

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Resource Management in the Computing Continuum using Machine Learning

Diploma Thesis

Ioannis Roumpos

Supervisor: Parisis Flegkas

July 2024



UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Resource Management in the Computing Continuum using Machine Learning

Diploma Thesis

Ioannis Roumpos

Supervisor: Parisis Flegkas

July 2024



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διαχείριση Πόρων στο Υπολογιστικό Συνεχές με Χρήση Μηχανικής Μάθησης

Διπλωματική Εργασία

Ιωάννης Ρούμπος

Επιβλέπων: Παρίσης Φλέγκας

Ιούλιος 2024

Approved by the Examination Committee:

Supervisor **Parisis Flegkas**
Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Athanasios Korakis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Antonios Argyriou**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

First and foremost, i would like to express my deepest gratitude to my supervisor Ass. Professor Paris Flegkas for proposing me such a state-of-the-art topic and being able to work on a research based thesis. It was his careful guidance and proficiency that enabled me to successfully carry out the challenging tasks incorporated in completing this thesis.

A special thank you goes to my fellow graduate students and friends, for their support throughout this 5-year journey. Our shared experiences and mutual support have made this journey both enjoyable and memorable.

Also,i am profoundly grateful to my family for their unwavering support and understanding. To my parents and my brother, a big thank you for instilling in me the value of education and hard work.

Last but not least, I would like to express deep appreciation to Professor Athanasios Korakis and Associate Professor Antonios Argyriou for serving on the thesis evaluation committee.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Ioannis Roumpos

Diploma Thesis

Resource Management in the Computing Continuum using Machine Learning

Ioannis Roumpos

Abstract

Telecommunications are undergoing a transformation due to the introduction of 5G and the upcoming 6G technologies, which promise previously unheard-of speeds, extremely low latency, and widespread access. The resource management of edge, fog, and cloud infrastructures is facing new difficulties as a result of these improvements. In order to optimize performance, cost-efficiency, and energy consumption, this thesis investigates how machine learning (ML) might be used to improve resource management across this continuum, specifically in the context of 5G and 6G networks.

In this work, a thorough survey of the current resource management needs is made, across the computing continuum, in order to understand the current solutions that can ensure better QoS to the users and try to find a gap in them. This analysis inspired us to develop a Q-learning algorithm that can provide better solution than an iterative solution proposed in the literature. The experimental analysis, that follows with the comparison of the two solutions, show that the proposed algorithm performs better in medium size networks. To evaluate the performance of this study we take into consideration network parameters such as overall traffic and delay.

Keywords:

computing continuum, resource management, machine learning, Q-learning

Διπλωματική Εργασία

Διαχείριση Πόρων στο Υπολογιστικό Συνεχές με Χρήση Μηχανικής Μάθησης

Ιωάννης Ρούμπος

Περίληψη

Οι τηλεπικοινωνίες υφίστανται μια μεταμόρφωση λόγω της εισαγωγής των τεχνολογιών 5G και των επερχόμενων 6G, οι οποίες υπόσχονται πρωτοφανείς ταχύτητες, εξαιρετικά χαμηλή καθυστέρηση και ευρεία πρόσβαση. Η διαχείριση των πόρων των υποδομών edge, fog και cloud αντιμετωπίζει νέες δυσκολίες ως αποτέλεσμα αυτών των βελτιώσεων. Αυτή η διατριβή διερευνά πώς η μηχανική μάθηση (ML) μπορεί να χρησιμοποιηθεί για τη βελτίωση της διαχείρισης πόρων σε όλο αυτό το συνεχές, ειδικά στο πλαίσιο των δικτύων 5G και 6G, με στόχο τη βελτιστοποίηση της απόδοσης, της αποδοτικότητας κόστους και της κατανάλωσης ενέργειας.

Σε αυτή τη δουλειά, πραγματοποιείται μια διεξοδική επισκόπηση των τρεχουσών αναγκών διαχείρισης πόρων κατά μήκος του υπολογιστικού συνεχούς, προκειμένου να κατανοηθούν οι τρέχουσες λύσεις που μπορούν να εξασφαλίσουν καλύτερη ποιότητα υπηρεσίας (QoS) στους χρήστες και να προσπαθήσουμε να βρούμε ένα κενό σε αυτές. Αυτή η ανάλυση μας ενέπνευσε να αναπτύξουμε έναν αλγόριθμο Q-learning που μπορεί να παρέχει καλύτερη λύση από μια επαναληπτική λύση που προτείνεται στη βιβλιογραφία. Η πειραματική ανάλυση, που ακολουθεί με τη σύγκριση των δύο λύσεων, δείχνει ότι ο προτεινόμενος αλγόριθμος αποδίδει καλύτερα σε μεσαίου μεγέθους δίκτυα. Για την αξιολόγηση της απόδοσης αυτής της μελέτης λαμβάνουμε υπόψη παραμέτρους δικτύου όπως η συνολική κυκλοφορία και η καθυστέρηση.

Λέξεις-κλειδιά:

υπολογιστικό συνεχές, διαχείριση πόρων, Μηχανική Μάθηση, Q-learning.

Table of contents

Acknowledgements **ix**

Abstract **xii**

Περίληψη **xiii**

Table of contents **xv**

List of figures **xix**

List of tables **xxi**

Abbreviations **xxiii**

1 Introduction **1**

1.1 Motivation 1

1.2 Thesis subject 2

1.2.1 Contribution of the Thesis 2

1.3 Outline 2

2 Background **3**

2.1 Introduction 3

2.2 Resource Management in Edge Computing 4

2.3 Resource Management in Cloud Computing 5

2.4 Network Resource Management 7

2.5 Related Work on Resource Management 9

3 Machine Learning **11**

3.1 Introduction . . . 11

3.2 Reinforcement Learning . . . 12

3.2.1 Policy . . . 13

3.2.2 Reward Function . . . 14

3.2.3 Value function . . . 14

3.2.4 Q-Learning . . . 14

3.3 Related Work on ML-Based Service Placement . . . 15

4 Proposed Model **17**

4.1 Introduction . . . 17

4.2 System Model . . . 17

4.3 Network Model . . . 18

4.4 Problem Formulation . . . 19

4.5 Algorithm . . . 20

5 Experimental Analysis **23**

5.1 Introduction . . . 23

5.2 Tools and Environment . . . 23

5.2.1 Jupyter notebook . . . 23

5.2.2 NumPy . . . 24

5.2.3 Matplotlib . . . 24

5.2.4 NetworkX . . . 25

5.3 Configuration of the Environment . . . 25

5.3.1 Topology . . . 25

5.3.2 Algorithm . . . 26

5.4 Experiments and Results . . . 26

6 Conclusions **33**

6.1 Summary and Conclusions . . . 33

6.2 Future Work . . . 34

Bibliography **35**

APPENDICES **39**

A Figures of Experimental Analysis **41**

B Implementation Code **43**

B.1 Topology generator 43

B.2 Algorithms 44

B.2.1 Proposed solution 44

B.2.2 Iterative approach 46

List of figures

3.1 Machine Learning Fields[1] 12
3.2 Reinforcement Learning basic scheme [2] 13

4.1 Edge-Cloud Environment 18
4.2 Sample graph with 20 nodes. 19

5.1 Overall traffic for different number of locations 27
5.2 Average hop count for different number of locations 28
5.3 Average hop count for k = 19 and n = 60 28
5.4 Average hop count for k = 23 and n = 60 28
5.5 Average hop count for k = 27 and n = 60 29
5.6 Overall traffic network for $n = 40$ 29
5.7 Average Hop Count for $n = 40$ 30
5.8 Overall traffic network for $n = 90$ 30
5.9 Average Hop Count for $n = 90$ 31

List of tables

5.1 Q-learning Parameters 26

Abbreviations

5G	5th Generation
6G	6th Generation
NFV	Networks Functions Virtualization
ML	Machine Learning
AI	Artificial Intelligence
MEC	Mobile Edge Computing
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
VM	Virtual Machine
OS	Operating System
SLA	Service Level Agreement
CSP	Cloud Service Provider
PM	Physical Machine
IoT	Internet of Things
QoS	Quality of Service
QoE	Quality of Experience
SDN	Software Defined Networks
CU	Centralized Unit
DU	Distributed Unit
UE	User Equipment
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
CDN	Content Delivery Networks
NOC	Networks Operations Centre

w.l.o.g	without loss of generality
e.g	exempli gratia
MDP	Markov Decision Process

Chapter 1

Introduction

1.1 Motivation

The rapid development of 5G networks and the work towards deploying 6G networks indicate a new era of telecommunications characterized by low latency and the ability to support high number of interconnected devices. This development has affected the growth of applications that range from autonomous vehicles, smart cities and virtual reality experiences. Thus, to handle the different requirements and the heterogeneity of the platforms, computing continuum, an integrated infrastructure that covers the edge,cloud and fog environments, plays a crucial role in managing computational resources.

