A Comparative Study on Service Migration for Mobile Edge Computing Based on Deep Learning

Sung woon Park

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Applied Science degree

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Over the past few years, Deep Learning (DL), a promising technology leading the next generation of intelligent environments, has attracted significant attention and has been intensively utilized in various fields in the fourth industrial revolution era. The applications of Deep Learning in the area of Mobile Edge Computing (MEC) have achieved remarkable outcomes. Among several functionalities of MEC, the service migration frameworks have been proposed to overcome the shortcomings of the traditional methodologies in supporting high-mobility users with real-time responses.

The service migration in MEC is a complex optimization problem that considers several dynamic environmental factors to make an optimal decision on whether, when, and where to migrate. In line with the trend, various service migration frameworks based on a variety of optimization algorithms have been proposed to overcome the limitations of the traditional methodologies. However, it is required to devise a more sophisticated and realistic model by solving the computational complexity and improving the inefficiency of existing frameworks. Therefore, an efficient service migration mechanism that is able to capture the environmental variables comprehensively is required.

In this thesis, we propose an enhanced service migration model to address user proximity issues. We first introduce innovative service migration models for single-user and multi-user to overcome the users' proximity issue while enforcing the service execution efficiency. Secondly, We formulate the service migration process as a complicated optimization problem and utilize Deep Reinforcement Learning (DRL) to estimate the optimal policy to minimize the migration cost, transaction cost, and consumed energy jointly. Lastly, we compare the proposed models with existing migration methodologies through analytical simulations from various aspects. The numerical results demonstrate that the proposed models can estimate the optimal policy despite the computational complexity caused by the dynamic environment and high-mobility users.

Acknowledgements

First, I would like to express my deepest appreciation to my supervisor, Professor Azzedine Boukerche, for allowing me to conduct my research at University of Ottawa and for all the resources and support he provided. Without his guidance and persistent help, my thesis would not have been possible.

I would also like to thank Dr. Shichao Guan and Dr. Peyvand Teymoori, who provided me with valuable suggestions on my research work and thesis writing. With their tremendous understanding and encouragement over the past years, it would be attainable for me to complete my study.

Finally, I must express my profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. My appreciation goes out to my spouse, Minhee, and my kids, Soyi and Chaewon. I cannot find the words to express my gratitude to them. I would like to give them all the honor.

Table of Contents

List of Tables

List of Figures

Abbreviations

AdaGrad Adaptive Gradient 36

Adam Adaptive Momentum Estimation 36

AHT service-area-handoff-time estimation scheme 20

AI Artificial Intelligence 3

ANN Artificial neural network 31

AR Augmented Reality 1

BPTT Backpropagation through time 40

CNN Convolutional Neural Network 36

D2D device-to-device 8

DAMP Destination And Mobility Path Prediction 24

DBSCAN Density Based Spatial Clustering of Applications with Noise 32

DDQN Dueling-DQN 65

DNN Deep Neural Network 28

DPM Destination Prediction Model 24

DQN Deep Q-Network 34

DRL Deep Reinforcement Learning 3

DTT data-transfer-throughput estimation scheme 20

E-UTRAN Evolved Universal Terrestrial Radio Access Network 12

E2E end-to-end 26

EC Edge computing 4

EPC Evolved Packet Core 12

ESM Extensive Service Migration 47

FC-SDES fog computing supported software-defined embedded system 9

FCL fully connected layers 65

FGMBGA fine-grained migration based on a generation algorithm 21

FMC Follow Me Cloud 14

GOP Green-Oriented Problem 21

H-SCM hierarchical SCM 12

HD High-definition 8

IoT Internet of Things 1

L-SCM local SCM 12

LSTM Long short-term memory 4

MCC Mobile Cloud Computing 2

MDCs micro data centers 20

MDP Markov Decision Process 18

MEC Mobile Edge Computing 2, 4

MINLP Mixed Integer Linear Program 21

MIQP Mixed-Integer Quadratic Programming 19

ML Machine Learning 28

MLP Multi-Layer Perceptron 36

MSMP mobility-based services migration prediction 20

NAG Nesterov accelerated gradient 36

NM Navigation Map 24

PPM Path Prediction Model 24

PRIMAL PRofIt Maximization Avatar pLacement 19

PSO Particle Swarm Optimization 45

QoS Quality of Service 2

R-SCM remote SCM 12

RAN Radio Access Network 9

ReLU Rectifier linear unit function 36

RIEs radio interfacing equipment 16

RL Reinforcement Learning 29

RMSProp Root Mean Square Propagation 36

RNN Recurrent Neural Network 36

SCC Small Cell Cloud 12

SCeNBce Small Cell enhanced NodeB's cloud-enabled 12

SCM small cell cloud manager 12

SDN Software Defined Network 13

SGD stochastic gradient descent 36

SL Supervised Learning 29

SMM service migration management scheme 20

SVM Support vector machine 31

Tanh hyperbolic tangent function 36

UC User Contextual 24

UE User Equipment 2

UFVLT User Frequently Visited Location Trace 24

UL Unsupervised Learning 29

V2X Vehicle-to-everything 2

VH-SCM virtual hierarchical SCM 12

VM virtual machine 7

VR Virtual Reality 1

Chapter 1

Introduction

With the continuous advancement of mobile devices and the dramatic development of network technology, people's everyday lives have enhanced tremendously. Based on the advanced equipment and the evolved network, various high-tech concepts, such as Smart City, Smart Factory, Autonomous Vehicle, Internet of Things (IoT), Virtual Reality (VR), and Augmented Reality (AR), have emerged, leading to an exponential increase in the complexity and throughput of network communication, requiring more computational power. Mobile applications derived from these advanced environments have required significant computation resources and energy consumption. Despite the constant evolution of capabilities, mobile devices are insufficient to keep up with rapidly increasing demands. Thus, Cloud Computing, which provisions flexible computation and network, draws increasing attention from academia and industry to implement the technologies mentioned above [4].

1.1 Background and Problem Statement

The Cloud Computing paradigm has directed in the ubiquitous era of connecting all devices and people to the network [209]. Even so, there are inherent constraints on Cloud

computing derived from the physical distance between User Equipment (UE) and cloud instances. The trade-off between proficient computational power and communication latency is significant, emanating tremendous efforts to obtain an adequate solution. Consequently, Mobile Cloud Computing (MCC) has emerged to provide cloud services in proximity to mobile equipment, coping with the inherent overhead and latency of Cloud computing. Several conceptual models and proposals related to EC, such as Cloudlet [169], Fog [12], and Mobile Edge Computing (MEC) [112], have been devised to improve the performance of the technology. Furthermore, mobile application scenarios based on MCC have been presented to bring the computing capability to the proximity of UE [96].

MEC deploys cloud devices as quickly as possible with users to efficiently use cloud capabilities. Due to these characteristics, MEC is in the spotlight as a paramount technology for realizing future-oriented concepts that require rapid interaction, such as Vehicle-to-everything (V2X) communication and autonomous driving. Numerous studies have been conducted to discover a more profitable MEC idea. Multiple researchers have proposed the offloading model, which determines where services execute between the UE and the edge server. Similarly, resource allocation models have also been introduced to utilize the limited cloud servers efficiently [59, 60, 70, 74, 83, 88, 102, 103, 116, 122, 185, 198, 210, 219]. Furthermore, as it becomes challenging to predict users' movement patterns due to their diversity and complexity, service migration, which moves services within distributed MEC servers, is a prominent solution to cope with the issues. Network performance degradation caused by the mobility of UE and the limited coverage of clouds can be mitigated by migrating services according to the UE location [194].

The service migration in MEC is a complicated optimization problem since the decision on whether, when, and where to migrate depends on many dynamic environmental variables involving user mobility, communication channel characteristics, and resource availability [128]. The migration model pursues to retain competent Quality of Service (QoS) with low latency and reasonable cost. It constantly moves the service to the proximity as responding to the user mobility to enhance communication efficiency. Although frequent migrations can reduce latency or reinforce network usage efficiency, they derive increasing

the migration cost and deteriorate energy efficiency and QoS on account of downtime or communication channel management overhead [37]. In this regard, notable achievements have been yielded from research on service migration. Nonetheless, they are insufficient to manage the real-world environment with more complex scenarios.

1.2 Motivation and Contribution

In recent years, owing to advancements in Artificial Intelligence (AI), several innovative algorithms based on enforced optimization methods have been developed, which have been utilized in different fields to solve a wide range of optimization problems. In terms of the service migration in MEC, the application of AI technology to recognize and respond to multi-dimensional information from the surrounded environment has been extensively studied, and promising outcomes have been achieved. Therefore, It is conceivable to implement an active solution that is able to make optimal decisions dynamically in the face of any unforeseen situation rather than simply operating in the known environment.

