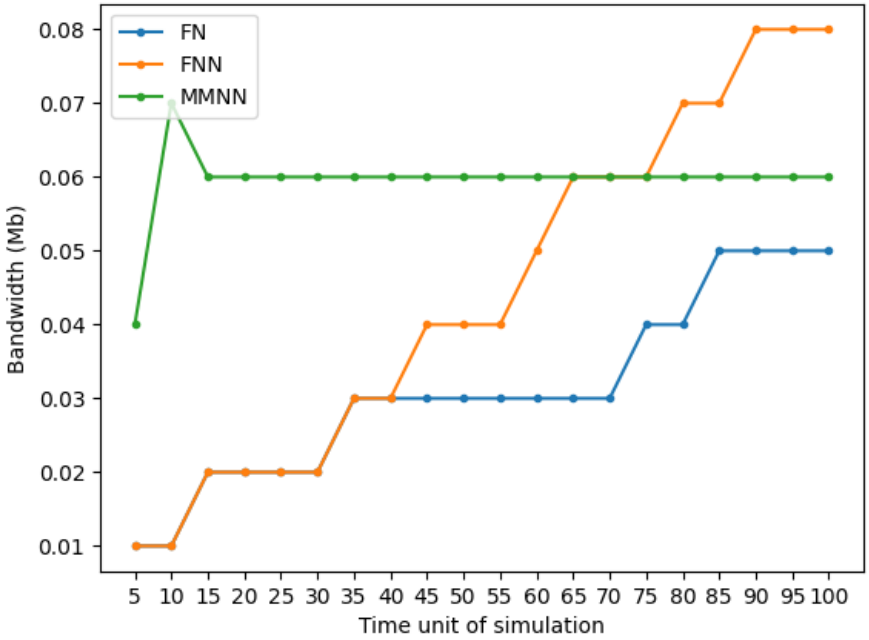


Figure 7.8.: Online Virtual Network Embedding (referenced setting), average bandwidth usage per link

the resources in the same locations, which present the same results in terms of energy consumption and bandwidth. When the neighbor cluster is used, there is a decrease on FN since there is a higher number of nodes and the average consumption decreases. When focusing on both objectives individually without fairness (MMNN), the energy consumption and bandwidth objectives have worse average and standard deviation values overall. MMNN aims to allocate services by minimising energy and maximising bandwidth, without focusing on fairness. This results in having some nodes and links overloaded and others underloaded, i.e., the distribution of resources in the overall network is not a concern in this approach. Then, the MMNN approach starts to reject most users due to being overloaded.

Figures 7.7 and 7.8 show the average energy consumption and bandwidth usage, respectively, for the referenced setting. Similarly to the previous experimental analysis, FN provides the lower average of both energy consumption and bandwidth, while providing the best distribution between nodes.

7.4.2.3 Virtual Network Reconfiguration

When the VNE has not generated a disjoint VN or when the VN generated is not available, for example, due to another failure or if it has not enough resources, we generate a new path. This implies that there may be one or more VN mapped into the link/node that is not available anymore. Thus, several users may be affected.

Table 7.7.: Online Virtual Network Embedding, average standard deviation for the average energy consumption

Time unit	**FN**	**FNN**	**MMNN**
5	0.66	0.57	1.89
10	0.64	0.60	2.24
15	0.43	0.79	4.16
20	0.66	1.16	6.10
25	0.40	1.29	7.24
30	0.65	1.72	9.51
35	0.42	1.94	11.02
40	0.68	1.67	12.79
45	0.67	1.58	13.17
50	0.63	1.48	13.56
55	0.38	1.19	14.03
60	0.67	1.56	14.62
65	0.38	1.71	15.01
70	0.64	1.27	15.41
75	0.57	1.08	15.76
80	0.70	1.36	16.03
85	0.44	1.50	16.24
90	0.70	1.25	16.68
95	0.59	0.96	16.89
100	0.45	0.78	17.11

Since it is assumed that these failures only occur on dynamic environments, i.e., when we have new users requests over time, we want to analyse how many VNs need to be reconfigured. When a VN is rejected, it is removed and sent to the heuristic to be reallocated in the next time window. Moreover, we compare the results to the ones obtained on the VN mapping, with and without resilience, to understand which one obtains better results.

In order to evaluate the performance of the reoptimisation heuristic, we assume a random link failure (Figure 7.9) and a random node failure (Figure 7.10) on each 5 time units (5, 10, ..., 100). On each figure we have the number of rejected VNs due to insufficiency of resources and the extra Virtual Networks to be reconfigured. In the case of VN mapping with fully node/link disjoint Virtual Networks, there is no VN to be reconfigured since each Virtual Network has a disjoint link/node.

Figure 7.9 shows similar results when considering VN with and without fully node/link disjoint VNs. This is due to the need to have extra VNs to ensure resilience, therefore the resources deplete faster and there is a higher number of VNs rejected before being allocated. In the setting where resilience is not considered, any time a link fails, all VNs using that link need to be reconfigured and are rejected, presenting an overall higher number of rejections. In this case, we obtain the best results when reconfiguration is considered, since it is possible to ensure, in most cases, different paths to allow connectivity. When

Table 7.8.: Online Virtual Network Embedding, average standard deviation for the average bandwidth usage

Time unit	FN	FNN	MMNN
5	0.12	0.10	0.33
10	0.11	0.11	0.39
15	0.09	0.15	0.74
20	0.13	0.21	1.10
25	0.08	0.24	1.31
30	0.12	0.31	1.70
35	0.10	0.35	1.97
40	0.14	0.30	2.25
45	0.14	0.28	2.34
50	0.13	0.25	2.42
55	0.08	0.15	2.50
60	0.14	0.22	2.56
65	0.09	0.25	2.57
70	0.14	0.18	2.57
75	0.13	0.11	2.57
80	0.16	0.15	2.57
85	0.12	0.15	2.57
90	0.16	0.18	2.57
95	0.15	0.17	2.57
100	0.13	0.15	2.57

there are node failures, as shown in Figure 7.10, the reconfiguration approach has closer results to the VN mapping with fully node/link disjoint VNs. This is due to the fact that when a node fails, all the associated links are deleted as well, which may imply a larger number of users to be reallocated. Yet, the reconfiguration approach shows that it provides lower number of rejections as it is able to reallocate several users and does not use as many resources as the VN mapping with fully node/link disjoint VNs.

In order to analyse the performance, we compare the time complexity of our work ($\mathcal{O}(t_F)$) with a base algorithm by Yang et al. [2018] ($\mathcal{O}(|E|(|E| + |V| \log |V|))$, where $|E|$ is the number of links and $|V|$ is the number of nodes. In our approach, the links with not enough bandwidth are discarded, which reduces the search space, and it is only necessary to search for links where VNs are allocated, making it faster to execute. Regarding execution time, our proposed approach obtains results in the range of ms for each run, even if there are several VNs to reconfigure, since it is based on Dijkstra's algorithm.

7.4.2.4 Service Placement (Offline)

In order to compare both formulation and heuristic, it is considered the offline version with six types of topologies: 10 to 15 nodes, 1 and 2 applications, with 2 services each.

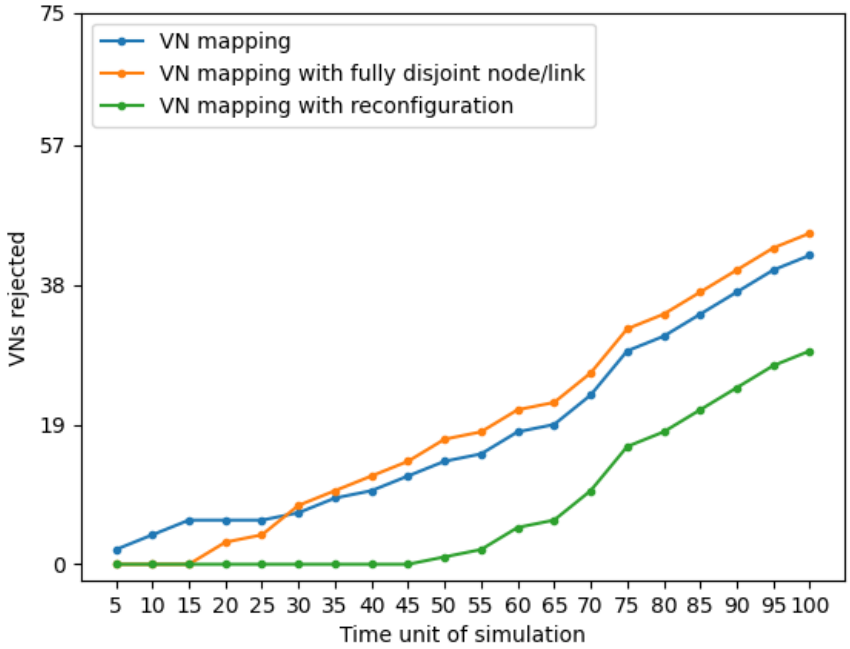


Figure 7.9.: Comparison of random link failures on different Virtual Network Embeddings