One of the challenges on managing the continuum resources is the placement problem. The placement problem involves determining the optimal deployment of computational tasks and services across the various layers of the continuum. In the edge, decentralized resources are located closest to end-user, such as edge servers and gateways while in the cloud resources are centralized in data centers providing large-scale computational power. Intermediate resources withing network infrastructure including fog servers and network functions virtualization (NFV) elements provide the needed balance from cloud to edge regarding latency and resource availability.

All the above indicate the need for different approaches while tackling with a problem such as service placement. One such approach lies on the use of Machine Learning (ML) and Artificial Intelligence (AI), since it is a potential part of integration in the 5G and beyond networks and it can bring new breakthroughs regarding resource management and overall network performance.

1.2 Thesis subject

The goal of this thesis is to address the problem of service placement in different network locations using a novel algorithm based on ML/AI. To do that, network topologies are created with a central cloud node and many edge nodes where clients are connected. The overall network traffic and the delay are measured and compared with a traditional placement algorithm. In this thesis, we propose a RL-based algorithm using Q-learning for determining the best locations of services across the continuum using different network configurations.

1.2.1 Contribution of the Thesis

The contribution of the thesis is summarized as follows:

1. An overview of resource management in the computing continuum is provided.
2. Problem formulation and implementation of a Q-learning algorithm is proposed for the service placement problem with the aim to optimize overall network traffic and delay.
3. Comparison with an iterative solution proposed in the literature and analysis of the results based on certain metrics.

1.3 Outline

The rest of this Thesis is organized as follows: Chapter 2 is a thorough analysis of the current resource management schemes in the computing continuum including the aspects of the network resources. Chapter 3 gives an overview of Machine Learning (ML) and Reinforcement Learning (RL) and its components in more detail. In both of this chapters, related work to the placement problem is also presented. In Chapter 4 the problem and its formulation is presented as well as the implementation of the proposed algorithm. In Chapter 5 the experimental evaluation is performed between the proposed algorithm and an iterative approach. Lastly, Chapter 6 summarizes the key points of the proposed work and ideas for its extension are provided.

Chapter 2

Background

2.1 Introduction

In computing settings, resource management refers to the economical distribution and use of computing resources, including memory, CPU, storage, and network bandwidth, in order to guarantee best practices and economy. Since it directly affects the scalability, dependability, and performance of applications and services, effective resource management is essential. Organizations may save operating costs, improve user experience, and make greater use of their infrastructure by managing resources well. However, different computer paradigms—such as edge computing, cloud computing, and network resources—have different resource management techniques.

The emergence of the compute continuum further complicates, yet empowers, resource management. This distributed landscape, where processing power can reside anywhere from centralized cloud data centers to local edge devices like wearables, necessitates a holistic approach. Resource allocation needs to be dynamic and intelligent, seamlessly shifting workloads across the continuum based on factors like latency requirements, processing power needs, and real-time network conditions.

5G and future 6G networks, with their focus on ultra-reliable, low-latency communication (uRLLC) and high bandwidth, will require even more sophisticated resource management strategies. The ability to manage resources not just within individual paradigms (cloud, edge) but also across the entire continuum becomes critical. Network slicing, a key 5G/6G technology, allows for the creation of virtual networks with customized resource allocation, catering to specific application demands. Additionally, advancements in machine learning

and artificial intelligence can automate and optimize resource management decisions in real-time, ensuring efficient utilization across the continuum.

Because every paradigm is different and calls for different approaches to deal with the environment's demands and limitations, resource management is a complicated but crucial component of contemporary computing.

2.2 Resource Management in Edge Computing

Edge computing refers to an architecture or generally a computing model that utilises the resources located at the edge of the network or near the end-user[3]. The resources at the edge are typically more limited due to the physical constraints of the devices, their workload changes over the time and the edge devices are described by their heterogeneity both of the hardware architecture and the produced data[4]. In this context, the resources could be in the form of Internet components such as routes, gateways or data centres in a version that can be stored in a private/public structure.

Thus, to make edge computing a viable reality, efficient management of resources at the edge is crucial. Managing these edge resources allows devices to save energy, speed up computations, and access resources they lack. Keeping data close to its source also ensures better control, especially concerning privacy. Edge computing's proximity to users enables better service quality through tailored profiling without compromising privacy or overwhelming user numbers.

A major paradigm shift in edge computing, is mobile edge computing (MEC) which brings fast, dependable, secure distributed services closer to users and data sites by combining network control, processing, and storage. Because user applications have different performance requirements and surroundings are dynamic, resource management in MEC is a major area of research and its application on 5G networks and the services that would benefit from it ,like augmented reality and virtual reality applications, highlight the importance of this shift. In terms of methods of optimization both AI-ML-based and conventional optimization-based techniques can be used to find solutions for MEC resource allocation. In resource-constrained edge situations, balancing training latency and device energy consumption presents challenges[5].

Resource management at the edge can differ based on the resource type, the objective, the

resource location and how the resources are used. The objective answers to the goal of doing resource management and it can be further broken down on resource discovery, resource estimation, resource sharing, resource optimization and resource allocation. In resource allocation, the main aim is to find the scheme that allows to the system each time to perform optimally assigning the available resources to the users. This procedure can be done with various methods like finding the best placement or the right scheduling policy.

One of the most surveyed area is about the placement of a task. The placement method is used to identify the resources that will best meet the needed workload. Assigning the right edge resources to the incoming compute tasks is the primary goal. The resources may vary accordingly with the needs of the task taking into account both system (CPU, RAM, Disk) and network (latency, bandwidth) parameters.

To sum up, effective resource management in edge computing is essential to realizing this paradigm's full potential. Edge computing can improve privacy control, save energy, and deliver higher service quality by making the best use of scarce resources. As we proceed, overcoming the inherent challenges of edge resource management will mostly depend on the integration of AI-driven methodologies and advanced optimization techniques.This thorough comprehension of edge computing resource management paves the way for investigating related issues and solutions in cloud systems. Next, we explore cloud computing resource management, where scalability, multi-tenancy, and cost-effectiveness offer special opportunities and problems.

2.3 Resource Management in Cloud Computing

Cloud computing is a technology that uses the internet to provide computing resources as services.This groundbreaking advancement enables the flexible and cost-effective sharing of resources among cloud customers as per their needs. It offers a range of online computing services at competitive rates, facilitating efficient data management, processing, and storage for users. The cloud eliminates the necessity for customers to install software on their devices; as long as an internet connection is available, users can access their data globally from any computer.[6]

The architecture of cloud computing can be summarised in four distinct technology models and three service models[7]. The technology models are divided in the private cloud,

where big organizations e.g (Google) manage and make available the resources itself, the public cloud in which the user pay a provider e.g (Amazon) for the use of the services, the hybrid cloud a combination of the previous models and the community cloud where users take advantage of a certain pool of several providers. As stated in [7], the cloud application delivers *Software as a Service (SaaS)* giving the opportunity to the users to run the application they want in a central unit and not in their system. Platform services *Platform as a Service (PaaS)* provide a computing platform using the cloud infrastructure to develop and run applications. Lastly, *Infrastructure as a Service (IaaS)* offers to the users the suitable virtualized infrastracture with the appropriate system and network resources[8].

Although the fast acceptance of cloud computing and cloud technology generally, some challenges arise. One of them is multi-tenacy which refers to the usage of resources by a large pool of users (tenants). Resource contention may arise which will affect the performance and the services provided. Also, providing the right isolation between resources is a challenge in order to ensure compliance for the users and the shared nature of resources can lead to performance variability, where the actions of one tenant can impact the performance of others. Another challenge is scalability that is the ability of handling a increasingly amount of resources either vertically(more power to the existing machines) or horizontally (adding more machines). Physical limits of the machines may be a factor making this possibility not feasible and in the same time it is observed a waste of resources due to the lack of provisioning.

The basis of tackling with these challenges and providing efficient resource utilization is virtualization and containerization. In the case of virtualization, a hypervisor can create and run Virtual Machines (VMs) that can facilitate many tasks. VMs are heavy software packages that provide complete emulation of low level hardware devices like CPU, Disk and Networking devices. Therefore, it is not needed by the application using a VM is a waste of resources and since it emulates a Operating System (OS) the startup time is a significant drawback. On the other hand, at the application layer, containers are an abstraction that bundles dependencies and code. On a single machine, several containers can operate independently as separate processes in user space, sharing the OS kernel with one another. Compared to virtual machines (VMs), containers are smaller, more capable, and require fewer operating systems and virtual machines[9]. It is clear that managing VMs and containers in regards with the available resources and the tasks the users want to complete is a necessity. Resource management is considered to be one of the most essential aspects of cloud computing and is

essential in order to provide higher performance utilization and efficient use of the underlying hardware. One of the most common techniques for resource management is load balancing in which specific algorithms try to distribute the workload in multiple servers. Beside that another concern is how to assigning the workload taking into account the SLAs in IaaS and PaaS cloud environments. The application scaling and provision of resources is also another way to address the needs of the users. Managing pro-actively the available resources and taking the decision where to scale up or down is important since the operation cost will be decreased and it will be more flexible.