In our work, we propose a novel service migration model based on Deep Reinforcement Learning (DRL) that deals with the complicated real-world service migration scenarios in MEC. While devising a single-user DRL-based service migration model, we propose the possibility of applying DRL to complex service migration problems. Furthermore, we can achieve realistic scenarios by considering more environmental elements and multiple users with random mobility patterns. Different from the single-user models, this extensive service migration model allows the agent not only to recognize the users' mobility pattern but also to manage the migration process with a comprehensive perspective, which involves more environmental constituents, such as migration and transaction costs, energy consumption, and computational capabilities. The DRL methods are also applied to deal with the high computational complexity caused by the model's extended parameters. The major contributions of our work are as follows:

- We design an extensive service migration model that performs the service migration

according to the optimization policy related to the encountered circumstances. At the same time, we also formulate the problem of service migration in a multi-user and multi-server system. The problem consists of various components such as the migration and transaction costs, QoS, the energy efficiency of the mobile devices, and the load balancing on the servers. We also consider practical, real-world scenarios where environmental conditions vary randomly and users follow random mobility patterns.

- To cope with a high computational complexity that may be caused by an unpredictable dynamic environment, we employ DRL algorithms, as well as their improved versions, such as Dueling and Long short-term memory (LSTM), to recognize the environmental factors and estimate the optimal policy.

- A comprehensive set of simulations are conducted to evaluate the effectiveness of the proposed DRL algorithms in a dynamic environment with high mobility users. Then, The performance of the proposed algorithms has been compared with each other as well as other conventional and heuristics, such as Always-Migration, No-Migration, and Q-learning from various aspects.

1.3 Thesis Outline

The remainder of this paper is organized as follows.

In Chapter 2, we start with the basic knowledge of Edge computing (EC) concepts. After that, we overview the related literature related to service migration in the MEC area that contains solutions to improve the efficiency of the service migration process. Furthermore, conventional approaches to solving service migration problems are presented. We then overview the state of the art with regard to DL technology and discuss learning-based solutions for service migration.

In Chapter 3, we propose an enhanced service model for the optimization problem of the service migration with a single user. An expanded reward function is formulated with various determinants to determine the optimal migration policy. We demonstrate the efficiency of the model by analyzing the simulation results.

In Chapter 4, we introduce a multi-user service migration model to estimate the optimal solution to the proposed problem. We present the numerical results regarding our proposed DQN-based models and compare them with the existing migration models to validate the improvement of the proposal.

Chapter 5 concludes our work and presents the open issues and challenges related to the service migration problem.

Chapter 2

Related Work

In this chapter, we first overview the detail of EC technology, including primary concepts and literature related to them, and provide specific information on MEC architectures. Then, we show the importance of service migration in MEC by discussing the related literature that contains solutions to enhance the efficiency of the migration strategy. In the subsequent, we review the valuable DL technology regarding functionalities and applications. The hierarchy of existing methodologies is described with use-cases that utilize DL to accomplish the optimal solutions for diverse issues.

2.1 Edge computing technology

As the applications of EC, which pursue deploying computing, storage, and service resources in the proximity of UE, have been diffused, multiple EC structures have been introduced for practical processing or deploying tasks. UEs generate requests that involve services or data to be processed at the edge or cloud server in the EC environment. These tasks are offloaded depending on the network communication conditions and the computing capacity of nearby edge servers or cloud servers, as well as migrated following the user's

mobility. Although they have slightly different technical approaches, the EC architectures are generally classified into three main concepts: Cloudlet, Fog, and MEC.

In this section, we discuss existing studies related to detailed features for each of the three prominent EC architectures. Furthermore, we also describe the MEC concepts' fundamental principles, including characteristics, merits, and demerits.

2.1.1 Cloudlet

Cloudlet architecture, which consists of three-tier hierarchies: mobile devices, cloudlet, and cloud, exploits a data center in a box to bring the centralized cloud to the vicinity of UEs, as shown in Figure 2.1. Although the computational performance and storage capacity of end-user devices are enormously improved, it is inadequate to process rich media content solely. Cloud computing also has restrictions in performing services that require high-speed processing because of the latency caused by the distances to the server. Cloudlet enables vivid interactive response to tackle end-to-end latency by deploying the virtualized service on a nearby cloudlet, generally one hop from UEs [169, 170].

In [170], the improved cloudlet concept, GigaSight, was proposed to solve the high cumulative data rate issue of incoming videos from many cameras. The authors employed a type of "analytics" supported on cloudlets, which preprocesses video streams by editing out frames or blurring objects. T. Verbelen et al. [188] introduced a fine-grained cloudlet idea, which manages virtual machine (VM) applications on a component level to avoid transferring a whole VM from the cloud to the cloudlets.

The applications of cloudlet technology intensively have been studied to ensure the efficient usage of cloud resources. Some researchers have conducted studies related to improving cloudlet performance by evaluating the impact of user mobility or execution locations and implementing novel models based on cloudlet [72, 75, 105, 120]. Moreover, extended studies based on cloudlets, such as Data Prefetching [89], Data Caching [7, 104], and Data Transfer [90], have proceeded.

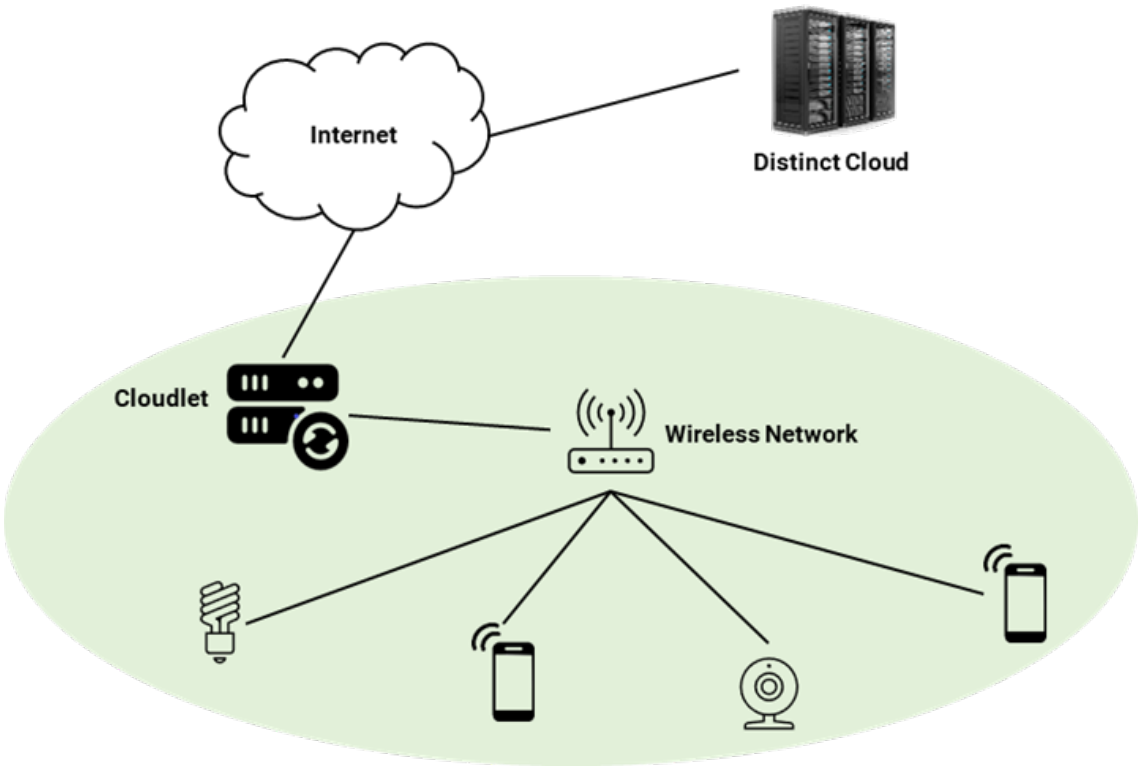


Figure 2.1: Cloudlet Computing.

2.1.2 Fog

Fog computing, first introduced by CISCO, is an extended notion of a Cloud computing framework to support a number of IoT services and applications. The virtualized platform extends computing and storage capacity in distinct cloud computing servers to the end-users' neighborhood [12]. As shown in Figure 2.2, the Fog nodes composed of numerous Fog components are widely distributed close to the UEs and interact with not only each UE but also other Fog nodes. Services and applications, such as real-time or big data analysis and High-definition (HD) video, can be performed in Fog nodes. Apart from that, fog nodes are also required to interplay with the cloud to keep data and service coherence.

Several applications have been proposed based on the Fog paradigm to facilitate the concept's benefits. In the paper [154], the authors devised a device-to-device (D2D) of-

floading framework to minimize the time-average energy consumption for all users' task executions. K. Intharawijitr et al. [113] concerned latency regarding communication and computing delay in Fog architecture. They established a mathematical model to clarify the latency in the Fog network. In [216], a computation-efficient solution based on mixed-integer nonlinear programming is offered to reduce the computation and transmission latency in a fog computing supported software-defined embedded system (FC-SDES). The authors concentrated on the task scheduling and image placement problem to balance the workload on the UE and fog node sides.

Meanwhile, the challenges are also apparent despite the advantages of Fog computing in terms of low latency, bandwidth economy, heterogeneous features, and low energy consumption. Fog nodes, which are spread out innumerably, have relatively limited security preparation and monitoring compared to cloud servers. Thus, security issues such as authentication, secure communication, and privacy are inevitable [139, 177].