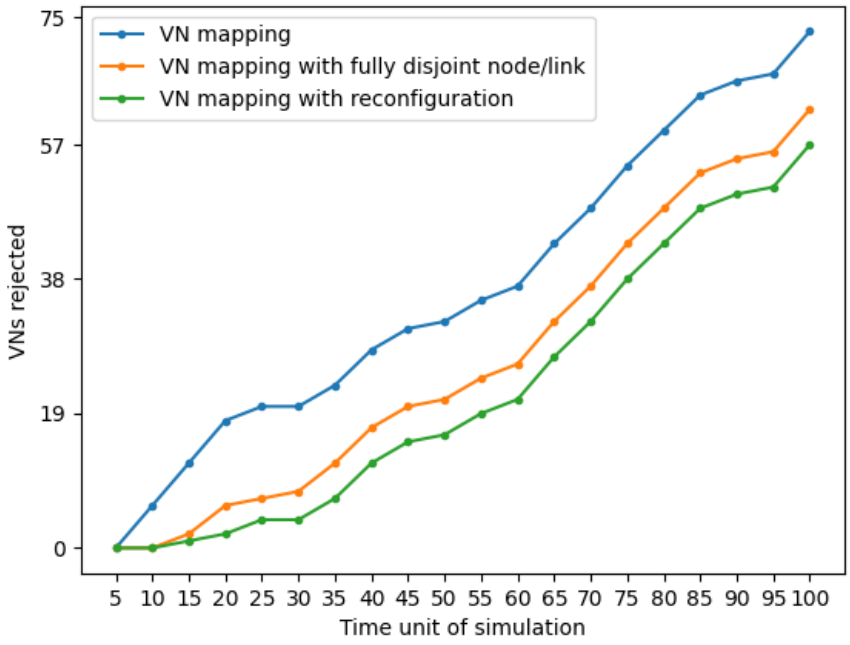


Figure 7.10.: Comparison of random node failures on different Virtual Network Embeddings

Figure 7.11 shows the objective function value results for the topologies with 10

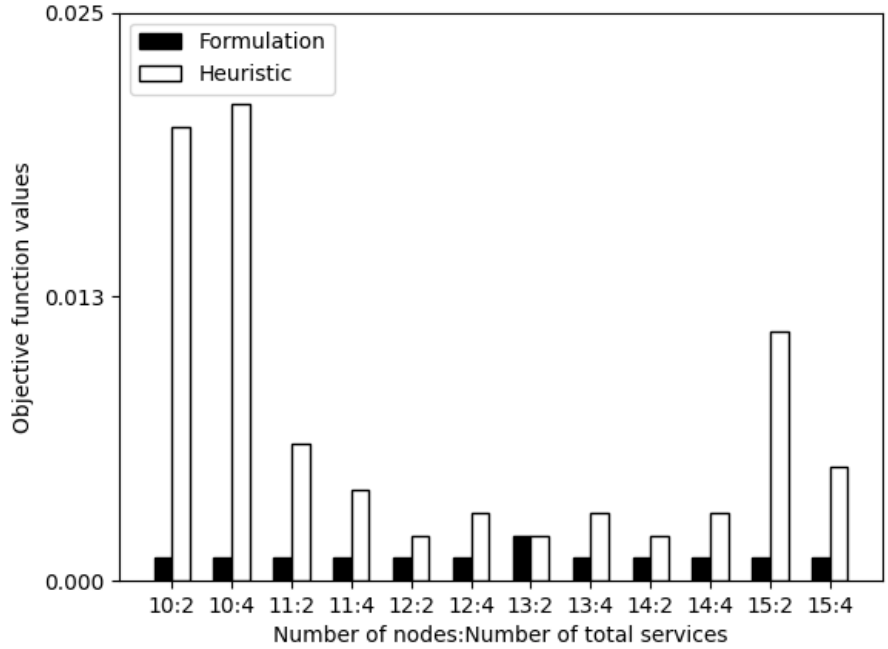


Figure 7.11.: Offline Service Placement formulation and heuristic objective function values, 10 to 15 nodes, 1 and 2 applications and 2 services each

to 15 nodes, respectively. It is possible to observe that, for smaller topologies, the obtained results are equal or close to the optimal, even if the value slightly increases when the number of applications increases. With 10 nodes, there is a larger difference between the heuristic and the formulation. This may occur since the path necessary to achieve the target is larger and the services may be placed on nodes that consume more energy or have a higher execution time.

7.4.2.5 Service Placement (Online)

In the online version, the aim to analyse the overall energy consumption and latency, and the cloud usage when different sets of applications are requested on each time window. Moreover, we present the results obtained by the FTO algorithm (Zhang et al. [2018]) to compare with our approach. In this case, it is considered a user topology composed by 25 nodes and 100 time units of simulation.

Each user has a request of 2 applications with 2 services each in each iteration. In order to analyse the load of the VN, we assume that the previous applications are still being used, i.e., the resources used by the previous allocations are not freed. Figures 7.12-7.14 show the obtained results for the randomised setting. In each time unit, we present the energy consumed (in Joule) and the latency (in s) on each link. Moreover, we present the percentage of services sent to the cloud when it is not possible to offload to the fog. It is possible to observe that our approach provides a solution with lower energy consumption and slightly larger latency on higher iterations. Our approach sends a slightly higher number of

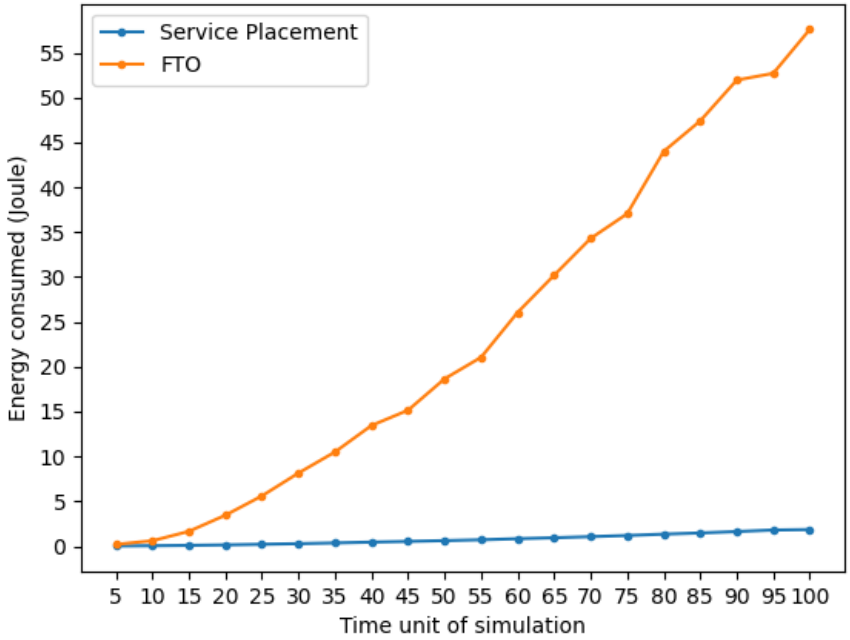


Figure 7.12.: Online Service Placement (randomised setting), overall energy consumption

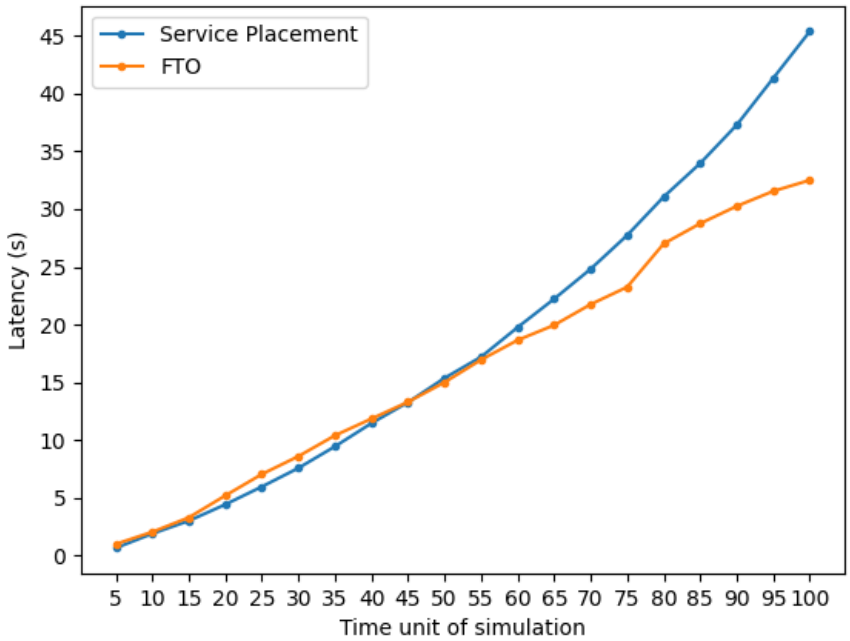


Figure 7.13.: Online Service Placement (randomised setting), overall latency

requests to the cloud overall, yet does not overload the network as much. This shows the advantage of our proposal when focusing on minimisation of energy