Scheduling the local virtualized resources is a key aspect for resource management and has been highly researched. These kinds of solutions aim to offer a suitable compromise between automated resource distribution across competing virtual machines and static, user-specified resource allocations. We should not dismiss the importance of scheduling the network access which eventually keep consistent bandwidth and latency in a data center networks despite changing traffic patterns.

Lastly, but as important as the other methods is the placement of both services,VMs and containers. Eventually, a proper virtual machine placement algorithm links these virtual machines to the cloud data center's Physical Machines (PMs).An effective virtual machine (VM) placement algorithm arranges the VMs on PMs in a way that best serves the needs of the cloud service provider (CSP) and the cloud user[10]. Not only VMs but containers that can presented as services can be placed using different strategies with Kubernetes being the most popular one[11]. Thus, it it obvious that together, these techniques enable cloud environments to deliver scalable, reliable, and cost-effective solutions for a wide range of applications.

2.4 Network Resource Management

These days, physical objects, cars, home appliances, and other devices with electronics, software, sensors, and actuators incorporated in them are connected via the Internet of Things (IoT). In order to facilitate the data traffic generated and ensure the best Quality of Service (QoS) to users, 5G and beyond networks is looking like a viable answer to the above demands. Resources for networking, processing, and memory have grown scarce, making efficient resource management and sharing essential. Softwarization and virtualization-based technologies, such SDN, NFV, have made it possible to manage resources efficiently in 5G

networks. However, resource management will be more difficult in 6G due to the constantly expanding subscriber base and the connectivity needs of increasingly data-hungry and computationally sophisticated apps[12].

One of the key aspects of network resource management is traffic engineering. Understanding the instantaneous traffic demands is necessary for network resource provisioning, which then makes use of sophisticated resource admission control. Though, this provisioning may be quite a challenge since estimating demands based on the history is uncertain and routing protocols do not handle well dynamic situations[13]. In addition, how bandwidth is allocated to applications or users is an important challenge. Ensuring that sufficient bandwidth is available for different applications and services to prevent congestion and maintain performance concerns a several fields from Teleco to cloud computing. Also, edge and cloud computing are closely connected fields regarding network resource management due to the fact that the emerging architectures for the new age networks combine these approaches. Thus, challenges described in the previous sections affected the overall network resources and consequently the Quality of Experience (QoE) for the users.

To deal with these challenges two are the main technologies (SDN, NFV) developed and are currently in the first line of research in the latest generation of cellular networks. SDN, is an approach that decouples the control plane from the data plane making easier the dynamic configuration of the network resources. This approach provides with better flexibility since it enables dynamic adjustments to meet changing demands and facilitates automation of network management tasks, reducing the need for manual intervention. Many resource management techniques regarding SDN are proposed, one of them is finding the optimal placement for the controller which aligns closely with our own problem statement.

In the case of Network Function Virtualization (NFV) functions that are commonly run on hardware such as routers or firewalls are virtualized and are implemented typically in VMs or containers[14]. Since the need for specialized hardware is decreased the operational expenses and cost is decreased as well making this solutions cost-friendly. Scalability is also another benefit from this technology because users needs are changing from task to task and adapting quickly and providing new services is vital. As mentioned and in the previous section, the fact that these functions run on VMs or containers provide isolation to the users and the tasks securing the exchange of data and keeping the wanted privacy. It should be mentioned that, the two technologies mentioned are highly complementary the one to the other. For exam-

ple, using a SDN controller we can configure the NFV infrastructures (routers, gateways) as mentioned in [14] or NFV may also use service chaining, a method known as SDN flow programming, to direct traffic through one or more virtualized network functions[15].

Ensuring high availability, preserving optimal performance, and offering a safe and dependable cloud environment all depend on efficient network resource management. Performance is optimized via traffic shaping and QoS regulations, and flexibility and scalability are provided by techniques like SDN and NFV. Efficient traffic distribution is ensured by load balancing, while ongoing network monitoring guarantees proactive problem resolution and security. When combined, these tactics allow cloud environments to satisfy the needs of contemporary users and applications by providing reliable and effective network services.

2.5 Related Work on Resource Management

The resource management problem encompasses the broad challenge of efficiently and effectively allocating various resources, such as computational power, storage, and network bandwidth, to meet the demands of different tasks and applications. Within this broader context, the placement problem emerges as a critical subcategory, specifically focusing on the optimal assignment or location of these resources. The placement problem aims to determine the best possible locations for services to achieve specific objectives like minimizing latency, balancing load, or reducing energy consumption. Addressing the placement problem requires sophisticated algorithms and models that account for dynamic workloads, resource availability, and other constraints, thereby contributing to the overall effectiveness of resource management. As depicted, the placement problem can be found in different applications varying from IoT, Big data analytics and Telecommunications. Also, since the rapid development of 5G networks and beyond and the different proposed architectures of the Radio Access Networks (RAN) and MEC the need for different approaches in the placement placement that covers the different service types in one scheme is also critical.

The following presented related work will be focused in a scheme with multi-edge nodes and a central cloud since it resembles the architecture of our problem too. Some of the works, similar to the problem we want to set are discussed in the topic of Content-Centric Networks [16][17] where the data objects, in our case services, cooperate to help users have access to contents. This work, although it provides a novel and efficient placement scheme it does

not address network specific parameters to provide better QoS. In the rapid evolving area of VNFs, many researches are proposing new ideas to tackle with the placement problem. Notably, in [18] the authors address the placement and routing problem for edge-cloud with focus on minimizing the latency but it does not show promising results when increment of parameters and the dynamism of the environment occurs. Also, as depicted in [19] an issue when providing optimal placement solutions is the increased complexity and running time which is a critical trade-off with parameters optimizations such overall traffic and link bottlenecks. Another trade-off is presented in [18], where an online efficient fair algorithm achieves near optimal solution but the authors suggest that it should not be considered for edge networks with limited resources making the application of it more restricted.

In the area of distributed placement [20],[21] give an solution based on mathematical optimization that provide better overall network traffic cost, lower the energy consumption and overcomes scalability problems respectively. A different area that tackles the problem of placement is in Software Defined Networks (SDN). In this context, an optimal placement of the controller is mostly the case to meet the needs for better QoS. For instance, Heller et al. (2012)[22] introduced an early exploration of controller placement, highlighting the trade-off between latency and reliability. Later, other studies expanded on this by incorporating dynamic and adaptive strategies to cope with network changes and failures. Lastly, in the latest proposed network architectures like 5G RANs the placement of virtualized units are important and as shown in [23] it has a great impact regarding the latency. The strategic distribution and optimization of these virtualized units, such as Centralized Units (CUs) and Distributed Units (DUs), within the network infrastructure are essential for achieving low-latency communication, which is a key requirement for applications like autonomous driving, remote surgery, and augmented reality. Proper placement ensures efficient processing, minimizes data transfer delays, and enhances overall network performance by reducing the distance data needs to travel between the user equipment (UE) and the core network

Chapter 3

Machine Learning

3.1 Introduction

In general, machine learning refers to computer techniques that leverage experience to enhance performance or generate precise predictions. In this context, ”experience” refers to the learner’s prior knowledge, which is usually represented by gathered and analyzed data. In machine learning, there are four commonly used learning models usually categorized as supervised learning, semi-supervised learning, unsupervised learning and reinforcement learning. [24][25]

- In the **Supervised Learning (SL)** paradigm of machine learning, a model is trained using desirable output value (also called human-labeled supervisory signal) and input objects (e.g., a vector of predictor variables). After processing the training set of data, a function mapping fresh data to anticipated output values is created. In an ideal situation, the algorithm will be able to accurately calculate output values for cases that are not visible.

- Algorithms for **Unsupervised Learning** discover structures in unlabeled, uncategorized data. Unsupervised learning algorithms find patterns in the data and make decisions based on whether or not these patterns are present in each new piece of data, rather than reacting to feedback. Density estimation, dimensionality reduction, and clustering are three important uses of unsupervised machine learning.

- Intermediary between unsupervised and supervised learning, which involve labeled training data, is **Semi-supervised Learning**. Although there are some training exam-

ples without training labels, a large body of research in machine learning has shown that a modest quantity of labeled data combined with unlabeled data can significantly increase learning accuracy.

- In **Reinforcement Learning (RL)** there is an incorporation which combines both training and testing stages. In order to gather data, the student actively engages with the surroundings, sometimes changing them in the process, and is rewarded right away for each activity. The learner's goal is to maximize his reward through a series of activities and interactions with the surroundings.[26]

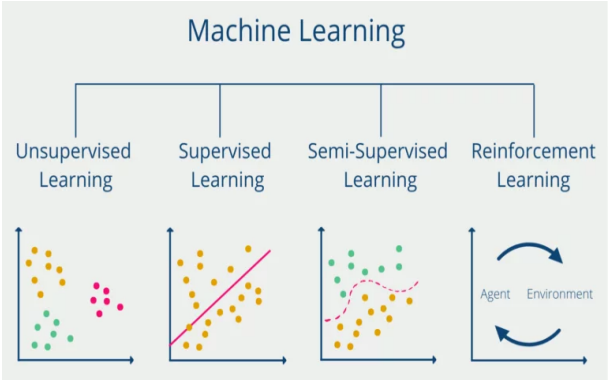


Figure 3.1: Machine Learning Fields[1]

In this thesis, the focus will be on RL and especially a well-known RL algorithm, Q-learning, will be used. In the next sectios, a thorough review of the algorithm will follow and the basic components,that define a RL paradigm, will be analyzed. In that way, there will a better understanding of the benefits from using this technique and the specific algorithm.