2.1.3 MEC

Emerging technologies for 5G networks and advanced mobile devices facilitate the development of MEC, which provides virtualized service components and cloud computing capacities at the edge of the mobile network. MEC results from ETRI's efforts to achieve low latency, proximity, high bandwidth, and real-time insight over the Radio Access Network (RAN). As described in Figure 2.3, MEC servers exist in the base stations to ensure a high-quality network operation and provide cloud services in the proximity of mobile users. The MEC paradigm, which supports high data rates and low latency computation, enables diverse, innovative services such as augmented reality, Intelligent Video Acceleration, Connected Vehicles, and IoT gateways. It allows mobile users accessing the base station to efficiently use the abundant computational power with less latency, facilitating flexible applications that demand intensive computation and ample storage space. Unlike Cloudlet or Fog, which can be located anywhere for convenience, the MEC server must be deployed near the BS due to the dependency on mobile operators [4, 98, 112, 125].

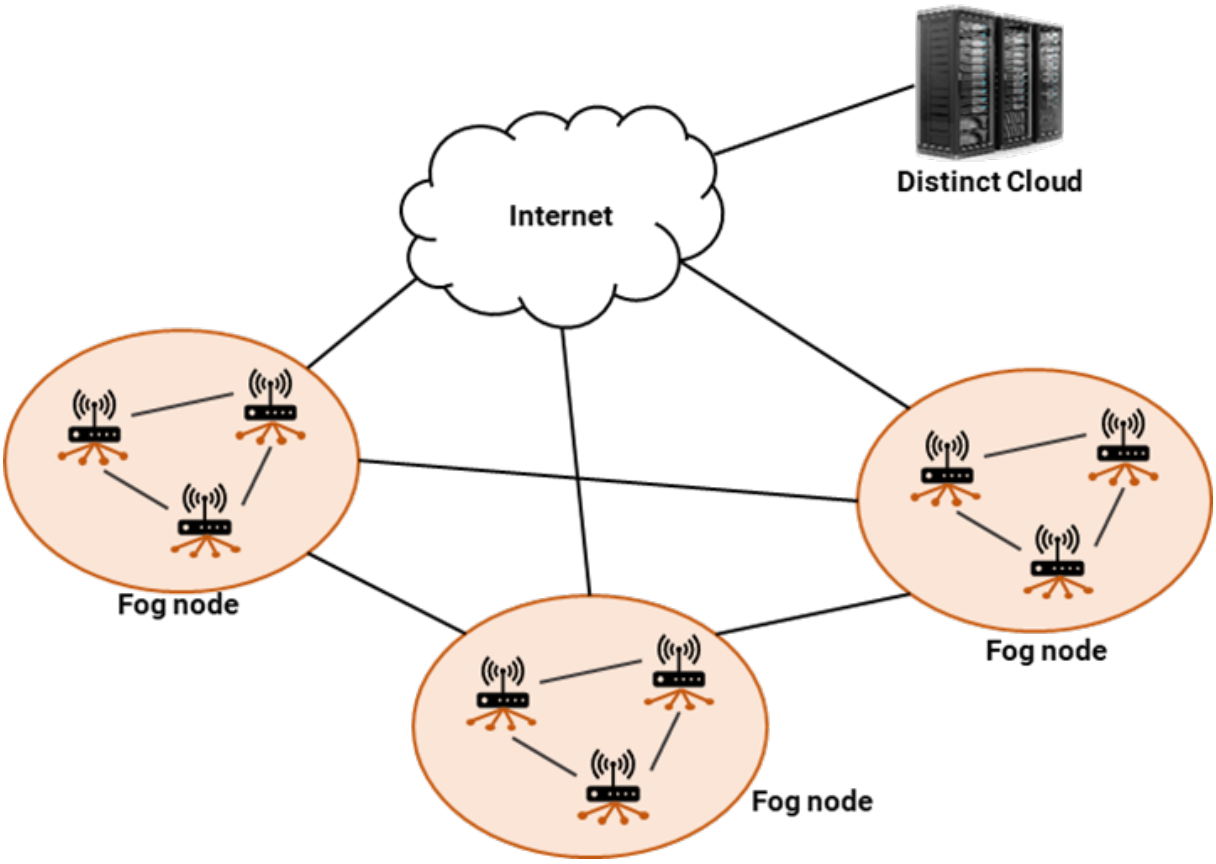


Figure 2.2: Fog Computing

On the other hand, there are still open challenges requiring attention, even though the advantage of MEC has introduced several opportunities and service usages in various aspects. In the architecture, the allocation and management of MEC resources are crucial for improving the quality of mobile networks. Therefore, it is necessary to implement a model that comprehensively considers resource allocation and service execution location. An alternative must be prepared to obtain an optimization solution to handle the complexity induced by complex factors and many users. It is also expected to focus on investigating a novel method to predict the mobility of fast-moving users and efficiently and persistently migrate virtualized services.

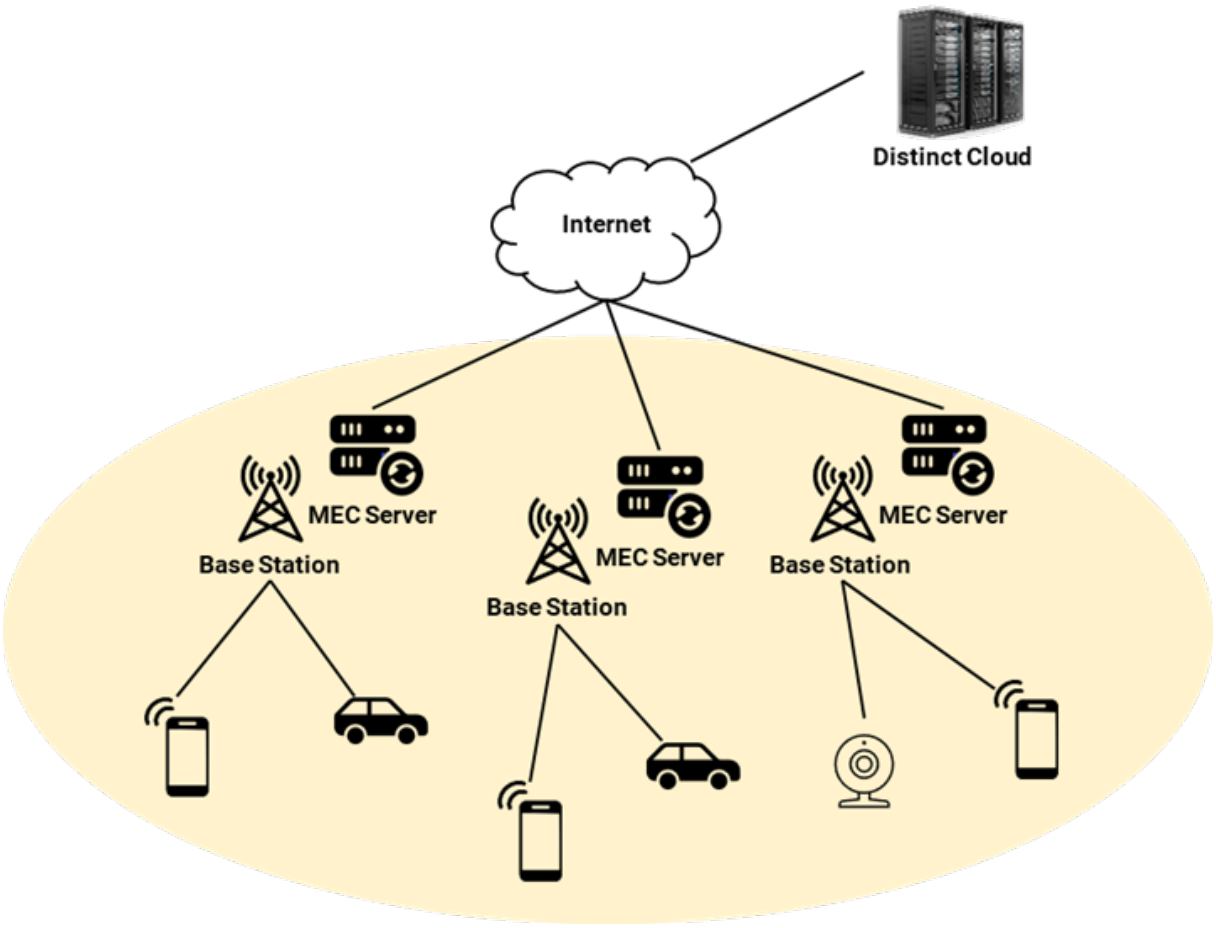


Figure 2.3: Mobile Edge Cloud Computing

2.1.4 MEC Conceptual framework

Numerous attempts to obtain a smoothly operated mobile cloud network architecture have produced various MEC concepts in the literature. Many optimization methodologies are adopted based on these concepts to derive more practical MEC models. We briefly describe each MEC concept and examine the features and applications.

Small Cell Cloud

Small Cell Cloud (SCC) is a more advanced technology than Cloudlet to provide higher spectral efficiency. The Small Cell enhanced NodeB's cloud-enabled (SCeNBce) enables short-range wireless access and proximity computation resources by deploying resources of computation and radio in the vicinity of the end-user. A small cell cloud manager (SCM) plays a significant role as a control entity of SCC, which is reliable in handling the cloud resources and is involved in the deployment of new services and the migration of services in progress to enhance the service utilization for the end-users [124].