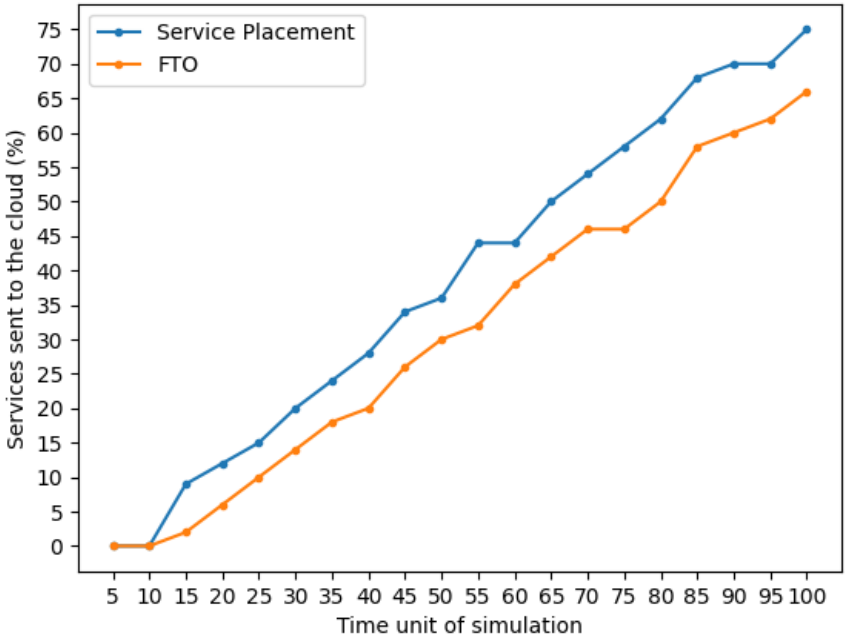


Figure 7.14.: Online Service Placement (randomised setting), percentage of services sent to the cloud

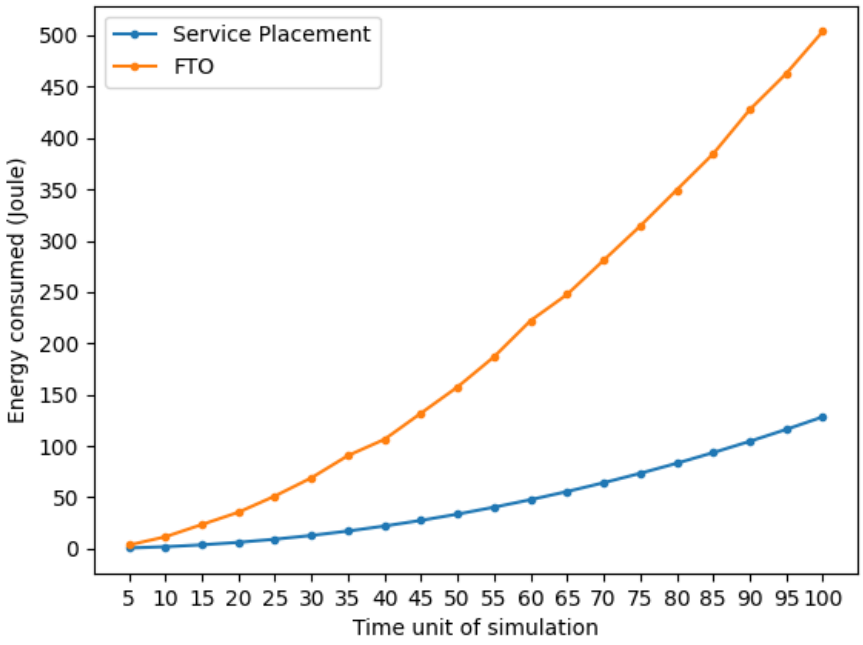


Figure 7.15.: Online Service Placement (referenced setting), overall energy consumption

consumption and latency since it is able to find a compromise solution between

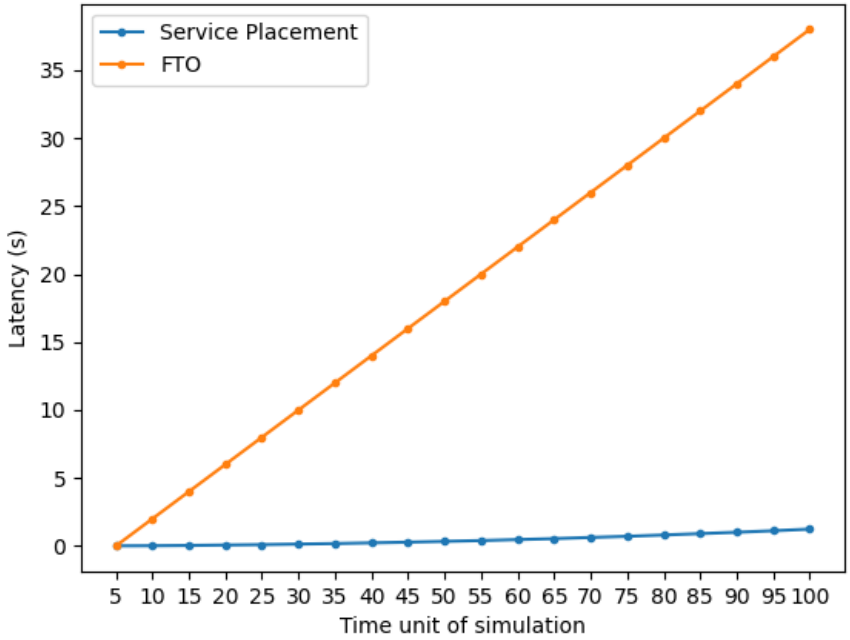


Figure 7.16.: Online Service Placement (referenced setting), overall latency

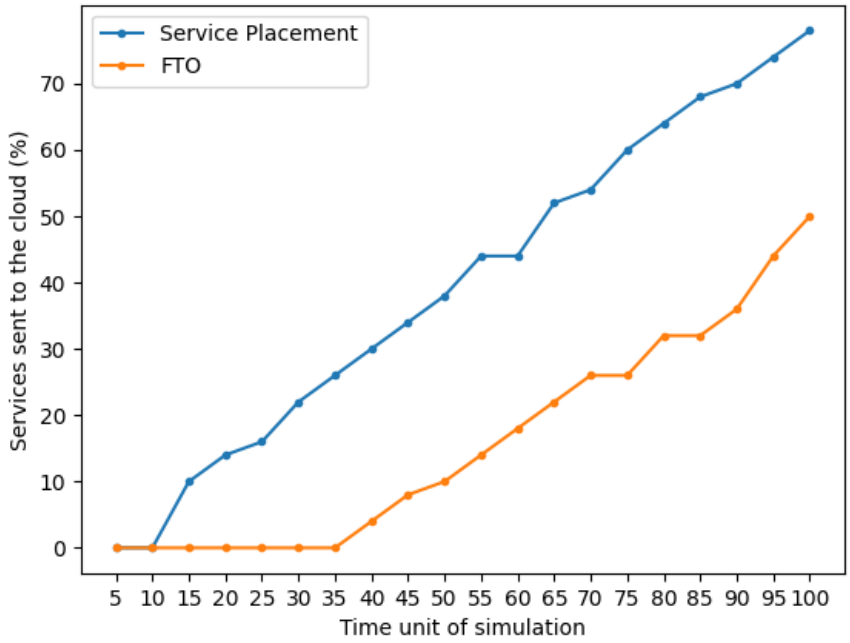


Figure 7.17.: Online Service Placement (referenced setting), percentage of services sent to the cloud

both objectives with lower value overall.

Figures 7.15-7.17 show the overall energy consumption, latency and services

sent to the cloud, respectively, for the referenced setting. In this analysis, it is possible to observe that our approach provides better results in terms of energy consumption and latency overall and sends a slightly higher number of services to the cloud.

7.5 Summary

In this chapter, a framework to provide Virtual Network Embedding, Virtual Network reconfiguration and Service Placement of users services in a fog environment was proposed. For the Virtual Network Embedding problem, a Mixed Integer Linear Programming formulation and heuristic with two objectives was described to maximise fairness of energy consumption and bandwidth. Moreover, a Virtual Network reconfiguration heuristic to be used when it is not possible to allocate additional resources was developed. This allows to prevent users rejection by reallocating only the Virtual Networks that were affected by the node/link fails. Finally, a Mixed Integer Linear Programming formulation and heuristic for the Service Placement problem, on the users Virtual Networks, with two objectives was presented, to minimise overall energy consumption and latency. It is possible to observe that the heuristics are able to obtain results close to the optimal in most offline settings. In the online version, the Virtual Network Embedding approach is able to obtain better distribution of resources when comparing to an approach that does not use fairness. The dynamic reconfiguration approach is able to reject less users than an approach based on allocating a node/link disjoint Virtual Network for each user and the Service Placement approach is able to consume less energy and has lower latency overall when compared to the state-of-the-art approach.

Chapter 8.

Conclusions and Future Work

In this thesis, methods to resource allocation problems that arise in fog computing are presented. Fog computing is an extension of cloud computing closer to users since Internet of Things (IoT) devices are fastly increasing and resource-constrained, which makes them unsuited to large computation tasks or communications to the cloud. To solve this, fog computing was proposed as a layer in a network architecture composed of a user layer, a fog layer and a cloud layer. Its most important characteristics are to allow low latency communication, temporary storage, low energy consumption, and to save bandwidth to the cloud. The focus on this thesis is to optimise emerging resource allocation problems, such as Service Placement, Virtual Network Embedding and Reconfiguration/Migration with Mixed Integer Linear Programming formulations and heuristics to obtain compromise solutions and taking into account optimisation models that promote the fair use of resources.