3.2 Reinforcement Learning

Learning how to connect situations to behaviors in order to maximize a numerical reward signal is known as **Reinforcement Learning**. Instead of being directed, the learner must experiment to determine which courses of action will result in the greatest rewards. In this context, the learner is called *agent* and the situations in which the agent takes decisions in called *environment*. More than the agent and the environment in a

reinforcement system the following sub elements can be identified: *policy*, *reward* and *value function*.

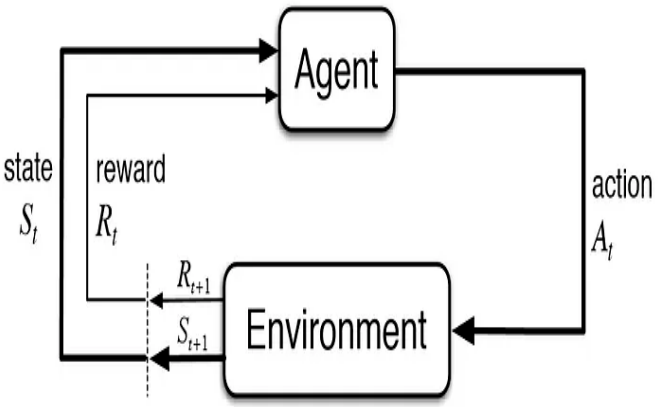


Figure 3.2: Reinforcement Learning basic scheme [2]

As we can see from the figure above, at each time step t = 0, 1, 2.. the agent is in a state $s_t \in S$ where it chooses an action $a_t \in A(t)$ based on the reward R_t and the current state s_t. Then, the environment will reply with a new pair of reward R_{t+1} and state s_{t+1} and this task will continue. In each episode, the goal is to maximize the cumulative reward based on a policy π.

3.2.1 Policy

A key idea in RL is a policy, which establishes the agent's behavior at each given moment. It is a tactic, or a mapping, that determines what steps the agent ought to take in each condition of the environment. Formally, a policy can be either stochastic—in which case actions are selected based on a probability distribution—or deterministic—in which case a certain action is selected for every state. Learning an ideal strategy that maximizes the predicted cumulative reward over time is the aim of reinforcement learning. This calls for striking a balance between exploitation—selecting courses of action that are known to provide large rewards—and exploration—trying new things to see how they affect things.[26]

3.2.2 Reward Function

In RL, reward function is the way to address to the agent how the actions lead to the desired result which is, as mentioned above, to maximize the overall cumulative reward. This function maps each state or state-action pair to a numerical reward, which indicates the immediate benefit or cost of being in that state or taking that action. It is important to take into account that a well defined function will lead to an improved learning process and decide better which action is more efficient than another. An improperly defined reward function can lead to unintended or suboptimal behaviors, making it a delicate and essential aspect of RL system design.

3.2.3 Value function

Value function techniques maintain a set of estimates of expected discounted for some policy (typically either the ”current” [on-policy] or the optimal [off-policy] one), and then seek to discover a policy that maximizes the discounted return. In a formal notation value function is defined as follows

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$

where,

$$G_t = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$$

γ, $0 \leq \gamma \leq 1$, is the discount factor and since it is smaller than 1 in the later states rewards will be weighted lower in comparison with rewards in early states and R_{t+1} is the reward for transitioning from state s_t to s_{t+1}

3.2.4 Q-Learning

The Q-learning approach was introduced by Watkins in 1989[27]. It is a model-free algorithm, meaning that it does not need the prior knowledge of the environment, and the main goal is to find the optimal policy in the context of maximizing the expected value of the reward. The key of the algorithm consists of the Q-table and a Bellman equation. The Q-table is an 2D-array, with dimensions based on the states and the actions of the problem, and is used in order to save the Q values after each iteration. The

Q values are produced by a simple value iteration Bellman equation which is formally defined below.

$$Q^{new}(S_t, A_t) = (1 - a) \cdot Q(S_t, A_t) + a \cdot (R_{t+1} + \gamma \cdot maxQ(S_{t+1}, a))$$

- $(1 - a) \cdot Q(S_t, A_t)$: the current value weighted by the learning rate a
- $a \cdot R_{t+1}$: reward from state S_t to S_{t+1} weighted by learning rate a
- $a \cdot \gamma \cdot maxQ(S_{t+1}, a)$: maximum reward in state S_{t+1} weighted by a discount factor and a learning rate

The algorithm will come to an end if it reaches a termination state or a maximum number of iterations which in both cases is defined by the problem.

3.3 Related Work on ML-Based Service Placement

As mentioned in the previous chapter, the problem of placement is quietly well researched using different methodologies and various scenarios that cover a wide range of applications and domains of research. In the latest years, since the rapid development of ML, new ideas and optimization techniques are proposed in order to solve placement problems. To get a better grasp of how this problem is being tackled, we will focus on RL since it is a growing trend in the research area and seems to have important results compared with the current used algorithms and approaches.
In, [27] a Q-learning algorithm is proposed to place different services in the context of Software Defined Networks (SDN). The support of heterogeneous data and the better performance in comparison with the standard algorithms shows the advantages of modelling such a problem into a RL problem. In the area of Mobile Edge Computing (MEC), which is a key area regarding the deployment of 5G and beyond networks, different RL frameworks are proposed to achieve better QoS for the users, from minimizing the total response time and improving the number of services [28] to minimizing the cost of the overall cost of MEC design [29]. Although, in the first work the desired result is accomplished, the convergence speed and the stability of the model is something that should be improved. In the second work, only a static environment is taken into consideration and a small number of parameters too. Improving resource

usage in the continuum is also an important task especially for the new age networks. As proposed by the authors in [30] and in [31] the use of Deep Reinforcement Learning (DRL) can be proven a scalable and dynamic solution to accommodate the needs of the fast evolving networks. The use of it, as noticed in both papers, should be with the appropriate formulation since its complexity, is a major factor due to the fact neural networks are used. Making easily explainable models is a key to the proposed solutions since that will enable more adoption from the industry and will an easier expansion of these models will be possible.

Chapter 4

Proposed Model

4.1 Introduction

In this chapter, a thorough review of our suggested model is examined. In more detail, a reinforcement learning algorithm is proposed in order to find the ”optimal” placement of services in edge nodes that, in conjunction with a cloud server (node) ,will provide better QoS (latency, overall traffic, link utilization). The detailed functionality of the algorithm is presented as well the overall proposed model.

4.2 System Model

In this thesis, a reinforcement learning algorithm is proposed in order to place multiple services in an edge-cloud infrastructure. This environment consists of multiple edge nodes that are providing services to the network. The services could be anything from video streaming to tasks for UAV applications or even medical data. The cloud node is part of the overall network and is the one node that is going to process the data from the services the edge nodes will offload. To achieve that, a total of k nodes have to be chosen from the edges nodes and in them the services will be deployed.

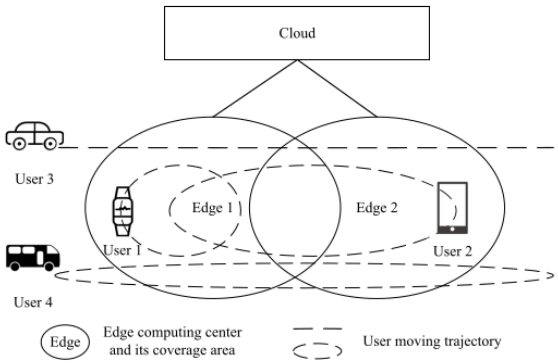


Figure 4.1: Edge-Cloud Environment

An abstracted edge-cloud environment is shown in the figure above where the network environment of this thesis is based on. The cloud is a centralized entity and the edge nodes will serve as potential candidates to accommodate the placement phase of the tasks. A similar work is found in the context of Content Delivery Networks (CDN) where clients act as publishers or subscribers of the content and the content delivery servers, to provide high availability and performance (”speed”), distribute the service spatially relative to end users. In this problem, the RL agent that makes the decision whether or not to choose a node as a placement node has a detailed view of the network topology and the status of the demands of each node. Thus, it is a centralized entity in the network architecture and can be integrated in the data centers network management centre or regarding telecommunication networks it can be in the Network Operations Centre (NOC) or in the context of SDN in the functionality of the controller.