According to the deployment of SCM, the architecture of the SCC network is categorized into the centralized SCM and the distributed hierarchical SCM, which is in a distributed manner. As shown in Figure 2.4, the centralized SCM has two options: the standalone SCM and the SCM as an extension. Concerning the first option, SCM is deployed closer to users within the radio access network to reduce communication overhead between the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and the Evolved Packet Core (EPC). The other option is to install SCM in the EPC to maximize the computing usage of all SCeNBces [10, 124].

In terms of the distributed hierarchical SCM, two novel architectures, a hierarchical SCM (H-SCM) and a virtual hierarchical SCM (VH-SCM), are introduced to control computation efficiently in the SCC network [10]. In H-SCM, the SCM is physically separated into a local SCM (L-SCM) and a remote SCM (R-SCM), as shown in Figure 2.5. The L-SCM adjusts several SCeNBces nearby and deals with relatively low complexity computing requests. Otherwise, the R-SCM conducts a role similar to the centralized SCM, which is able to exploit the computing capacity of all SCeNBces interacting with the EPC. Furthermore, the VH-SCM is devised to reduce the high cost of the H-SCM (see Figure 2.6). It virtualizes the L-SCM into the SCeNBces, which means that the functionality is incorporated into the deployed VMs. Therefore, it is not required to install additional hardware.

Fast Moving Personal Cloud

In the paper [193], MobiScud, an evolutionary mobile network architecture, was introduced to constantly provide personalized virtual machines to mobile network users near the framework. MobiScud is a standard mobile network based on cloud and Software Defined Network (SDN) technologies to support low latency services or applications. The architecture is displayed in Figure 2.7.

A MobiScud control interacts with the cloud resources and the SDN machines to offload traffic to a private VM hosted in the cloud. It also monitors the mobile network activity and routes the network data flow between the SDN and the cloud of the provider through the RAN. Although the network architecture has aroused appropriate usages of the computation resources with low latency, it has a limitation of allowing only the selected application to use closely located private VMs, and other requests are processed

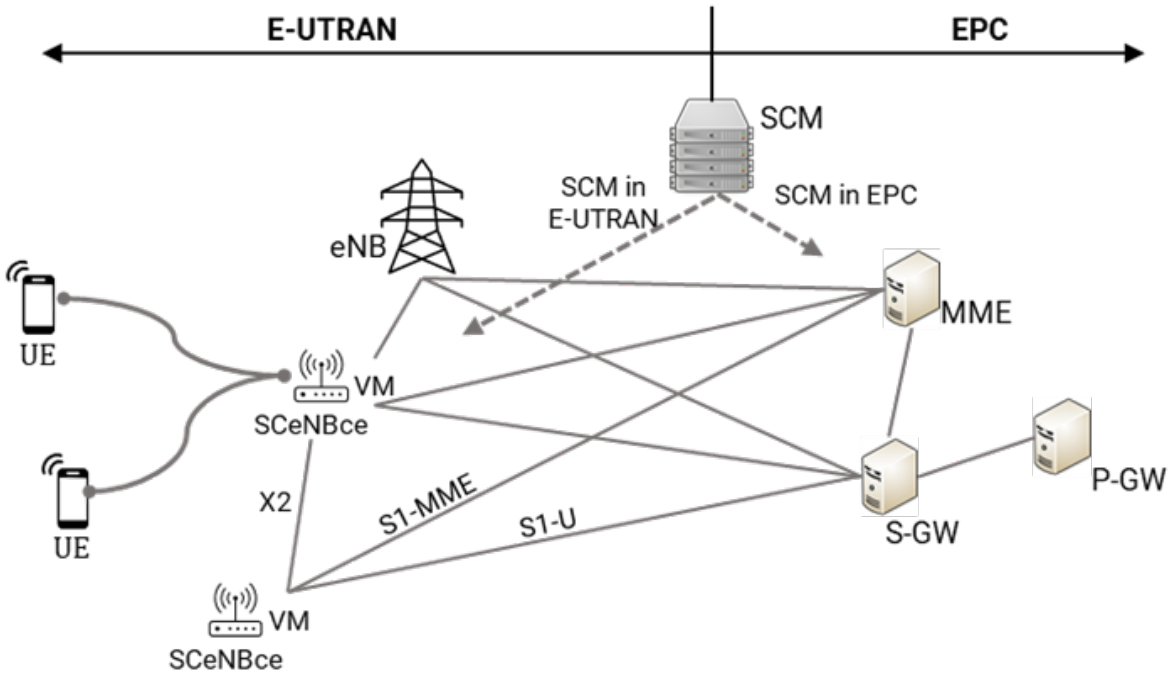


Figure 2.4: The centralized SCM in SCC architecture

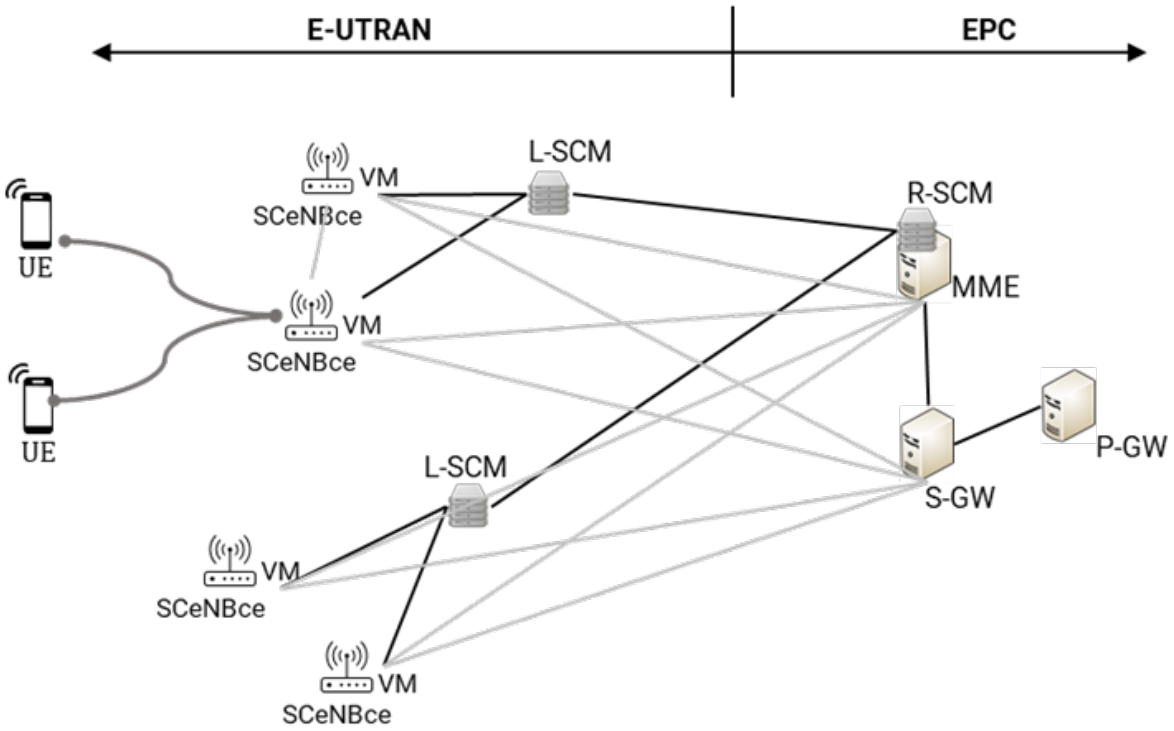


Figure 2.5: The structure of the hierarchical SCM

through the default gateway. Therefore, there are still optimization issues for choosing proper offloading.

Follow Me Cloud

The Follow Me Cloud (FMC) concept and framework was introduced by T. Taleb et al. to allow cloud services in the distributed datacenter to follow users on the move by involving the decision process [183, 184]. The two essential elements of the concept are the FMC controller and the DC/GW mapping entity. The DC/GW mapping entity connects the data centers to the distributed S-GWs and P-GWs based on diverse metadata statically or dynamically. These can either be two independent architectural components, two functional entities collocated with existing nodes, or operate as software on any data center of