Chapter 2 describes different concepts used throughout this thesis. Starting from the description of the Cloud-Fog Continuum, Network Virtualisation and Network Slicing, followed by the resource problems to solve in this architecture, namely routing, Service Placement, Virtual Network Embedding and Reconfiguration and Migration. Finally, a brief introduction to fairness, Multi-objective Optimisation, Integer Linear Programming and heuristics is also provided.

Afterwards, a comparison between different works is made in Chapter 3. The aim is to analyse the most common metrics and objectives, how the problems can be formulated, and what type of approaches can be used to solve these problems. Considering the communications perspective, an analysis of the implemented architecture, the types of tools and resources, and the applications in realistic scenarios is performed. With this, several major gaps are identified in the literature:

- In most cases, a direct comparison between the optimal solution and heuristic methods is not conducted. This lack of comparison may result from the complexity of obtaining optimal solutions or the assumption that heuristic approaches are sufficient for practical applications. As a consequence, the performance gap between these two types of solutions often remains unexplored, leaving room for potential improvements and more informed decision-making in the application of these techniques.

- The majority of existing methodologies in this field tackle the complexity of multi-objective problems by applying some form of transformation that reduces the problem to a single-objective version. This transformation usually involves techniques such as weighted sum approaches, scalarisation, or goal programming, where multiple objectives are combined into a single objective function. This simplification, while making the problem more tractable, may sometimes lead to a loss of the rich information present in the original multi-objective framework, potentially impacting the overall effectiveness of the solution.
- Hybrid approaches, which integrate multiple aspects such as Service Placement, Virtual Network Embedding, and Reconfiguration/Migration within a unified framework, are generally underdeveloped. The lack of such integrated solutions may be due to the inherent complexity of simultaneously addressing these interconnected challenges. As a result, current research often treats these issues in isolation, which could lead to suboptimal outcomes when applied in real-world scenarios where these processes are interdependent and need to be managed collectively.
- Numerous studies addressing the Virtual Network Embedding problem fail to specify or assume any particular network architecture. This omission can limit the applicability and generalisability of the proposed solutions, as different architectures may present unique challenges and constraints. By not detailing the underlying architectural assumptions, these works might overlook critical factors that influence the performance and feasibility of the VNE solutions in practical network environments.
- A significant number of studies in this domain rely solely on numerical analysis to assess the performance of their proposed approaches, rather than employing simulation tools for validation. While numerical analysis can provide valuable insights and theoretical understanding, it may not fully capture the dynamic and stochastic nature of real-world systems. Simulators, on the other hand, can offer a more comprehensive evaluation by modeling complex interactions and uncertainties, leading to a more robust validation of the proposed methodologies.
- The majority of research in this area tends to overlook the incorporation of realistic service values in their analyses, opting instead to use randomly generated parameters for experimentation. This approach, while simplifying the evaluation process, may lead to results that are not fully representative of actual service conditions. The use of random parameters can obscure the practical applicability and effectiveness of the proposed solutions, as they may not adequately reflect the constraints, demands, and behaviors encountered in real-world scenarios. Incorporating realistic service values would enhance the relevance and impact of the research findings.

Chapter 4 describes a solution to the Service Placement problem, with the aim of evaluating the performance of the proposed approaches in a static two-layered network architecture, composed by fog and cloud. The fog consists of clusters,

where each contained a master node and a set of fog nodes. In order to place the services, two objectives are considered: maximisation of fog usage and maximisation of energy fairness throughout the system, using a min-max technique. For this, a Mixed Integer Linear Programming formulation and a heuristic are proposed, which model both objectives taking into account the devices' constraints and the services' requirements. Both approaches are compared using a set of randomly generated applications composed by different deadlines and services. While the formulation distributed better the services regarding energy consumption, the heuristic distributed better the number of services. Moreover, both approaches provide similar usage of the fog.

Chapter 5 proposes a Mixed Integer Linear Programming formulation and a heuristic for the data communication and migration problem after the change of location of a set of users by minimising energy consumption and latency in a fog environment. These approaches, as well as a baseline heuristic for reference purposes, are implemented in a simulator and tested with different topology sizes, number of users and different energy and latency values for migration and data communication. The results show that the proposed heuristic obtains results close to the optimal in most scenarios while avoiding rejection of users and obtaining paths with low number of hops.

Chapter 6 details a Mixed Integer Linear Programming formulation and heuristic to solve the Virtual Network Embedding problem in fog environments with Network Slicing architectures. The aim is to maximise fairness of energy consumption and resource/slice interference in two different settings. First, we assumed a set of current requests that need resources and, secondly, there exist a set of future known requests that need resources in a given time window. A range of scenarios has been considered. The presented results show that the heuristic is able to provide results close to the optimal, while allocating Virtual Networks with low energy consumption and low resource/slice interference.

Chapter 7 aims to develop a framework to provide Virtual Network Embedding, Virtual Network reconfiguration and Service Placement of users services' in static and dynamic fog environments. For the Virtual Network Embedding problem, a Mixed Integer Linear Programming formulation and heuristic are proposed with two objectives: maximise fairness of energy consumption and bandwidth. Moreover, a Virtual Network reconfiguration heuristic is proposed to be used when it is not possible to allocate additional resources. This allowed to prevent users' rejection by reallocating only the Virtual Networks that are affected by the node/link fails. Finally, a Mixed Integer Linear Programming formulation and heuristic are proposed for the Service Placement problem, on the users Virtual Networks, with two objectives: minimise overall energy consumption and latency. We observed that the heuristics are able to obtain results close to the optimal in most settings. In the online versions, the Virtual Network Embedding approach is able to obtain better distribution of resources when comparing to an approach that did not use fairness. The dynamic reconfiguration approach is able to reject less users than an approach based on allocating a node/link disjoint Virtual Network for each user and the service placement approach used less energy and had lower latency overall when compared to the state-of-the-art

approach.

As future work, the following proposals are described:

- To develop and propose tighter heuristics aimed at achieving improved performance in larger and more complex environments. While the current heuristics demonstrate a strong ability to approximate the optimal solution effectively in smaller settings, their scalability and efficacy in larger environments remain uncertain. This raises concerns about their robustness and adaptability when faced with the increased complexity, scale, and diverse challenges presented by larger systems. By refining these heuristics, it is possible to enhance their performance and ensure that they remain competitive and efficient even as the problem size grows, thereby making them more applicable to real-world scenarios where large-scale environments are common.
- To implement the proposed Mixed Integer Linear Programming formulations and heuristics within real-world testbeds to assess their practical effectiveness. While simulators provide a controlled environment to evaluate the performance of these algorithms, they often lack the complexity and unpredictability of real network conditions. This gap makes it difficult to fully understand how these approaches will perform when applied to actual connections and live data. By testing in real-world settings, we can gain valuable insights into how well the algorithms optimise resource usage under varying and unpredictable conditions, thereby validating their practicality and identifying any potential shortcomings or areas for further refinement. One example is the OREOS project, where some Mixed Integer Linear Programming formulations are being developed to be implemented in MiniZinc and integrated into testbeds.
- To evaluate the energy consumption and time lag of the proposed heuristics. In fog computing, minimising energy consumption and reducing delay are critical objectives. Although the proposed approaches have shown improvements in these areas, additional research is required to fully address these challenges. For instance, a deeper understanding is needed of how the execution of algorithms affects the energy consumption of individual devices. Additionally, the impact of the time lag between making a decision and the actual allocation of resources on overall delay remains unclear. Addressing these gaps will be essential for developing more efficient and responsive fog computing solutions.
- To broaden the scope of connectivity to accommodate a diverse array of devices, including both terrestrial and non-terrestrial entities, as we move towards the deployment of next-generation networks. This expansion will not only increase the sheer number and variety of connected devices but will also significantly heighten the complexity of managing and optimising these networks. For instance, aerial devices such as satellites, which follow predictable orbits, may only be accessible during limited time windows, whereas other devices like drones might appear sporadically and unpredictably. Researching how these intermittently available devices can

be leveraged to enhance key metrics in fog computing environments is of paramount importance. This includes understanding their potential to improve latency, bandwidth utilisation, and overall network resilience, thereby contributing to the development of more robust and efficient next-generation network architectures.

Bibliography

Afolabi, I., Ksentini, A., Bagaa, M., Taleb, T., Corici, M., and Nakao, A. (2017). Towards 5g Network Slicing over Multiple-Domains. *IEICE Transactions on Communications*, E100.B(11):1992–2006.

Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., and Flinck, H. (2018). Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions. *IEEE Communications & Surveys Tutorials*, 20(3):2429–2453.

Agarwal, S., Yadav, S., and Yadav, A. K. (2016). An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing. *International Journal of Information Engineering and Electronic Business(IJIEEB)*, 8(1):48.

Aissi, H., Bazgan, C., and Vanderpooten, D. (2009). Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197:427–438.

Alfoudi, A. S. D., Newaz, S. H. S., Otebolaku, A., Lee, G. M., and Pereira, R. (2019). An Efficient Resource Management Mechanism for Network Slicing in a LTE Network. *IEEE Access*, 7:89441–89457. Conference Name: IEEE Access.

Applegate, D. L., Bixby, R. E., and Cook, V. C. W. J. (2011). *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics; 17. Princeton University Press, course book edition.

Azimzadeh, F. and Biabani, F. (2017). Multi-objective job scheduling algorithm in cloud computing based on reliability and time. In *2017 3th International Conference on Web Research (ICWR)*, pages 96–101, Tehran, Iran. IEEE.