4.3 Network Model

The network model will be formulated as a undirected graph with n vertices where each vertex will be an edge node or a cloud node. In total there will be $n-1$ edge nodes and one cloud node. In a formal notation, the topology is a graph $G = (V, E)$ where $|V| = n$. Also, w.l.o.g the graph G is a connected graph without any isolated node. Each node i is associated with a demand value d_i and therefore a demand vector $D \in \mathbb{R}^n$ is created. Also, delay between two nodes is represented as a adjacency matrix $P \in \mathbb{R}^{nxn}$. The distance between two nodes i,j is p_{ij} and is calculated as the number of hops between them in the graph G. In the figure below, a topology of the network

is shown. The hierarchy between cloud and edge nodes as presented in the previous section is logical, since the connections between the nodes is our only interest. In the figure below the cloud node is depicted with yellow, as a node with a higher degree it is a better candidate for a cloud node since it represents realistic scenarios more accurately.

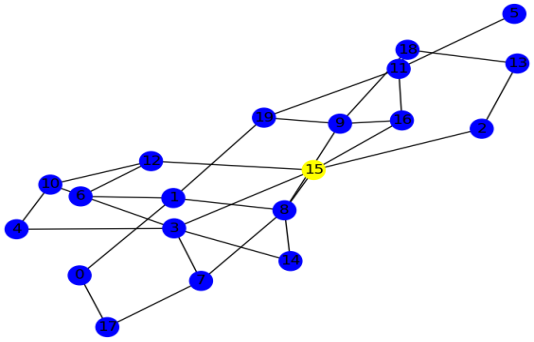


Figure 4.2: Sample graph with 20 nodes.

4.4 Problem Formulation

The placement problem, with the aim to minimize the overall traffic and delay, is formulated as a Markov Decision Process (MDP) and characterized as 4-tuple (S,A,P,R). In more detail:

- S is the finite state space. This space in this problem formulation represents the configuration of k different placements in the given topology. The total number of possible states is $n-1$ since the cloud node is not considered as a possible placement node.
- A is the finite action space. There is a close reference with the state S and an action is defined by the choice of location to place the k replication nodes each time. There are a total of $n-1$ possible actions at each state. Thus, the state-action pairs are $(n-1)*(n-1)$.
- P is the transition strategy. The strategy that is selected is epsilon-greedy. Epsilon-greedy is a simple policy that tries to balance between exploration and exploitation. The epsilon-greedy, where epsilon refers to the probability of choosing to

explore, exploits most of the time with a small chance of exploring. With a probability of e a random action is chosen and with a probability of $1 - e$ the best Q-values is chosen.

– R is the reward function which is associated with each state-action pair.The reward function is calculated as the product of the distance of the placement node to the cloud node and the sum of the demands of the nodes in this path. In more detail, if i the node we examine as possible placement node and j is the targeted cloud node then p_{ij} = path (i,j) represents the length of the path from i to j. In this path, the sum of demands is $\sum d_m$, where m is the index of the nodes in the path. More formally, if r is the reward value of R then,

$$r = p_{ij} \cdot \sum_{m=i}^{m=j} d_m$$

4.5 Algorithm

In this section, the q-learning algorithm that is proposed to solve the placement problem, is analyzed. The main goal is to provide better QoS for the users and the metrics that are optimized are the overall traffic and the delay.

Algorithm 1 Q-learning Algorithm for Service Placement

Initialize Q_matrix to a zero matrix
Total number of iterations T
Define k number of placement nodes
Set t = 0
while $t \leq T$ **do**
    $state \leftarrow$ Random integer between 0 and $num_of_nodes - 1$
    while $state = cloud_node$ **do**
        $state \leftarrow$ Random integer between 0 and $num_of_nodes - 1$
    $action$ based on e-greedy policy
    $reward$ based on reward function R
    Define the next state and set of actions
    Update the $Q(s, a)$ based on the Bellman equation
Return the top k best placements from Q-matrix

The proposed algorithm is based on Q-learning[32], which is used to find the optimal state-action pair combination and formally expressed as $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. In the proposed scheme, the state is randomly chosen but it should not be the cloud node. Then based on e-greedy policy (line 9) an action is selected in order not to get stuck on local maximum during the training process. This optimal state-pair combination is used to calculate the optimal Q value and is expressed by the Bellman equation as follows (line 12):

$$Q^{new}(S_t, A_t) = (1 - a) \cdot Q(S_t, A_t) + a \cdot (R_{t+1} + \gamma \cdot maxQ(S_{t+1}, a))$$

To calculate this value, as shown from the equation above, the next state is needed (line 11) and the reward function (line 10). Also, important parameters are the discount factor γ and the learning rate α . The learning rate is used to determine the degree to which freshly learned knowledge supersedes previously learned information whether the rewards in the future have the same importance as in the past. It should be mentioned that based on [33] the q-learning algorithm will converge in optimality in a rapid way.

Chapter 5

Experimental Analysis

5.1 Introduction

In this section, the experimental results will be presented while applying the Q-learning algorithm. The proposed algorithm will be compared to an iterative solution proposed in [16] in different size of network topologies and multiple number of placement locations. The key metrics that will be analyzed is the overall network traffic and the delay which is measured with the average hop distance between the location of the placement and the edge/client nodes. The experiments conducted show whether and in what context the proposed algorithm is a viable solution for the service placement problem.

5.2 Tools and Environment

In the course of this thesis, several tools and environments were utilized to facilitate data analysis, visualization, and network analysis.

5.2.1 Jupyter notebook

Jupyter Notebook was the primary environment used throughout the research and development phases of this thesis. This open-source web application is known for its interactive computing capabilities, which allow users to create and share documents

that integrate live code, equations, visualizations, and narrative text. Jupyter Notebooks support over 40 programming languages, including Python, which was the main language used in this thesis. The platform's interactive nature enabled an iterative approach to data analysis, where hypotheses could be tested and refined in real-time. Additionally, its ability to combine code with rich text elements, such as Markdown, LaTeX equations, and HTML, provided a cohesive and comprehensive way to document and present complex analytical workflows. This feature was particularly useful for maintaining a clear narrative thread throughout the research, making the work easily understandable and reproducible. Jupyter Notebook's extensive ecosystem of extensions further enhanced its functionality, supporting tasks ranging from version control with Git to collaborative editing, which was invaluable for maintaining a rigorous and organized research process.[34]

5.2.2 NumPy

NumPy, short for Numerical Python, was a cornerstone of the data manipulation and computational efforts in this thesis. As a fundamental package for scientific computing with Python, NumPy offers support for multi-dimensional arrays and matrices, along with a vast collection of mathematical functions to operate on these arrays. Its powerful N-dimensional array object, ndarray, enabled efficient and intuitive handling of large datasets. NumPy's performance is largely due to its ability to leverage optimized C and Fortran libraries, allowing for operations on arrays to be executed at high speed. This capability was crucial for performing a wide range of numerical tasks, including statistical analysis, linear algebra, and random number generation, all of which were essential components of the research. The library's integration with other scientific computing tools and its compatibility with a variety of data formats ensured a seamless workflow, facilitating the import, manipulation, and export of data throughout the analysis process.[35]

5.2.3 Matplotlib

Matplotlib was the primary tool used for data visualization in this thesis. As a comprehensive library for creating static, animated, and interactive visualizations in Python,

Matplotlib excels in producing publication-quality plots and charts. The library's design is rooted in the principles of clarity and precision, allowing users to create highly customized visual representations of data. Its extensive plotting capabilities include support for line plots, scatter plots, bar charts, histograms, pie charts, and many more complex visualizations. Matplotlib's versatility extends to its ability to handle a variety of data formats and its compatibility with other scientific libraries such as NumPy and pandas. This made it an invaluable tool for transforming raw data into insightful visualizations, facilitating the identification of trends, patterns, and anomalies. The ability to create intricate, multi-faceted plots with detailed annotations and custom styling was particularly beneficial for communicating the findings of this research effectively.[36]

5.2.4 NetworkX

NetworkX was used for the analysis of complex networks. NetworkX is a Python library designed for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. It provides tools for the analysis of the relationships and connections within data, enabling the examination of network properties and the implementation of various network algorithms. This was particularly valuable for exploring the paths within the network topology and calculate the metrics for the given algorithms.[37]

5.3 Configuration of the Environment

5.3.1 Topology

The topology was created using the package NetworkX and a random generator. The generator has as an input the number of nodes and the average degree of the network. The cloud node, which is the reference node is selected, after the topology is generated, with the reasoning that it should have a big degree and should be ”centered” at the network. In that way, a more realistic environment is achieved.

5.3.2 Algorithm

In this section, the configuration parameters that are used to run the Q-learning algorithm is presented. In more detail, as stated and in previous chapters the tunable parameters are α which is the learning rate the γ which is the discount factor and T is the total number of iterations that the algorithm will execute and ϵ that is used in e-greedy policy. In the next table, the parameters and their associated values are shown.

Table 5.1: Q-learning Parameters

Learning rate α	0.1
Discount factor γ	0.9
Possibility ϵ	0.1
Number of iterations T	500

Regarding the number of iterations, after several executions of the algorithm and having in mind the trade-off between execution time and convergence, the mentioned value was selected. Another approach would have been to set a threshold with respect to an solution and stop the iteration until it passes that threshold. The logic, behind considering the rest parameters, is based on the fact that we want to advance values that optimize the regarded parameters and we want a quick rate of convergence.