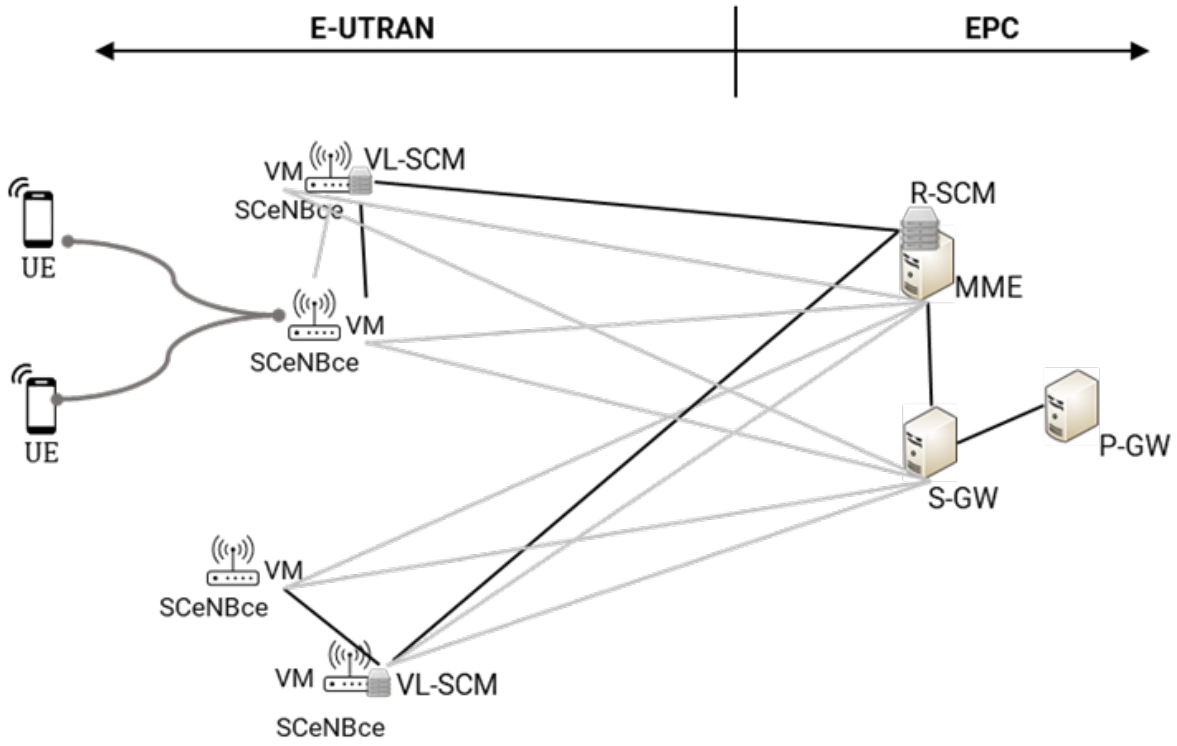


Figure 2.6: The structure of the virtual hierarchical

the fundamental cloud.

The framework does not demand extra complexity to the existing mobile network architecture and is highly reasonable, practical, and standardized. As illustrated in Figure 2.8, a user receives an application or a service from a distributed datacenter located nearby at a particular moment. Then, the user moves to a different place and continues to utilize the remaining service from the current nearest server since the FMC controller migrates the service.

CONCERT

In [121], the authors applied the SDN paradigm to devise a converged edge infrastructure for future cellular networks named as CONCERT concept. As shown in Figure 2.9, the

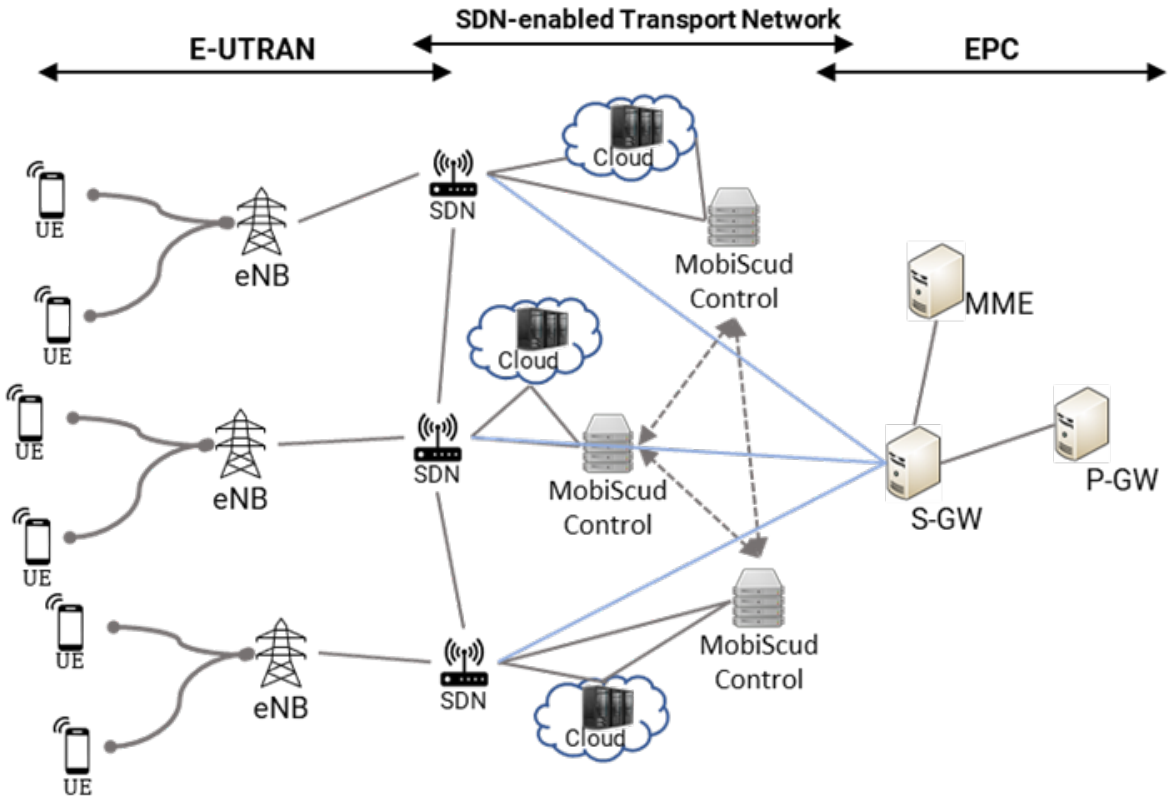


Figure 2.7: The architecture of MobiScud.

Conductor of the architecture coordinates and virtualizes data plane resources, such as the radio interfacing equipment (RIEs), the software-defined switches, and computational resources, resulting in the decoupling of the control and data planes. Although the control plane is denoted as one entity, the Conductor can be additionally designed hierarchically for better control scalability since it is a logically centralized entity.

The control functions interfacing toward the data plan include radio interfacing, managing wired network, and handling location-aware computing. The controller deals with data plane resources and receives network context information through dedicated control channels. All the resources for the interface related to the software-defined service, such as computational, networking, and radio, are virtualized. Based on these virtualized re-

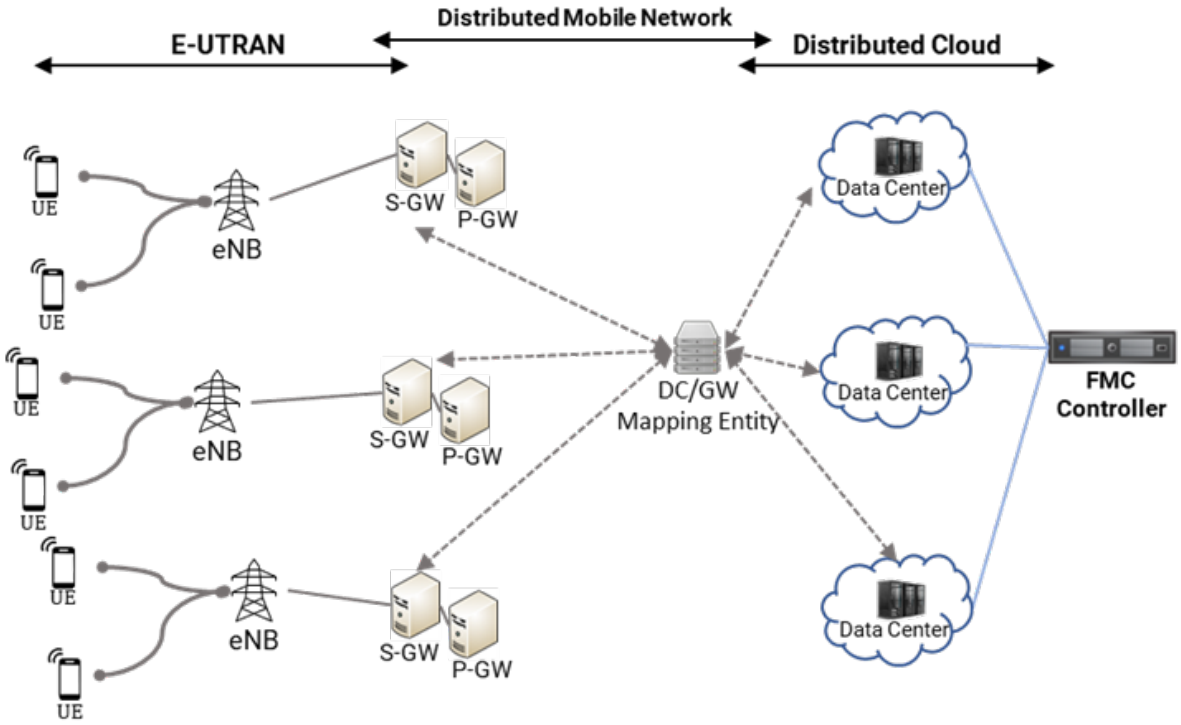


Figure 2.8: Follow-me cloud high-level architecture.

sources, CONCERT facilitates various applications of software-defined services.