Barakabitze, A. A., Ahmad, A., Mijumbi, R., and Hines, A. (2020). 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 167:106984.

Bertsimas, D. and Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. Athena Scientific series in optimization and neural computation. Athena Scientific.

Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S. J., Matter, F., Mühmer, E., Müller, B., Pfetsch, M. E., Rehfeldt, D., Schlein, S., Schlösser, F., Serrano, F., Shinano, Y., Sofranac, B., Turner, M., Vigerske, S., Wegscheider, F., Wellner, P., Weninger,

D., and Witzig, J. (2021). The SCIP Optimization Suite 8.0. Technical report, Optimization Online.

Bi, Y., Han, G., Lin, C., Deng, Q., Guo, L., and Li, F. (2018). Mobility Support for Fog Computing: An SDN Approach. *IEEE Communications Magazine*, 56(5):53–59.

Bizanis, N. and Kuipers, F. A. (2016). SDN and Virtualization Solutions for the Internet of Things: A Survey. *IEEE Access*, 4:5591–5606.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the Internet of Things. *Proceedings of the MCC Workshop on Mobile Cloud Computing*, 1:13–15.

Cao, H., Du, J., Zhao, H., Luo, D. X., Kumar, N., Yang, L., and Yu, F. R. (2021). Resource-Ability Assisted Service Function Chain Embedding and Scheduling for 6G Networks With Virtualization. *IEEE Transactions on Vehicular Technology*, 70(4):3846–3859.

Cerroni, W. and Callegati, F. (2014). Live migration of virtual network functions in cloud-based edge networks. In *2014 IEEE International Conference on Communications (ICC)*, pages 2963–2968, Sydney, NSW. IEEE.

Chang, Z., Zhou, Z., Ristaniemi, T., and Niu, Z. (2017). Energy Efficient Optimization for Computation Offloading in Fog Computing System. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Singapore. IEEE.

Chowdhury, N. M. K. and Boutaba, R. (2010). A survey of network virtualization. *Computer Networks*, 54(5):862–876.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, 3 edition.

Dalvandi, A., Gurusamy, M., and Chua, K. C. (2015). Time-Aware VMFlow Placement, Routing, and Migration for Power Efficiency in Data Centers. *IEEE Transactions on Network and Service Management*, 12(3):349–362.

Deng, R., Lu, R., Lai, C., Luan, T. H., and Liang, H. (2016). Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption. *IEEE Internet of Things Journal*, 3(6):1171–1181.

Dimolitsas, I., Dechouniotis, D., and Papavassiliou, S. (2023). Time-efficient distributed virtual network embedding for round-trip delay minimization. *Journal of Network and Computer Applications*, page 103691.

DOCOMO 5G White Paper (2014). 5G Radio Access: Requirements, Concept, and Technologies.

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer-Verlag, Berlin Heidelberg, 2nd edition.

Eramo, V., Ammar, M., and Lavacca, F. G. (2017). Migration Energy Aware Reconfigurations of Virtual Network Function Instances in NFV Architectures. *IEEE Access*, 5:4927–4938. Conference Name: IEEE Access.

Eramo, V., Miucci, E., and Ammar, M. (2014). Study of Migration Policies in Energy-Aware Virtual Router Networks. *IEEE Communications Letters*, 18(11):1919–1922. Conference Name: IEEE Communications Letters.

Feng, J., Zhao, L., Du, J., Chu, X., and Yu, F. R. (2018). Energy-Efficient Resource Allocation in Fog Computing Supported IoT with Min-Max Fairness Guarantees. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6.

Fischer, A., Botero, J. F., Beck, M. T., de Meer, H., and Hesselbach, X. (2013). Virtual Network Embedding: A Survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906.

Fossati, F., Moretti, S., Perny, P., and Secci, S. (2020). Multi-Resource Allocation for Network Slicing. *IEEE/ACM Transactions on Networking*, 28(3):1311–1324. Conference Name: IEEE/ACM Transactions on Networking.

Foundation, R. P. (2021). Raspberry Pi 4 Model B Specification. `https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/`. [Online; accessed 01-September-2021]].

Frascolla, V., Miatton, F., Tran, G. K., Takinami, K., De Domenico, A., Strinati, E. C., Koslowski, K., Haustein, T., Sakaguchi, K., Barbarossa, S., and Barberis, S. (2017). 5g-MiEdge: Design, standardization and deployment of 5g phase II technologies: MEC and mmWaves joint development for Tokyo 2020 Olympic games. In *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 54–59, Helsinki, Finland. IEEE.

Gao, L. and Rouskas, G. N. (2018). Virtual Network Reconfiguration with Load Balancing and Migration Cost Considerations. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 2303–2311, Honolulu, HI. IEEE.

Gao, X., Ye, Z., Fan, J., Zhong, W., Zhao, Y., Cao, X., Yu, H., and Qiao, C. (2015). Virtual Network Mapping for Multicast Services With Max–Min Fairness of Reliability. *Journal of Optical Communications and Networking*, 7(9):942.

Godinho, N., Curado, M., and Paquete, L. (2019). Optimization of Service Placement with Fairness. In *2019 IEEE Symposium on Computers and Communications (ISCC) (IEEE ISCC 2019)*, Barcelona, Spain.

Godinho, N., Silva, H., Curado, M., and Paquete, L. (2020). Energy and latency-aware resource reconfiguration in fog environments. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pages 1–8.

Godinho, N., Silva, H., Curado, M., and Paquete, L. (2022). A reconfigurable resource management framework for fog environments. *Future Generation Computer Systems*, 133:124–140.

Gomes, R. L., Bittencourt, L. F., Madeira, E. R. M., Cerqueira, E., and Gerla, M. (2017). Two-criteria Pareto frontier for virtual network allocation on Edge-as-a-Service networks. *Computer Communications*, 102:58–66.

Guerzoni, R., Trivisonno, R., Vaishnavi, I., Despotovic, Z., Hecker, A., Beker, S., and Soldani, D. (2014). A novel approach to virtual networks embedding for SDN management and orchestration. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–7, Krakow, Poland. IEEE.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296.

H. Papadimitrou, C. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., 1st edition.

Hanauer, K., Henzinger, M., Ost, L., and Schmid, S. (2023). Dynamic Demand-Aware Link Scheduling for Reconfigurable Datacenters. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10. ISSN: 2641-9874.

Hu, P., Dhelim, S., Ning, H., and Qiu, T. (2017). Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications*, 98:27–42.

Huang, X., Ganapathy, S., and Wolf, T. (2009). Evaluating Algorithms for Composable Service Placement in Computer Networks. In *2009 IEEE International Conference on Communications*, pages 1–6, Dresden, Germany. IEEE.

Iwai, T. and Nakao, A. (2021). Bandwidth Allocation with Slice Quality Fairness in Network Slicing under Variable Link Capacity. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

Jain, R. K., Chiu, D.-M. W., and Hawe, W. R. (1984). A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *Technical Report DEC-TR-301, Digital Equipment Corporation*, pages 1–37.

Jalalian, A., Yousefi, S., and Kunz, T. (2023). Network slicing in virtualized 5G Core with VNF sharing. *Journal of Network and Computer Applications*, 215:103631.

Jia, M., Cao, J., and Yang, L. (2014). Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 352–357.

Jiang, D., Zhang, P., Lv, Z., and Song, H. (2016). Energy-Efficient Multi-Constraint Routing Algorithm With Load Balancing for Smart City Applications. *IEEE Internet of Things Journal*, 3(6):1437–1447.

JSCIPOpt (2016). JSCIPOpt. `https://github.com/scipopt/JSCIPOpt`. [Online; accessed 01-July-2023]].

Kattepur, A., Dohare, H., Mushunuri, V., Rath, H. K., and Simha, A. (2016). Resource Constrained Offloading in Fog Computing. In *Proceedings of the 1st Workshop on Middleware for Edge Clouds & Cloudlets*, MECC '16, pages 1–6, New York, NY, USA. ACM.

Kaup, F., Gottschling, P., and Hausheer, D. (2014). Powerpi: Measuring and modeling the power consumption of the raspberry pi. In *39th Annual IEEE Conference on Local Computer Networks*, pages 236–243.

Kayal, P. and Liebeherr, J. (2019). Autonomic Service Placement in Fog Computing. In *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pages 1–9, Washington, DC, USA. IEEE.

Khai, N. T., Baumgartner, A., and Bauschert, T. (2017). A multi-step model for migration and resource reallocation in virtualized network infrastructures. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 730–735, Atlanta, GA. IEEE.

Laghrissi, A. and Taleb, T. (2018). A Survey on the Placement of Virtual Resources and Virtual Network Functions. *IEEE Communications Surveys & Tutorials*, pages 1–1.

Leconte, M., Paschos, G. S., Mertikopoulos, P., and Kozat, U. C. (2018). A Resource Allocation Framework for Network Slicing. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 2177–2185.

Lera, I., Guerrero, C., and Juiz, C. (2019). YAFS: A Simulator for IoT Scenarios in Fog Computing. *IEEE Access*, 7:91745–91758.

Li, T. and Zhu, Z. (2023). QoS-Aware Management Reconfiguration of vNF Service Trees With Heterogeneous NFV Platforms. *IEEE Transactions on Network and Service Management*, 20(2):2013–2024. Conference Name: IEEE Transactions on Network and Service Management.