5.4 Experiments and Results

The metrics used to evaluate the efficiency of the proposed algorithm are the overall traffic network and the delay. Evaluating an algorithm using overall traffic is fundamental for ensuring its robustness, scalability, and effectiveness in real-world conditions. It provides a holistic view of performance, helps in maintaining high quality of service, and supports continuous improvement. On the other hand, delay is a key component of latency, a critical performance metric in network systems. By evaluating delay, one can gauge the responsiveness of the system.

In order to evaluate the algorithm with the mentioned metrics, two different experiments were made. In the first one, a medium size topology was used with various

numbers of locations to place the services. In more detail a size of $n = 60$ nodes was used and $k = 3, 8, 11, 15, 19, 23, 27$ to be the different number of locations.

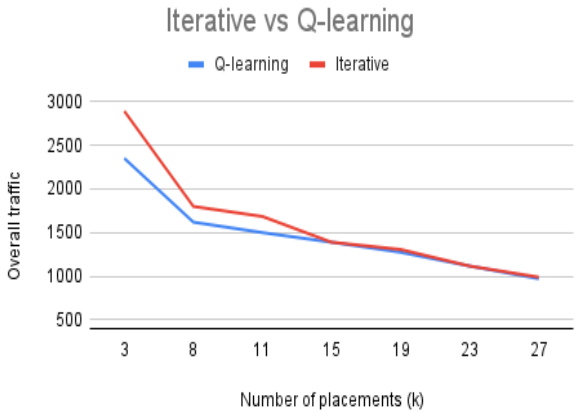


Figure 5.1: Overall traffic for different number of locations

As we can see from the figure, the Q-learning algorithm outperforms the iterative one in all of the settings for different values of k reagrding overall traffic for the network. This is a very important finding since it justifies not only the use of the RL algorithm but it shows that better solutions can be provided in comparison with algorithms that utilize only the graph characteristics.

In the next figure [5.2] we can measure the average hop count which represents the delay in our problem formulation. As we can see, for $k < 15$ our algorithm still outperforms the iterative one but then the iterative performs better. For large number of locations approximately 50% of the network the gap between of the two algorithms is quite negligible. This maybe due to the fact that for more placement locations the complexity of the q-learning in calculating and finding all the placements is close to the iterative one. Despite the fact that the decision the algorithm makes is outperformed, as the next figures [5.3,5.4,5.5] suggest the Q-learning finds the same solution as the iterative one but until it stops the iterations it is not the optimal due to convergence. This could be solved by increasing the number of iterations but this will cost in the execution time.

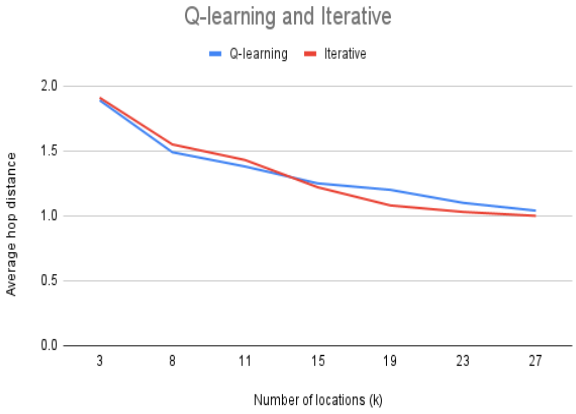


Figure 5.2: Average hop count for different number of locations

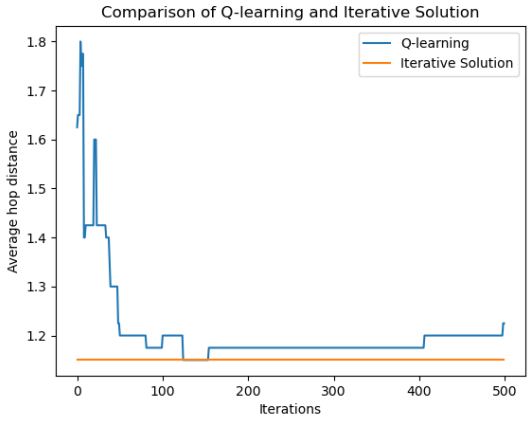


Figure 5.3: Average hop count for k = 19 and n = 60

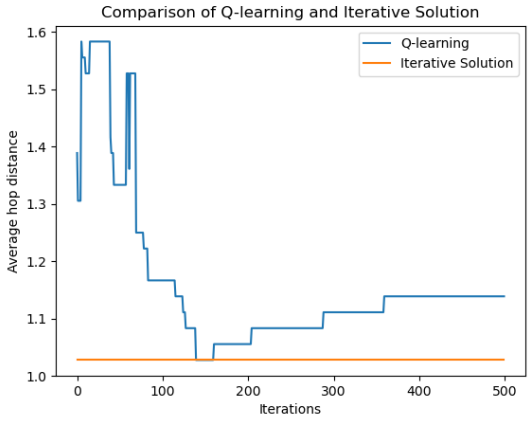


Figure 5.4: Average hop count for k = 23 and n = 60

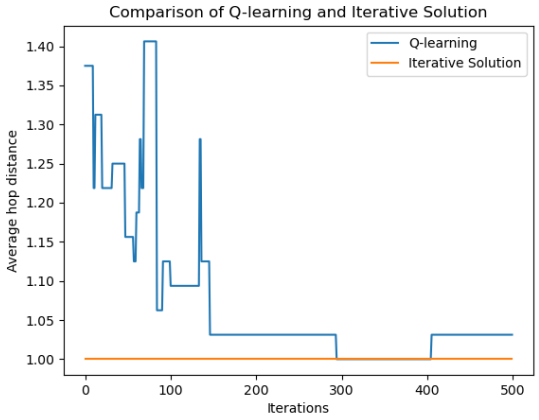


Figure 5.5: Average hop count for k = 27 and n = 60

The next experiment is conducted to evaluate the performance of the algorithm in different sizes of network topologies (small,medium,large) and different number of locations. This will show how scalable the algorithm is for different network configurations. The small network topology consists of $n = 40$ nodes ,the medium is using $n = 60$ nodes and the large is using $n = 90$ nodes. While, the number of possible locations is $k = 3, 5, 7$.

The first figures presented are the ones for a small network with size $n = 40$. The findings, for the majority of the cases depict that the iterative solution perform better than the proposed solution. This could be due to the fact that the algorithm cannot converge fast enough for small network topologies or due to the limited number of iterations.

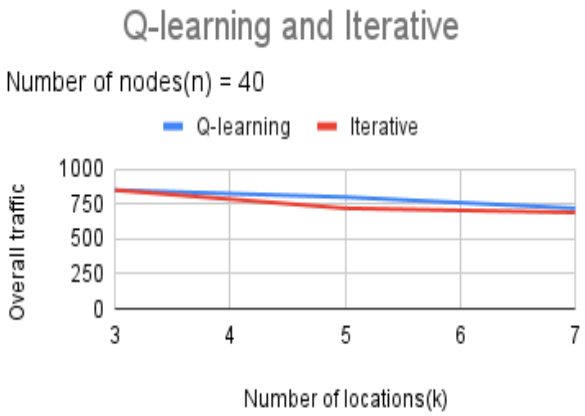


Figure 5.6: Overall traffic network for $n = 40$

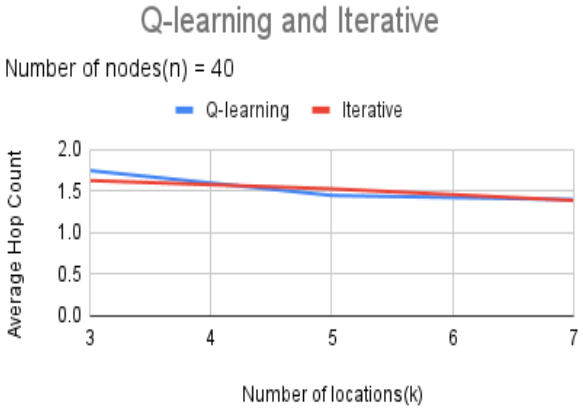


Figure 5.7: Average Hop Count for $n = 40$

On the other hand, the following figures show the performance of the two algorithms in a large scale network. The overall traffic metric is optimized better using the proposed Q-learning algorithm and that is also depicted as the number of locations increases. In the average hop count metric even though the iterative solution is performing better in the initial settings at the latest settings the two algorithms show similar behavior.