2.2 SERVICE MIGRATION IN MEC

Due to limited resources and capacity in the MEC environment, it is imperative to efficiently allocate and manage resources related to the optimization problem of minimizing process latency, balancing loads, and maximizing computation efficiency. In the field of MEC, constant research has been conducted on resource allocation methods and service execution locations. As a result, various MEC concepts, such as SCC, MMC, Fast Moving Personal Cloud, FMC, and CONCERT, have been derived [125]. Besides resource distribution issues, energy consumption issues are attracting attention to appropriately control

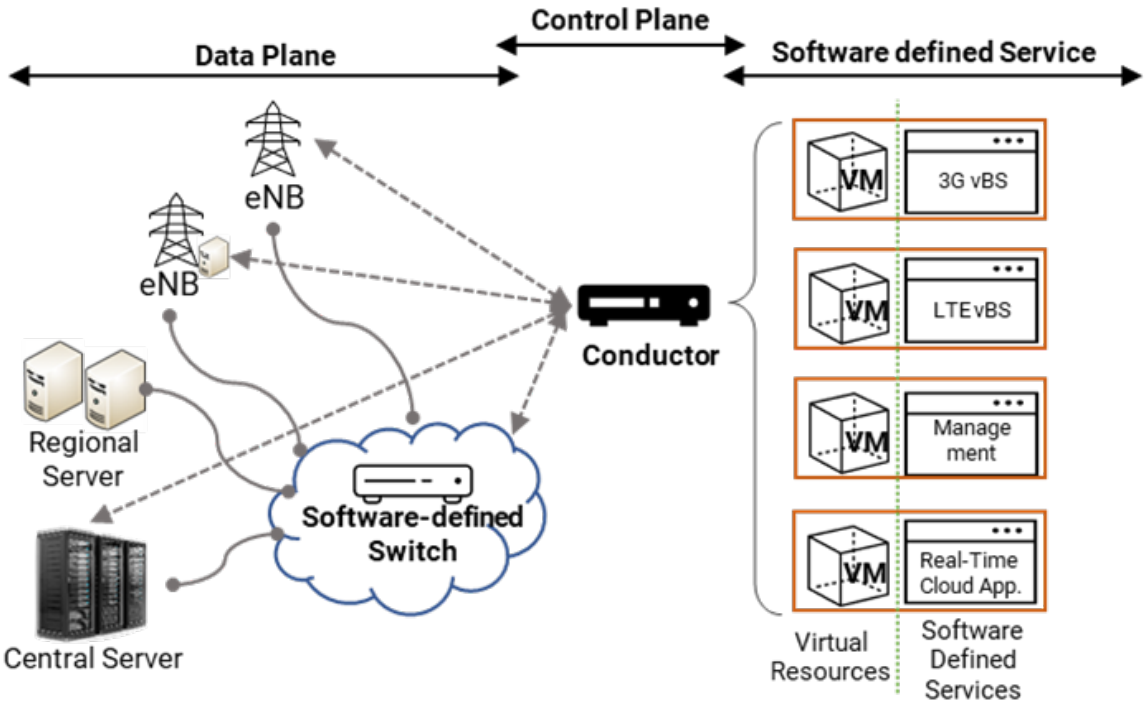


Figure 2.9: The architecture of CONCERT.

the burden of maintaining energy as the importance of energy management surges [37].

Significantly, the interest in service migration has increased concerning mobility due to mobile device usage in recent years, and related studies are actively being conducted. Therefore, we investigate and analyze existing service migration models to strengthen the knowledge for future research.

2.2.1 MDP-based Migration Models

Traditional service migration models are primarily employed Markov Decision Process (MDP) [118]. In the MDP decision procedure (Figure 2.10), a mobile user has two action spaces: moving forward with a probability p and going back with a probability $(1 - p)$. Based on the probability, the MDP policy μ is defined as the tendency to take a

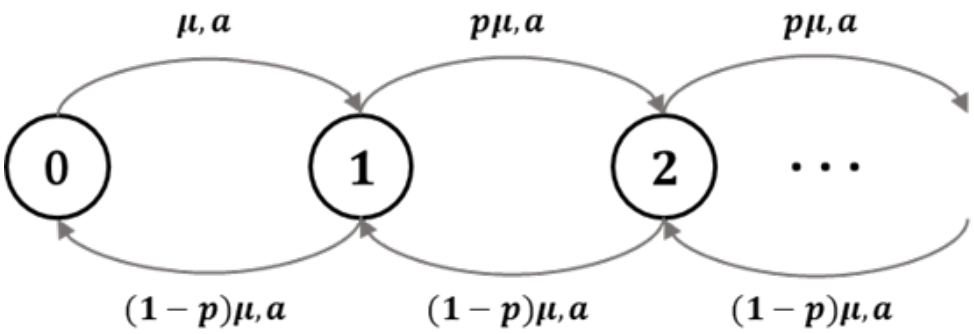


Figure 2.10: MDP model for service migration.

specific action a in a particular state s. Optimal decision-making policy is realized through immediate and expected future rewards given according to the selected actions.

The polynomial time-complexity algorithm was invented by S. Wang et al. [195] to define the optimal thresholds. The users followed a one-dimensional asymmetric random walk mobility model in the proposed model. They showed the optimal threshold policy for determining the optimal action based on MDP. Unlike previous works related to the one-dimension model, the distance-based MDP framework was proposed to compute a migration strategy for the $2-D$ mobility users efficiently in [196, 197]. They formulated the numerical cost function to indicate the migration and transition costs. Furthermore, the author conducted the simulation model with the real mobility traces of taxis in San Francisco to discover a real-world scenario.

2.2.2 Application of Various Optimization Algorithms

Many studies have attempted to apply diverse optimization algorithms and methodologies to optimize the service migration models. The authors in [179] introduced the PRofIt Maximization Avatar pLacement (PRIMAL) strategy to optimize the trade-off between the migration gain and cost in the cloudlet network. They employed an Avatar concept placed in a cloudlet to interact with each end-user. Besides, they adopted the Mixed-Integer

Quadratic Programming (MIQP) tool to solve the PRIMAL problem.

A. Nadembega et al. [141] presented a mobility-based services migration prediction (MSMP) model to achieve both the optimal micro data centers (MDCs) selection and the requested service adjustment. The suggested model adjusts the data transfer sequence between distinct MDCs and the user based on the user mobility and the data center's estimated capacity. It concerns three schemes: a data-transfer-throughput estimation scheme (DTT), a service-area-handoff-time estimation scheme (AHT), and a service migration management scheme (SMM).

A. Machen et al. [128] presented a live service migration framework to handle both the service down-time and the overall migration time. Three layers, such as the base, the application, and the instance layer, were contained in the generic layered migration structure. They separated the running and idle services into the application and instance layers and used incremental file synchronization to minimize service downtime.

The paper [146] explored the service performance optimization in the mobile edge regarding a long-term cost budget constraint, resulting in an online service placement framework to manage the performance-cost trade-off. The authors revamped the long-term budget into a real-time problem based on Lyapunov optimization to handle unanticipated user mobility. They controlled the designed issue by applying the Markov approximation. Furthermore, a distributed approximation scheme was offered based on the best response update technique to guarantee the proposed model's scalability and amenability.

2.2.3 Energy efficiency in MEC

In addition to works regarding the service migration cost, several studies related to energy efficiency issues in MEC have been performed. In the article [111], the authors investigated the correlation between migration distance and equipment transmit power. They established a dynamic service migration model through the migration strategy and the energy consumption anticipation derived based on the optimal stopping theory. Ultimately,

they demonstrated the effectiveness of the proposal in solving the migration path selection problem through intensive simulations.

Y. Yang et al. [207] used a Mixed Integer Linear Program (MINLP) to build the Green-Oriented Problem (GOP) formulation, which minimizes energy consumption on mobile devices. Their solution comprehensively considered the offloading decision, wireless communication resource allocation, and computational resource allocation. Moreover, they adopted the Q-learning method to find the optimal policy of the mathematical GOP model for the tasks migration problem.

The paper [199] focused on reducing the energy consumption of smart mobile devices, developing the fine-grained migration based on a generation algorithm (FGMBGA). They not only attempted to achieve the smooth execution of tasks within migrations but also enhanced the migration strategy by applying a fine-grained linear chain task decomposition, which classifies the target task into a set of subtasks. They also measured the task delay threshold and the energy consumption between the UEs and MECs. They achieved a migration strategy that dynamically responds to complex situations through these.

In [36], A. Boukerche et al. invented a task-centric offloading framework to manage the communication overhead and the offloading energy consumption, both incurred by numerous offloading requests. In addition to that, a cloud task-centric scheduling algorithm has been presented to increase the energy efficiency during the offloading process between Cloudlet and remote Cloud. The priority of local task interest was measured via frequent tracking of the task execution to arrange the offloading resources.