Li, X., Zhou, Z., Zhu, C., Shu, L., and Zhou, J. (2022). Online Reconfiguration of Latency-Aware IoT Services in Edge Networks. *IEEE Internet of Things Journal*, 9(18):17035–17046. Conference Name: IEEE Internet of Things Journal.

Ma, F. and Zhang, L. (2015). Multi-objective optimization for dynamic virtual machine management in cloud data center. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 170–174, Beijing, China. IEEE.

Maher, S., Miltenberger, M., Pedroso, J. P., Rehfeldt, D., Schwarz, R., and Serrano, F. (2016). PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pages 301–307. Springer International Publishing.

Martin-Perez, J., Malandrino, F., Chiasserini, C. F., Groshev, M., and Bernardos, C. J. (2022). KPI Guarantees in Network Slicing. *IEEE/ACM Transactions on Networking*, 30(2):655–668.

Medeiros, V. N., Silvestre, B., and Borges, V. C. M. (2019). Multi-objective routing aware of mixed IoT traffic for low-cost wireless Backhauls. *Journal of Internet Services and Applications*, 10(1):9.

Medhi, D. and Ramasamy, K. (2007). *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Mei, C., Liu, J., Li, J., Zhang, L., and Shao, M. (2020). 5G network slices embedding with sharable virtual network functions. *Journal of Communications and Networks*, 22(5):415–427.

Melo, M., Sargento, S., and Carapinha, J. (2016). Optimal virtual network migration: A step closer for seamless resource mobility. *Journal of Network and Computer Applications*, 64:124–136.

Melo, M., Sargento, S., Killat, U., Timm-Giel, A., and Carapinha, J. (2013). Optimal Virtual Network Embedding: Node-Link Formulation. *IEEE Transactions on Network and Service Management*, 10(4):356–368.

Meskar, E. and Liang, B. (2020). Fair multi-resource allocation in mobile edge computing with multiple access points. In *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pages 11–20, Virtual Event USA. ACM.

Miettinen, K. (1998). *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management Science. Springer US.

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., and Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262.

Minh, Q. T., Nguyen, D. T., Van Le, A., Nguyen, H. D., and Truong, A. (2017). Toward service placement on fog computing landscape. In *2017 4th NAFOSTED Conference on Information and Computer Science*, pages 291–296.

Mouaci, A., Gourdin, E., Ljubic, I., and Perrot, N. (2020). Virtual Network Functions Placement and Routing Problem: Path formulation. In *2020 IFIP Networking Conference (Networking)*, pages 55–63.

Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R. H., Morrow, M. J., and Polakos, P. A. (2018). A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials*, 20(1):416–464.

Mukherjee, M., Shu, L., and Wang, D. (2018). Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. *IEEE Communications Surveys Tutorials*, PP(99):1–1.

Müller-Merbach, H. (1981). Heuristics and their design: a survey. *European Journal of Operational Research*, 8(1):1–23.

Nace, D. and Pioro, M. (2008). Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial. *IEEE Communications Surveys & Tutorials*, 10(4):5–17.

Navarro-Ortiz, J., Romero-Diaz, P., Sendra, S., Ameigeiras, P., Ramos-Munoz, J. J., and Lopez-Soler, J. M. (2020). A Survey on 5G Usage Scenarios and Traffic Models. *IEEE Communications Surveys & Tutorials*, 22(2):905–929.

Nishio, T., Shinkuma, R., Takahashi, T., and Mandayam, N. B. (2013). Service-oriented Heterogeneous Resource Sharing for Optimizing Service Latency in Mobile Cloud. In *Proceedings of the First International Workshop on Mobile Cloud Computing & Networking*, MobileCloud '13, pages 19–26, New York, NY, USA. ACM.

Noghani, K. A., Kassler, A., and Taheri, J. (2019). On the Cost-Optimality Trade-off for Service Function Chain Reconfiguration. In *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, pages 1–6, Coimbra, Portugal. IEEE.

Németh, B., Sonkoly, B., Rost, M., and Schmid, S. (2016). Efficient service graph embedding: A practical approach. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 19–25.

Padhy, S. and Chou, J. (2021). Reconfiguration Aware Orchestration for Network Function Virtualization With Time-Varied Workload in Virtualized Datacenters. *IEEE Access*, 9:48413–48428.

Panadero, J., Selimi, M., Calvet, L., Marquès, J. M., and Freitag, F. (2021). A two-stage Multi-Criteria Optimization method for service placement in decentralized edge micro-clouds. *Future Generation Computer Systems*, 121:90–105.

Pióro, M. and Medhi, D. (2004). *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Pozza, M., Nicholson, P. K., Lugones, D. F., Rao, A., Flinck, H., and Tarkoma, S. (2020). On Reconfiguring 5G Network Slices. *IEEE Journal on Selected Areas in Communications*, 38(7):1542–1554.

Promwongsa, N., Ebrahimzadeh, A., Glitho, R., and Crespi, N. (2022). Joint VNF Placement and Scheduling for Latency-sensitive Services. *IEEE Transactions on Network Science and Engineering*, pages 1–1. Conference Name: IEEE Transactions on Network Science and Engineering.

Qin, M., Li, M., and Othman Yahya, R. (2023). Dynamic IoT service placement based on shared parallel architecture in fog-cloud computing. *Internet of Things*, 23:100856.

Rahman, M. R. and Boutaba, R. (2013). SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization. *IEEE Transactions on Network and Service Management*, 10(2):105–118.

Ravindra K. Ahuja, Thomas L. Magnanti, J. B. O. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1 edition.

Sato, T., Kurimoto, T., Urushidani, S., and Oki, E. (2022). Virtualized Network Graph Design and Embedding Model to Minimize Provisioning Cost. *IEEE Transactions on Network and Service Management*, pages 1–1. Conference Name: IEEE Transactions on Network and Service Management.

SHI, H., Prasad, R. V., Onur, E., and Niemegeers, I. (2014). Fairness in Wireless Networks:Issues, Measures and Challenges. *IEEE Communications Surveys Tutorials*, 16(1):5–24. Conference Name: IEEE Communications Surveys Tutorials.

Skarlat, O., Nardelli, M., Schulte, S., Borkowski, M., and Leitner, P. (2017). Optimized IoT service placement in the fog. *Service Oriented Computing and Applications*, 11(4):427–443.

Skarlat, O., Schulte, S., Borkowski, M., and Leitner, P. (2016). Resource Provisioning for IoT Services in the Fog. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 32–39.

Sonkoly, B., Czentye, J., Szalay, M., Németh, B., and Toka, L. (2021). Survey on Placement Methods in the Edge and Beyond. *IEEE Communications Surveys & Tutorials*, 23(4):2590–2629. Conference Name: IEEE Communications Surveys & Tutorials.

Tran, P. N. and Timm-Giel, A. (2013). Reconfiguration of virtual network mapping considering service disruption. In *2013 IEEE International Conference on Communications (ICC)*, pages 3487–3492. ISSN: 1938-1883.

Vazirani, V. V. (2004). *Approximation algorithms*. Springer.

Wang, D., Liu, Z., Wang, X., and Lan, Y. (2019). Mobility-Aware Task Offloading and Migration Schemes in Fog Computing Networks. *IEEE Access*, 7:43356–43368.

White, D. J. (1982). *Optimality and Efficiency*. John Wiley & Sons.

Williams, H. P. (2013). Model Building in Mathematical Programming, 5th Edition | Wiley.

Xia, W., Wen, Y., Foh, C. H., Niyato, D., and Xie, H. (2015). A Survey on Software-Defined Networking. *IEEE Communications Surveys Tutorials*, 17(1):27–51.

Yang, S., Li, F., Trajanovski, S., Chen, X., Wang, Y., and Fu, X. (2021). Delay-Aware Virtual Network Function Placement and Routing in Edge Clouds. *IEEE Transactions on Mobile Computing*, 20(2):445–459.

Yang, Y., Xu, M., and Li, Q. (2018). Fast Rerouting Against Multi-Link Failures Without Topology Constraint. *IEEE/ACM Transactions on Networking*, 26(1):384–397.

Yi, S., Li, C., and Li, Q. (2015). A survey of fog computing: Concepts, applications and issues. *Proceedings of the Workshop on Mobile Big Data*, pages 37–42.

Zhang, G., Shen, F., Yang, Y., Qian, H., and Yao, W. (2018). Fair Task Offloading among Fog Nodes in Fog Computing Networks. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6.

Zhang, H., Xiao, Y., Bu, S., Niyato, D., Yu, R., and Han, Z. (2017). Computing Resource Allocation in Three-Tier IoT Fog Networks: a Joint Optimization Approach Combining Stackelberg Game and Matching. *IEEE Internet of Things Journal*, 4:1204–1215.

Zhang, Q., Liu, F., and Zeng, C. (2019). Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 2449–2457, Paris, France. IEEE.

Zhang, X., Li, Z., Lai, C., and Zhang, J. (2022). Joint Edge Server Placement and Service Placement in Mobile-Edge Computing. *IEEE Internet of Things Journal*, 9(13):11261–11274. Conference Name: IEEE Internet of Things Journal.

Zhang, Z., Wu, G., and Ren, H. (2021). Multi-attribute-based QoS-aware virtual network function placement and service chaining algorithms in smart cities. *Computers & Electrical Engineering*, 96:107465.