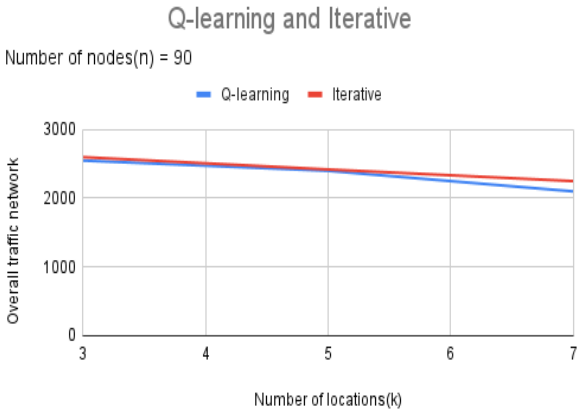


Figure 5.8: Overall traffic network for $n = 90$

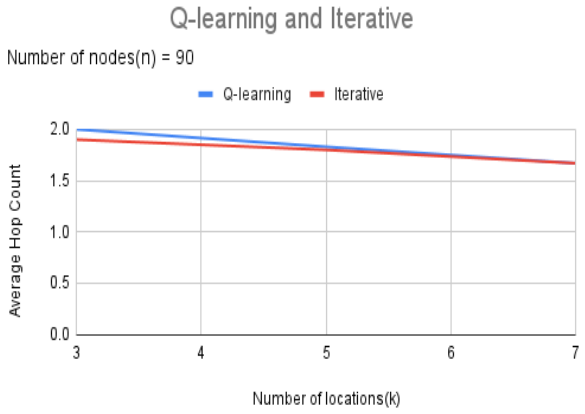


Figure 5.9: Average Hop Count for $n = 90$

In the medium size networks, as it has be shown in the first round of experiments, the Q-learning algorithm outperforms the iterative one. In the appendix of this thesis, all the plots from the experiments will be presented as a supplementary material and proof for the figures presented in this chapter.

In summary, the proposed algorithm show promising results regarding optimizing the overall traffic and the delay. Both of these metrics, were considered in the reward function of the algorithm. The algorithm could be useful for network topologies of medium size ($n = 60$) and number of locations for service placement around 20% of the network size. For smaller network topologies the iterative approach is performing better and it should be further investigated how to to change the Q-learning formulation in order to cover this case too. Last but not least, increasing the number of the nodes in the network it does not affect the performance of the algorithm but still the improvement of the network metrics is not as significant as in the medium size case.

Chapter 6

Conclusions

In this chapter, we provide a summary of the proposed work of this thesis and conclusions regarding the overall performance of the proposed algorithm. Also, ideas are presented that could help improve the proposed solution.

6.1 Summary and Conclusions

The main objective of this thesis was to propose an algorithm using ML to address the service placement problem in the computing continuum. In that context, taking into consideration all the related work around this problem and the different applications that this is appeared in, a system and a network model was proposed that could replicate a realistic scenario about the continuum and could be found in 5G/6G networks and IoT applications. Thus, the proposed offline Q-learning algorithm was proposed in order to provide better solutions regarding performance metrics in a network such as overall traffic and the delay. As shown, the algorithm can handle medium size networks in varying increasing number of possible locations and perform better than an iterative solution found in literature. Different scenarios were tested, with respect to the network size and the number of locations. The results showed that, for smaller sized networks the algorithm is not as efficient as in the previous mentioned settings and is outperformed by the compared algorithm. This is due to the fact that the Q-learning is too complex for small networks and the convergence rate is not the desired one. Also, it could be that the number of iterations used is not sufficient to find the same or better

solution.

6.2 Future Work

To extend the work presented in this thesis a numerous steps could be performed. First, the parameters introduced as network parameters could be extended in order to make the problem formulation more complex like link failures or clients that can leave or show up in the network and take into consideration metrics such as the bandwidth between node links, the demand capacity of each node/client and possible change in the demand during execution. A second step, could be to use a DRL algorithm. In that way, a neural network architecture could be utilized to make the algorithm more efficient and accurate that will also affect the scalability of the algorithm.

Bibliography

[1] https://databasecamp.de/en/ml/generative-adversarial networks.

[2] Qingfeng Lan. Variational quantum soft actor-critic. 12 2021.

[3] Mahadev Satyanarayanan. The emergence of edge computing. *Computer 50, 1 (2017)*, 2017.

[4] Cheol-Ho Hong and Blesson Varghese. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Comput. Surv. 52, 5, Article 97*, 2019.

[5] K. Tocze and S. Nadjm-Tehrani. A taxonomy for management and optimization of multiple resources in edge computing. *Wireless Communications and Mobile Computing, vol. 2018*, 2018.

[6] S. Buqrais M. Al-Roomi, S. Al-Ebrahim and I. Ahmad. Cloud computing pricing models: A survey. *International Journal of Grid and Distributed Computing, vol. 6, no. 5, pp. 93–106*, 2013.

[7] Yashpalsinh Jadeja and Kirit Modi. Cloud computing - concepts, architecture and challenges. 2012.

[8] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. 2017.

[9] https://www.docker.com/.

[10] Atul Tripathi, Isha Pathak, and Deo Vidyarthi. Modified dragonfly algorithm for optimal virtual machine placement in cloud computing. 2020.

[11] https://www.statista.com/topics/8409/kubernetes/editorsPicks.

[12] Anshuman Kalla, Chamitha de Alwis, Pawani Porambage, Gürkan Gür, and Madhusanka Liyanage. A survey on the use of blockchain for future 6g: Technical aspects, use cases, challenges and research directions. *Journal of Industrial Information Integration*, 2022.

[13] Fung Po Tso, Simon Jouet, and Dimitrios P. Pezaros. Network and server resource management strategies for data centre infrastructures: A survey. *Computer Networks*, 2016.

[14] https://en.wikipedia.org/wiki/Network_function_virtualization.

[15] https://www.networkcomputing.com/network-infrastructure/sdn-network-virtualization-and-nfv-in-a nutshell.

[16] G. S. Paschos-D. Katsaros V. Sourlas, P. Flegkas and L. Tassiulas. Storage planning and replica assignment in content-centric publish/subscribe networks. *Computer Networks*, December 2011.

[17] G. S. Paschos-D. Katsaros V. Sourlas, P. Flegkas and L. Tassiulas. Storing and replication in topic-based publish/subscribe networks. *Proceedings of IEEE Globecom*, 2010.

[18] Racha Gouareb, Vasilis Friderikos, and A. Aghvami. Virtual network functions routing and placement for edge cloud latency minimization. *IEEE Journal on Selected Areas in Communications*, 2018.

[19] Zhiqi Chen, Sheng Zhang, Can Wang, Zhuzhong Qian, Mingjun Xiao, Jie Wu, and Imad Jawhar. A novel algorithm for nfv chain placement in edge computing environments. 2018.

[20] Masoud Taghavian, Yassine Hadjadj-Aoul, Geraldine Texier, Nicolas Huin, and Philippe Bertin. A fair approach to the online placement of the network services over the edge. 2023.

[21] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros. Distributed placement of service facilities in large-scale networks. 2007.

[22] Brandon Heller, Rob Sherwood, and Nick McKeown. The controller placement problem. 2012.

[23] Miroslaw Klinkowski. Optimized planning of du/cu placement and flow routing in 5g packet xhaul networks. *IEEE Transactions on Network and Service Management*, 2023.

[24] https://en.wikipedia.org/wiki/Machine_learning.

[25] Afshin Rostamizadeh Mehryar Mohri and Ameet Talwalkar. Foundations of machine learning,2nd edition. 2012.

[26] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction (2nd edition). 2018.

[27] Ziyao Zhang, Liang Ma, Kin K. Leung, Leandros Tassiulas, and Jeremy Tucker. Q-placement: Reinforcement-learning-based service placement in software-defined networks. 2018.

[28] Yanlong Zhai, Tianhong Bao, Liehuang Zhu, Meng Shen, Xiaojiang Du, and Mohsen Guizani. Toward reinforcement-learning-based service deployment of 5g mobile edge computing with request-aware scheduling. *IEEE Wireless Communications*, 2020.

[29] Shuaibing Lu, Jie Wu, Jiamei Shi, Pengfan Lu, Juan Fang, and Haiming Liu. A dynamic service placement based on deep reinforcement learning in mobile edge computing. *Network*, 2022.

[30] Hani Sami, Azzam Mourad, Hadi Otrok, and Jamal Bentahar. Demand-driven deep reinforcement learning for scalable fog and service placement. *IEEE Transactions on Services Computing*, 2022.

[31] Yuming Xiao, Jiawei Zhang, Zhengguang Gao, and Yuefeng Ji. Service-oriented du-cu placement using reinforcement learning in 5g/b5g converged wireless-optical networks. 2020.

[32] Watkins. Learning from delayed rewards. phd thesis. 1989.

[33] M. J. Kearns and S. P. Singh. Finite-sample convergence rates for q- learning and indirect algorithms. *in Advances in Neural Information Processing Systems*, 1999.

[34] https://jupyter.org/.

[35] https://en.wikipedia.org/wiki/NumPy.

[36] https://matplotlib.org/.

[37] https://networkx.org/.