2.3 Conventional approaches to solve Service Migration

This section describes optimization algorithms and methodologies used in traditional service migration studies. We investigate structural optimization approaches, such as layering,

Table 2.1: SUMMARY OF RELATED WORKS

Problem Model	**Methodology**	**Target Issue**	**Evaluation Parameter**
Three Layer Model [128]	Layering	Active Service Migration	Migration time Down time
Threshold Policy -Based Mechanism [118]	MDP	Service Migration	Running Time Discounted Sum Cost
Distance Based MDP [195, 197]	MDP	Service Migration	Computation time Discounted Sum Cost
PRIMAL [179]	MIQP	Avatar Migration	E2e Delay Migration Overhead
MSMP [141]	DAMP	Service Migration	Data Latency
Mobility-Aware Online Framework [146]	Lyapunov Optimization Markov Approximation Best Response Update	Service Migration	User-perceived Latency Migration Cost
Reconfiguring Cloudlets [164]	PSO	Cloudlet Activation	Processing Delay Transmission Delay Backhaul Delay
DRL Based Model [97]	DQN	Service Migration	Migration Cost Communication Cost
DQN Based Model [218]	DQN	Service Migration	QoS, Migration Cost
DSM [111]	Optimal Stopping Theory	Migration Path Selection	Energy Consumption Migration Distance
MINLP [207]	Q-learning	Energy Minimization	Energy Consumption
FGMBGA [199]	Genetic Algorithm	Energy Minimization	Energy Consumption
ESM [147]	DQN	Service Migration	Migration Cost Transaction Cost Energy Consumption

as well as various algorithms to understand existing service migration studies comprehensively. In terms of optimization algorithms, although many conventional algorithms have been invented and utilized to find optimal solutions in research areas, we only concentrate on algorithms that were applied for service migration solutions.

2.3.1 Three-layered Migration Framework

In the process of migrating an application, the progress of the running application and related data are recorded in the in-memory state. In this process, the service downtime inevitably occurs when moving the recorded information and services. The authors of the paper [126, 128] devised a three-layered migration framework using layering to minimize service disruption (as shown in Figure 2.11). They modified the existing two-layer framework by separating the application layer into two layers: the instance and intermediate application layers.

The *base layer* consists of the guest OS, kernel, and other essential components, which are the foundation of running applications. All MECs have their own copied base layer to run on the migrated service. Hence, it is unnecessary to transfer this layer between MECs, which allows a service downtime to be decreased. The intermediate *application layer* is a layer for a deployed application and any data related to the application. When the migration of an application is required, the application layer may be preferentially migrated while the service is still running. The *instance layer* is involved in the running state of the application, which causes downtime of the service during it is migrated. The three-layered migration model can significantly reduce downtime due to service migration. In addition, the overall migration time can be reduced by checking whether the base and application layers exist in the target MEC and by transferring only the instance layer without moving all three layers.

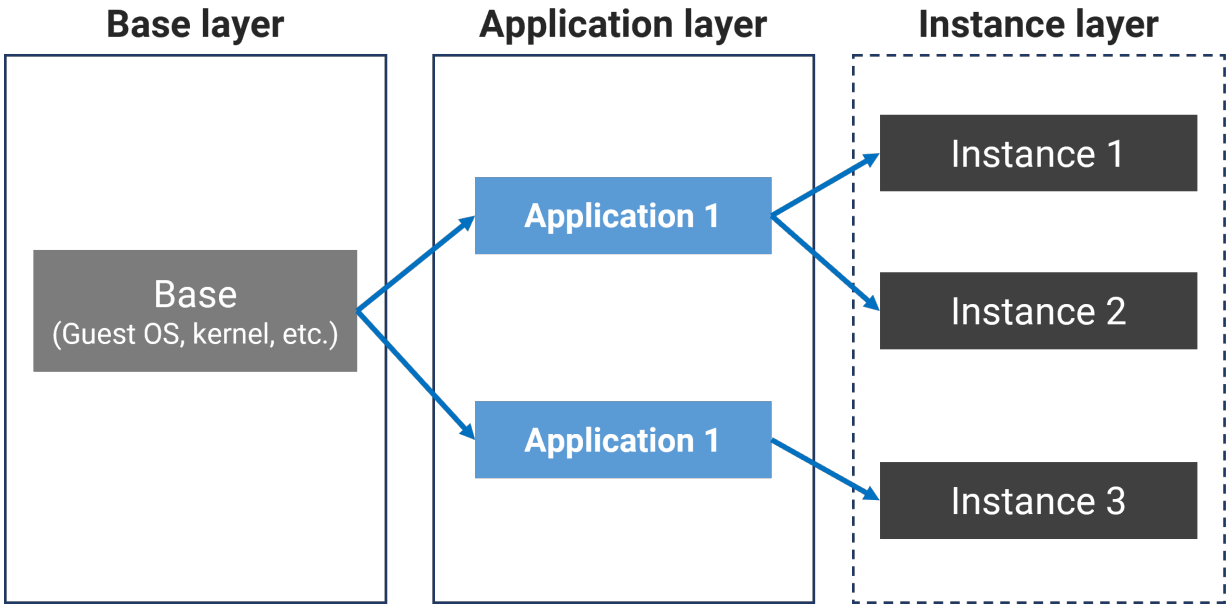


Figure 2.11: The three-layer model.

2.3.2 Destination And Mobility Path Prediction

A. Nadembega et al. [140] built a mobility prediction, Destination And Mobility Path Prediction (DAMP), to enlarge the network communication efficiency. The prediction model provides the destination and movement route of arbitrary users based on the user's moving history and information. DAMP has two processes: Destination Prediction Model (DPM) and Path Prediction Model (PPM), as shown in Figure 2.12 [141]. PM predicts a user's destination by using databases, which contain User Contextual (UC) knowledge, Navigation Map (NM), and User Frequently Visited Location Trace (UFVLT). Furthermore, BPM consists of selecting a road segment from among one or more road segments towards the destination based on prior knowledge of the destination obtained through DPM. The process starts at the current location and repeats until a pre-specified travel time or the destination is reached. At each road junction, PPM determines a set of road segments from among the road segments neighboring the current road junction, reducing the size of the set of adjacent road segments for the selection process. Then, the priority of road

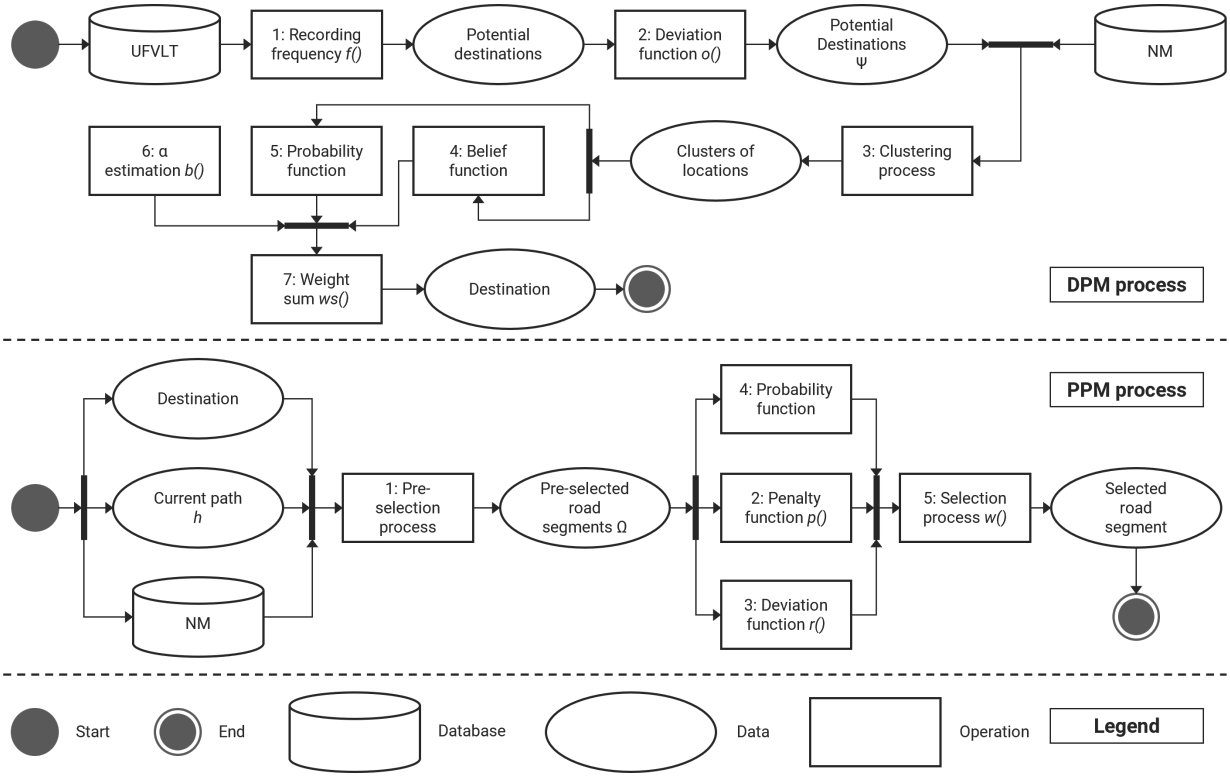


Figure 2.12: DPM and PPM processes.

segments is derived through Deviation, Penalty, and Probability functions.