Zhao, L., Du, M., and Chen, L. (2018). A new multi-resource allocation mechanism: A tradeoff between fairness and efficiency in cloud computing. *China Communications*, 15(3):57–77.

Appendix A.

Linearisation of the Mixed Integer Programming Formulations

The models presented in Chapters 4-7 are Mixed Integer Programming (MIP) formulations. The linearisation of these formulations are detailed in this appendix.

A.1 Chapter 4

$$\min \xi \tag{A.1}$$

$$\text{s.t. } \xi \geq U_b - W(x) \tag{A.2}$$

$$\xi \geq F_i(x), \qquad \forall i = 1, \ldots, n \tag{A.3}$$

$$\ell_i^1 \leq CM_i, \qquad \forall i = 1, \ldots, n-2 \tag{A.4}$$

$$\ell_i^1 \geq c_{ij} x_{ij} C_j, \qquad \forall i = 1, \ldots, n-2, j = 1, \ldots, m \tag{A.5}$$

$$\ell_i^2 \leq MM_i, \qquad \forall i = 1, \ldots, n-2 \tag{A.6}$$

$$\ell_i^2 \geq c_{ij} x_{ij} M_j, \qquad \forall i = 1, \ldots, n-2, j = 1, \ldots, m \tag{A.7}$$

$$\ell_i^3 \leq RM_i, \qquad \forall i = 1, \ldots, n-2 \tag{A.8}$$

$$\ell_i^3 \geq c_{ij} x_{ij} R_j, \qquad \forall i = 1, \ldots, n-2, j = 1, \ldots, m \tag{A.9}$$

$$T(x) \leq D_k, \qquad \forall k = 1, \ldots, a \tag{A.10}$$

$$y_k \leq N(x), \qquad \forall k = 1, \ldots, a \tag{A.11}$$

$$S_k y_k \geq N(x), \qquad \forall k = 1, \ldots, a \tag{A.12}$$

$$\sum_{i=1}^{n} c_{ij} x_{ij} = 1, \qquad \forall j = 1, \ldots, m \tag{A.13}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j = 1, \ldots, m \tag{A.14}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i = 1, \ldots, n, j = 1, \ldots, m \tag{A.15}$$

$$y_k \in \{0, 1\}, \qquad \forall k = 1, \ldots, a \tag{A.16}$$

$$\xi \geq 0 \tag{A.17}$$

$$\ell_i^{\{1,2,3\}} \geq 0, \qquad \forall i = 1, \ldots, n-2 \tag{A.18}$$

ξ and $\ell_i^{\{1,2,3\}}$ are real variables. Equations A.1-A.3 correspond to the linearisation of Equation 4.3 and Equations A.4-A.9 correspond to the linearisation of Equations 4.4-4.6.

A.2 Chapter 5

$$\min \xi \tag{A.19}$$

$$\text{s.t. } \xi \geq E(x, x') \tag{A.20}$$

$$\xi \geq L(x, x') \tag{A.21}$$

$$\ell_u^1 \leq D_u, \qquad \forall i = 1, \ldots, U \tag{A.22}$$

$$\ell_u^1 \geq \sum_{i=1}^{n} \sum_{j=1}^{n} d_{iju} x_{iju} + s_u, \qquad \forall u = 1, \ldots, U \tag{A.23}$$

$$\ell_u^1 \geq \sum_{i=1}^{n} \sum_{j=1}^{n} d'_{iju} x'_{iju} + s'_u, \qquad \forall u = 1, \ldots, U \tag{A.24}$$

$$\ell_{ij}^2 \leq B_{ij}, \qquad \forall i, j = 1, \ldots, n \tag{A.25}$$

$$\ell_{ij}^2 \geq b_u x_{iju} + b'_u x'_{iju}, \qquad \forall i, j = 1, \ldots, n, u = 1, \ldots, U \tag{A.26}$$

$$\sum_{u=1}^{U} C_u y_{iu} \leq CM_i, \qquad \forall i = 1, \ldots, n \tag{A.27}$$

$$\sum_{u=1}^{U} M_u y_{iu} \leq MM_i, \qquad \forall i = 1, \ldots, n \tag{A.28}$$

$$\sum_{u=1}^{U} S_u y_{iu} \leq SM_i, \qquad \forall i = 1, \ldots, n \tag{A.29}$$

$$\sum_{j=1}^{n} x_{iju} - \sum_{j=1}^{n} x_{jiu} = V_{iu}, \qquad \forall i = 1, \ldots, n, u = 1, \ldots, U \tag{A.30}$$

$$\sum_{j=1}^{n} x'_{iju} - \sum_{j=1}^{n} x'_{jiu} = V'_{iu}, \qquad \forall i = 1, \ldots, n, u = 1, \ldots, U \tag{A.31}$$

$$x_{iju} + x'_{iju} \leq 1, \qquad \forall i, j = 1, \ldots, n, u = 1, \ldots, U \tag{A.32}$$

$$\ell_{iu}^3 \geq y_{iu}, \qquad \forall i = 1, \ldots, n, u = 1, \ldots, U \tag{A.33}$$

$$\ell_{iu}^3 \leq \sum_{j=1}^{n} x_{iju} + x_{jiu}, \qquad \forall i = 1, \ldots, n, u = 1, \ldots, U \tag{A.34}$$

$$\ell_{iu}^3 \leq \sum_{j=1}^{n} x'_{iju} + x'_{jiu}, \qquad \forall i = 1, \ldots, n, u = 1, \ldots, U \tag{A.35}$$

$$\sum_{i=1}^{n} y_{iu} = 1, \qquad \forall u = 1, \ldots, U \tag{A.36}$$

$$s_v + d_{eu} - s_u \leq T(1 - \delta_{uv} + 2 - x_{eu} - x_{ev}), \qquad \forall e \in E, u \neq v, u, v = 1, \ldots, U \tag{A.37}$$

$$s_v - s_u \leq T\delta_{uv}, \qquad \forall u \neq v, u, v = 1, \ldots, U \tag{A.38}$$

$$\delta_{uv} + \delta_{vu} = 1, \qquad \forall u < v, u, v = 1, \ldots, U \tag{A.39}$$

$$s'_v + d'_{eu} - s'_u \leq T'(1 - \delta'_{uv} + 2 - x'_{eu} - x'_{ev}), \qquad \forall e \in E, u \neq v, u, v = 1, \ldots, U \tag{A.40}$$

$$s'_v - s'_u \leq T'\delta'_{uv}, \qquad \forall u \neq v, u, v = 1, \ldots, U \tag{A.41}$$

$$\delta'_{uv} + \delta'_{vu} = 1, \qquad \forall u < v, u, v = 1, \ldots, U \tag{A.42}$$

$$\xi \geq 0 \tag{A.43}$$

$$\ell^1_u \geq 0, \qquad \forall u = 1, \ldots, U \tag{A.44}$$

$$\ell^2_{ij} \geq 0, \qquad \forall i, j = 1, \ldots, n \tag{A.45}$$

$$\ell^3_{iu} \geq 0, \qquad \forall i = 1, \ldots, n, u = 1, \ldots, U \tag{A.46}$$

ξ, ℓ^1_u, ℓ^2_{ij}, and ℓ^3_{iu} are real variables. Equations A.19-A.21 correspond to the linearisation of Equation 5.7. Equations A.22-A.24 correspond to the linearisation of Equation 5.8. Equations A.25-A.26 correspond to the linearisation of Equation 5.9. Equations A.33-A.35 correspond to the linearisation of Equation 5.16.

A.3 Chapter 6

$$\min \xi \tag{A.47}$$

$$\text{s.t. } \xi \geq CE_{ij}(x, x', y, y'), \qquad \forall i, j = 1, \ldots, n \tag{A.48}$$

$$\xi \geq FE_{ij}(x, x', y, y'), \qquad \forall i, j = 1, \ldots, n \tag{A.49}$$

$$\xi \geq CI_i(y, y'), \qquad \forall i = 1, \ldots, n \tag{A.50}$$

$$\xi \geq FI_i(y, y'), \qquad \forall i = 1, \ldots, n \tag{A.51}$$

$$\sum_{u=1}^{U} \sum_{v=1}^{V} \sum_{r=1}^{m_v} (y^{vr}_{iu} C^{vr} + y^{vr\prime}_{iu} C^{vr\prime}) \leq \gamma_{iu}, \qquad \forall i = 1, \ldots, n \tag{A.52}$$

$$\sum_{u=1}^{U} \sum_{v=1}^{\bar{V}} \sum_{r=1}^{\bar{m}_v} (\bar{y}^{vr}_{iu} \bar{C}^{vr} + y^{vr\prime}_{iu} C^{vr\prime}) \leq \gamma_{iu}, \qquad \forall i = 1, \ldots, n \tag{A.53}$$

$$\sum_{u=1}^{U} \sum_{v=1}^{V} \sum_{r=1}^{m_v} (y^{vr}_{iu} M^{vr} + y^{vr\prime}_{iu} M^{vr\prime}) \leq \mu_{iu}, \qquad \forall i = 1, \ldots, n \tag{A.54}$$