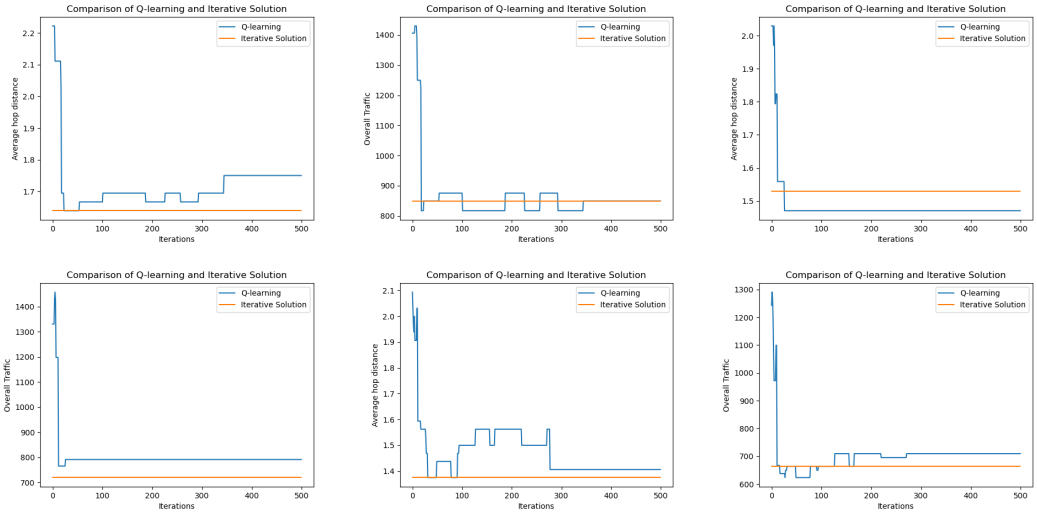
APPENDICES

Appendix A

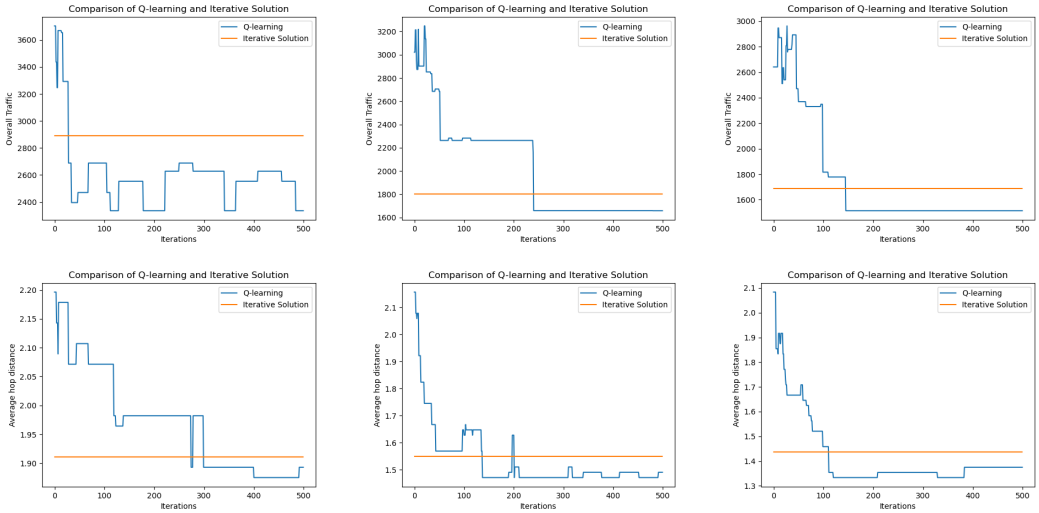
Figures of Experimental Analysis

This appendix has the figures of the experimental analysis from the comparison between the iterative approach and the proposed algorithm. The metrics from the below figures were combined and presented in the chapter of the experimental analysis.

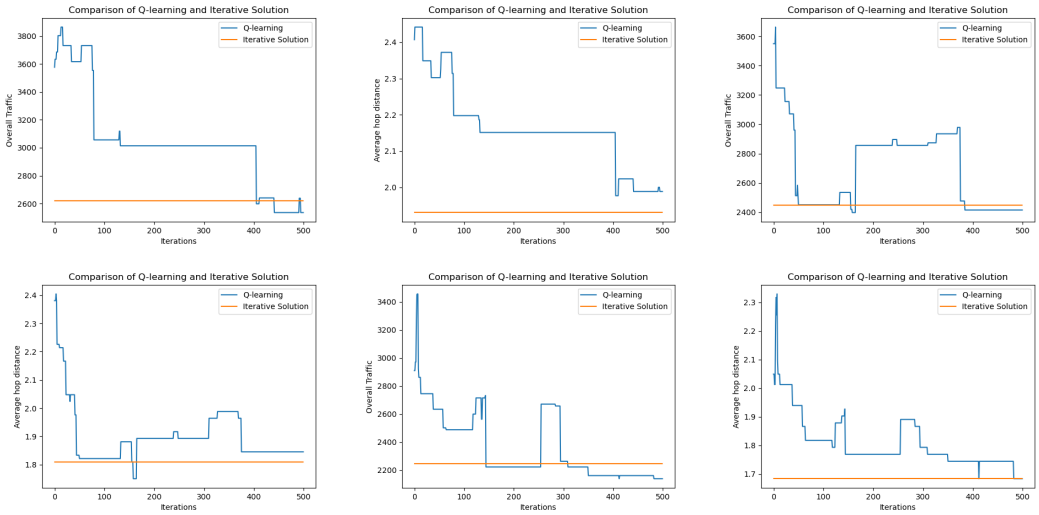
Number of nodes (n=40)



Number of nodes (n=60)



Number of nodes (n=90)



Appendix B

Implementation Code

B.1 Topology generator

```python
import networkx as nx
import numpy as np

def generate_adjacency_matrix(num_vertices, avg_degree):
    # Create a random graph with the desired number of nodes and average degree
    G = nx.gnm_random_graph(num_vertices, int(avg_degree * num_vertices / 2))

    # Get the adjacency matrix from the graph
    adjacency_matrix = nx.to_numpy_matrix(G)

    return adjacency_matrix, G

num_of_nodes = 30
avg_degree = 4
adjacency_matrix, graph = generate_adjacency_matrix(num_of_nodes, avg_degree)
initial_topology = adjacency_matrix
```

```
demand = np.array([np.random.uniform(1,20) for i in range(num_of_nodes)])
```

B.2 Algorithms

B.2.1 Proposed solution

```
    Q_matrix = np.zeros((num_of_nodes,num_of_nodes))
# Q-learning algorithm
epsilon = 0.1  # Exploration rate
alpha = 0.1  # Learning rate
gamma = 0.9  # Discount factor
k = 7
failure_probability = 0.05
overall_traffic_list = []
max_link_utilization_list_qlearning = []
avg_hop_distance_list_qlearning = []
T = 500
# Q-learning algorithm
for t in range(T):
    #adjacency_matrix = simulate_random_link_failures(initial_topology.copy(), failure_probability, cloud_node)

    state = np.random.randint(num_of_nodes)  # Random initial state
    while state == cloud_node:  # Ensure initial state is not the cloud node
        state = np.random.randint(num_of_nodes)

    action = choose_action(Q_matrix, state, epsilon)
    print("Action: ",action," State: ",state)
```

```
22      reward_val = reward(adjacency_matrix, state,
        ↪  cloud_node, action)
23      next_state = action  # Assuming action directly
        ↪  represents the next state
24      best_next_action = choose_action(Q_matrix, next_state,
        ↪  epsilon)  # Choose next action
25      Q_matrix[state, action] += alpha * (reward_val + gamma
        ↪  * Q_matrix[next_state, best_next_action] -
        ↪  Q_matrix[state, action])
26      state = next_state
27      #if state == cloud_node:  # Terminate if cloud node is
        ↪  reached
28          #break
29          # Calculate overall traffic at each iteration and
            ↪  accumulate
30      #print(Q_matrix)
31      matrix = np.sum(Q_matrix, axis = 1)
32      best_indices = np.argpartition(matrix, -k)[-k:]
33      overall_traffic =
        ↪  calculate_overall_traffic(adjacency_matrix,
        ↪  best_indices, cloud_node, demand)
34      overall_traffic_list.append(overall_traffic)
35      max_link_utilization, avg_hop_distance =
        ↪  calculate_metrics(adjacency_matrix, best_indices,
        ↪  cloud_node, demand)
36
        ↪  max_link_utilization_list_qlearning.append(max_link_utilizatio
37
        ↪  avg_hop_distance_list_qlearning.append(avg_hop_distance)
38      print("Iteration: ",t)
```

B.2.2 Iterative approach

```
adjacency_matrix = initial_topology
placement_list = np.zeros(num_of_nodes)
for pnode in range(num_of_nodes):
    m = 0
    for node in range(num_of_nodes):
        if node != pnode and node != cloud_node:
            path = dijkstra(adjacency_matrix, node,
             ↪  cloud_node)
            if pnode in path:
                m += demand[node]
            else:
                m += 0
    placement_list[pnode] = len(dijkstra(adjacency_matrix,
     ↪  pnode, cloud_node)) * m
best_indices = np.argpartition(placement_list, -k)[-k:]
overall_traffic =
 ↪  calculate_overall_traffic(adjacency_matrix,
 ↪  best_indices, cloud_node, demand)
max_link_utilization, avg_hop_distance =
 ↪  calculate_metrics(adjacency_matrix, best_indices,
 ↪  cloud_node, demand)
```