$$\sum_{u=1}^{U}\sum_{v=1}^{\bar{V}}\sum_{r=1}^{\bar{m}_v}(\bar{y}_{iu}^{vr}\bar{M}^{vr}+y_{iu}^{vr\prime}M^{vr\prime})\leq\mu_{iu},\qquad\forall i=1,\ldots,n \tag{A.55}$$

$$\sum_{u=1}^{U}\sum_{v=1}^{V}\sum_{r,s=1}^{m_v}(x_{iju}^{vrs}b^{vrs}+x_{iju}^{vrs\prime}b^{vrs\prime})\leq w_{iju},\qquad\forall i,j=1,\ldots,n \tag{A.56}$$

$$\sum_{u=1}^{U}\sum_{v=1}^{\bar{V}}\sum_{r,s=1}^{\bar{m}_v}(\bar{x}_{iju}^{vrs}\bar{b}^{vrs}+x_{iju}^{vrs\prime}b^{vrs\prime})\leq w_{iju},\qquad\forall i,j=1,\ldots,n \tag{A.57}$$

$$\sum_{u=1}^{U}\sum_{i,j=1}^{n}\sum_{r,s=1}^{m_v}\left(\frac{MS^v}{w_{iju}}+PD_{ij}\right)x_{iju}^{vrs}\leq l^v,\qquad\forall v=1,\ldots,V \tag{A.58}$$

$$\sum_{u=1}^{U}\sum_{i,j=1}^{n}\sum_{r,s=1}^{\bar{m}_v}\left(\frac{\overline{MS}^v}{w_{iju}}+PD_{ij}\right)\bar{x}_{iju}^{vrs}\leq\bar{l}^v,\qquad\forall v=1,\ldots,\bar{V} \tag{A.59}$$

$$\sum_{u=1}^{U}\sum_{i,j=1}^{n}\sum_{r,s=1}^{m_v{}'}\left(\frac{MS^{v\prime}}{w_{iju}}+PD_{ij}\right)x_{iju}^{vrs\prime}\leq l^{v\prime},\qquad\forall v=1,\ldots,V' \tag{A.60}$$

$$\sum_{j=1}^{n}\sum_{u=1}^{U}(x_{iju}^{vrs}-x_{jiu}^{vrs})=\sum_{u=1}^{U}(y_{iu}^{vr}-y_{iu}^{vs}),\qquad\forall i=1,\ldots,n,\ v=1,\ldots,V,\ r,s=1,\ldots,m_v,r<s \tag{A.61}$$

$$\sum_{j=1}^{n}\sum_{u=1}^{U}(\bar{x}_{iju}^{vrs}-\bar{x}_{jiu}^{vrs})=\sum_{u=1}^{U}(\bar{y}_{iu}^{vr}-\bar{y}_{iu}^{vs}),\qquad\forall i=1,\ldots,n,\ v=1,\ldots,\bar{V},\ r,s=1,\ldots,\bar{m}_v,r<s \tag{A.62}$$

$$\sum_{j=1}^{n}\sum_{u=1}^{U}(x_{iju}^{vrs\prime}-x_{jiu}^{vrs\prime})=\sum_{u=1}^{U}(y_{iu}^{vr\prime}-y_{iu}^{vs\prime}),\qquad\forall i=1,\ldots,n,\ v=1,\ldots,V',\ r,s=1,\ldots,m_v{}',r<s \tag{A.63}$$

$$\sum_{i=1}^{n}\sum_{u=1}^{U}y_{iu}^{vr}=1,\qquad\forall v=1,\ldots,V,\ r=1,\ldots,m_v \tag{A.64}$$

$$\sum_{i=1}^{n}\sum_{u=1}^{U}\bar{y}_{iu}^{vr}=1,\qquad\forall v=1,\ldots,\bar{V},\ r=1,\ldots,\bar{m}_v \tag{A.65}$$

$$\sum_{i=1}^{n}\sum_{u=1}^{U} y_{iu}^{vr\prime} = 1, \qquad \forall v = 1,\ldots,V', \; r = 1,\ldots,m_v{}' \tag{A.66}$$

$$\xi \geq 0 \tag{A.67}$$

ξ is a real variable. Equations A.47-A.51 correspond to the linearisation of Equation 6.10.

A.4 Chapter 7

Virtual Network Embedding

$$\min \xi \tag{A.68}$$

$$\text{s.t. } \xi \geq FE_{ij}(x,y), \qquad \forall i,j = 1,\ldots,n \tag{A.69}$$

$$\xi \geq U_b - T_{ij}(x), \qquad \forall i,j = 1,\ldots,n \tag{A.70}$$

$$\sum_{v=1}^{V}\sum_{r=1}^{m_v} y_i^{vr} C^{vr} \leq CM_i, \qquad \forall i = 1,\ldots,n \tag{A.71}$$

$$\sum_{v=1}^{V}\sum_{r=1}^{m_v} y_i^{vr} M^{vr} \leq MM_i, \qquad \forall i = 1,\ldots,n \tag{A.72}$$

$$\sum_{v=1}^{V}\sum_{r=1}^{m_v} y_i^{vr} S^{vr} \leq SM_i, \qquad \forall i = 1,\ldots,n \tag{A.73}$$

$$\sum_{v=1}^{V}\sum_{r=1}^{m_v}\sum_{s=1}^{m_v} x_{ij}^{vrs} b^{vrs} \leq \bar{b}_{ij}, \qquad \forall i,j = 1,\ldots,n \tag{A.74}$$

$$\sum_{j=1}^{n} x_{ij}^{vrs} - \sum_{j=1}^{n} x_{ji}^{vrs} = y_i^{vr} - y_i^{vs}, \qquad \forall i = 1,\ldots,n, v = 1,\ldots,V, \; r,s = 1,\ldots,m_v, r < s \tag{A.75}$$

$$\sum_{i=1}^{n} y_i^{vr} = 1, \qquad \forall v = 1,\ldots,V, r = 1,\ldots,m_v \tag{A.76}$$

$$\sum_{r=1}^{m_v} y_i^{vr} \leq 1, \qquad \forall i = 1,\ldots,n, v = 1,\ldots,V \tag{A.77}$$

$$\xi \geq 0 \tag{A.78}$$

ξ is a real variable. Equations A.68-A.70 correspond to the linearisation of Equation 7.3.

Service Placement

$$\min \xi \tag{A.79}$$

$$\text{s.t. } \xi \geq E(w,z) \tag{A.80}$$

$$\xi \geq L(w,z) \tag{A.81}$$

$$\ell_i^1 \geq w_{ikp} C_{kp}, \qquad \forall i = 1,\ldots,n-1,$$

$$k = 1, \ldots, \mathsf{K}, p = 1, \ldots, q_k \tag{A.82}$$

$$\ell_i^1 \leq CM_i \qquad \forall i = 1, \ldots, n-1 \tag{A.83}$$

$$\ell_i^2 \geq w_{ikp} M_{kp}, \qquad \forall i = 1, \ldots, n-1, \quad k = 1, \ldots, \mathsf{K}, p = 1, \ldots, q_k \tag{A.84}$$

$$\ell_i^2 \leq MM_i \qquad \forall i = 1, \ldots, n-1 \tag{A.85}$$

$$\ell_i^3 \geq w_{ikp} S_{kp}, \qquad \forall i = 1, \ldots, n-1, \quad k = 1, \ldots, \mathsf{K}, p = 1, \ldots, q_k \tag{A.86}$$

$$\ell_i^3 \leq SM_i \qquad \forall i = 1, \ldots, n-1 \tag{A.87}$$

$$s_g + t_{ih} - s_h \leq T(1 - \delta_{hg} + 2 - y_{ih} - y_{ig}), \qquad \forall i = 1, \ldots, n, h \neq g, \quad h, g = 1, \ldots, H \tag{A.88}$$

$$s_g - s_h \leq T\delta_{hg}, \qquad \forall h \neq g, h, g = 1, \ldots, H \tag{A.89}$$

$$\delta_{hg} + \delta_{gh} = 1, \qquad \forall h <_{lex} g, h, g = 1, \ldots, H \tag{A.90}$$

$$TD(w, z) \leq D_k, \qquad \forall k = 1, \ldots, K \tag{A.91}$$

$$\sum_{k=1}^{K} b_k z_{ijk} \leq \bar{b}_{ij}, \qquad \forall i, j = 1, \ldots, n \tag{A.92}$$

$$\sum_{j=1}^{n} z_{ijk} - \sum_{j=1}^{n} z_{jik} = U_{ik}, \qquad \forall i = 1, \ldots, n, k = 1, \ldots, K \tag{A.93}$$

$$w_{ikp} \leq \sum_{j=1}^{n} (z_{ijk} + z_{jik}), \qquad \forall i = 1, \ldots, n, k = 1, \ldots, K, \quad p = 1, \ldots, q_k \tag{A.94}$$

$$\sum_{i=1}^{n} w_{ikp} = 1, \qquad \forall k = 1, \ldots, K, p = 1, \ldots, q_k \tag{A.95}$$

$$\xi \geq 0 \tag{A.96}$$

$$\ell_i^{\{1,2,3\}} \geq 0, \qquad \forall i = 1, \ldots, n-1 \tag{A.97}$$

ξ and $\ell_i^{\{1,2,3\}}$ are real variables. Equations A.79-A.81 correspond to the linearisation of Equation 7.18 and Equations A.82-A.87 correspond to the linearisation of Equations 7.19-7.21.