In [141], the authors utilized the DAMP process to build the MSMP. DAMP was leveraged to predict the path of a given user in the model. Through DAMP, they obtained information related to the user's mobility, such as the frequently visited locations, the direction of the route, the contextual knowledge, the current trajectory, and the conceptual spatial maps. The proposed model predicted the optimal route to the final destination by iteratively selecting the most probable potential route among the subsequent road segments through DAPM. Historical and contextual knowledge of mobility contributed to enhancing the prediction's accuracy.

2.3.3 Mixed-Integer Quadratic Programming

MIQP is a mixed integer program in which the objective function is quadratic in the integer and the successive variables. Its constraints are linear in the variables of both integer and continuous variables [119]. The determined MIQP is in NP since a solution of polynomial size exists if the decision version of MIQP is feasible [152].

In [179], the authors formulated a MIQP problem to solve their PRIMAL strategy, which is a cloudlet network architecture to maintain lower end-to-end (E2E) delay for communications between each UE and its software clone in a cloudlet. In order to sustain low E2E delay of communications, it is necessary to monitor the movement of UEs and keep an optimal distance between UEs and Avatars. The authors used MIQP to optimize an objective function consisting of the gains and costs of the migrations.

H. Yao et al. [208] engaged a VM migration problem in roadside cloudlet to reduce VM migration as well as general network communication traffic. They structured a static off-line VM placement problem as MIQP and solved the complexity of the derived solution by using a heuristic algorithm, which is composed of two phases: a shortest path-finding algorithm based on link capacity constrained and a cost weighted-based VM placement algorithm. They derived their proposal by considering various constraints such as path selection, virtual machine placement, resource capacity constraints, link capacity, and network cost.

2.3.4 Optimal Stopping Theory

The optimal stopping theory is a statistical decision method to find the time to execute a specific action based on continuously given variables in order to obtain the optimal expected reward or cost. The optimal stopping problem is to repeatably determine when to stop a stochastic process among all possible candidate sets to maximize reward or utility. Representative optimal stopping problems include the Secretary Problem, the Sequential Probability Ratio Test, and the Two-Armed Bandit Problem. [187]

J. Hu et al. [111] presented a dynamic service migration strategy that minimizes the energy consumption resulting from service migration. They used the optimal stopping theory to obtain the expectation of the optimal migration energy consumption utilized in the proposed model. The optimal stopping rule is to choose the most appropriate moment to stop observation and take additional action established on a constantly monitored random variable.

In [203], the authors also applied the optimal stopping to achieve the maximum expected energy savings, which was an essential determinant of their strategy for the best cache replacement. The random variable distance sequence determined the energy consumption sequence. The energy-saving was sequentially checked by comparing the contents obtained from randomly selected nodes. The optimal node was selected according to the iterative comparison process. Optimal stopping determined the optimal stopping point during these repeated comparisons.

The authors of the paper [215] proposed a live virtual machine migration model to handle the continuity issue of cloud services in cloud-assisted vehicular networks. They developed the migration solution based on the optimal stopping framework. Furthermore, the one-step look-ahead approach was applied to optimize the organized optimal stopping problem. In [11], G. Best et al. established an optimal stopping formulation to allow a monitor vehicle to be located adjacent to an autonomous robot. Statistically predicting the autonomous robot's trajectory prevented the robot from going outside the tracker's surveillance range.

2.4 Learning-based solutions for Service Migration

This section overviews DL technology and describes some commonly applied deep learning algorithms that can be employed to solve complex service migration problems.

2.4.1 Deep Learning Overview

Recently, deep learning has become one of the most popular technologies, which has been applied in various fields of research in academia and industry. In this section, we overview deep learning technology regarding the fundamental principles and the details of Deep Neural Network (DNN).

In general, the advent of deep learning coincides with the beginning of the AI idea in the 1940s. After W. S. McCulloch and W. Pitts proposed the initial neuron network model [132], the concept was actively studied as F. Rosenblatt devised the perceptron capable of learning through a linear classification in neural networks [165]. However, as M. Minsky and S. Papert in [134] revealed the limitations of perceptron that nonlinear problems such as XOR cannot be solved, AI research interests had diminished dramatically. Although another AI winter period had come due to the problem of the gradient loss in error backpropagation method [166], Machine Learning (ML) is being actively used in various fields as diverse neural network models have developed and computing power has increased.

Classification of the AI technology.

Before discussing deep learning, we first describe the hierarchy and relationship of the associated technology. It is necessary to organize the root concepts, AI and ML, to investigate deep learning.

AI refers to the technology that allows computers to imitate the intelligent behavior of humans. While computers have significantly changed human society by enhancing computing power, AI technology has achieved innovative technological advances by creating the ability of computers to learn and make decisions without human participation. ML is an algorithm that gives computers the ability to learn without explicit programming and includes supervised learning, unsupervised learning, and reinforcement learning. ML algorithms generate rules or discover patterns to make better decisions or predictions, which

distinguishes ML from traditional cognitive algorithms. ML, which automatically learns from data, can be applied to complex problems that require much effort or have no solutions with conventional methods. Deep learning, a subordinate of ML methods, promises to recognize the future value of complicated optimization problems based on massive amounts of data and layered neural networks with numerous patterns. Deep learning can extract high-level features from data, which has contributed significantly to the application of machine learning in various studies. The deep learning model comprises a multi-layered neural network and can optimize numerous problems by combining layers and factors according to the circumstances. Figure 2.13 denotes the relation between AI, ML, and deep learning [137, 172, 217].

Type of Machine Learning

We can adapt different type of the ML technique amongst Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL) depending on the situations or the object problems. SL and UL can be classified by whether the answer is given. The

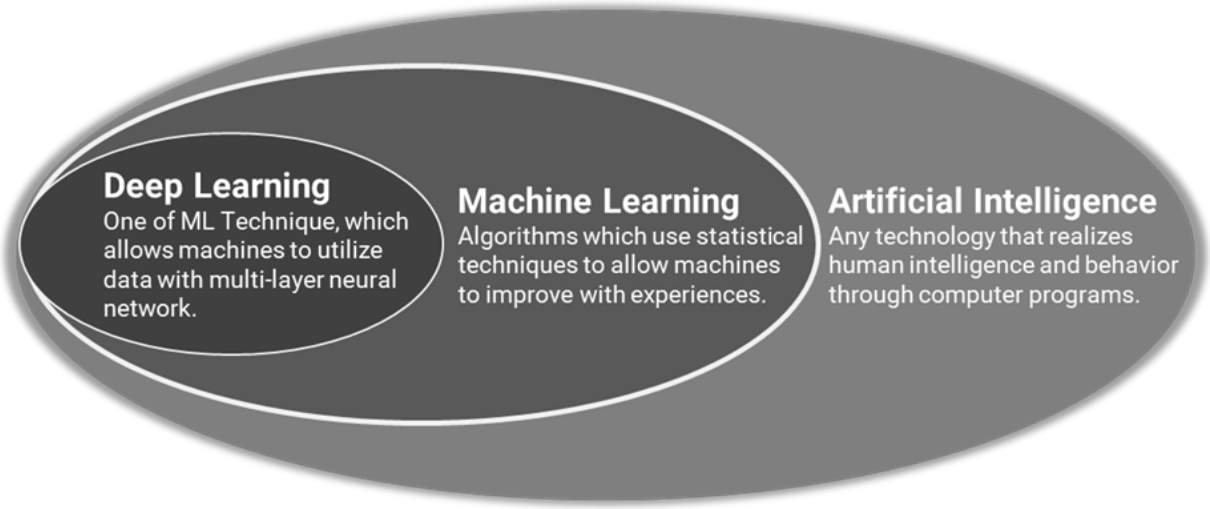


Figure 2.13: The relation between artificial intelligence, machine learning, and deep learning [137].

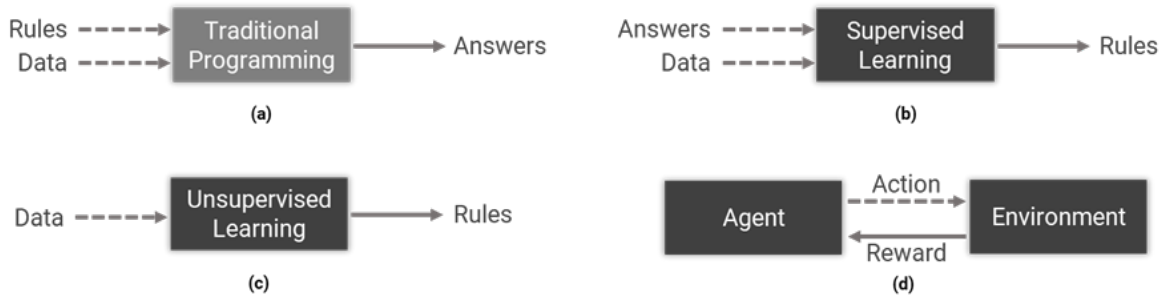


Figure 2.14: Diagrams of the learning procedures. (a) Traditional programming, (b) supervised learning, (c) unsupervised learning, and (d) reinforcement learning [137].

former aims to find out the relation between the given input and results, and in the latter case, it pursues to identify a pattern or rule in a state where only data is provided. The RL model reacting to a given state is entirely different from the previous two techniques. RL intends to choose an action according to the state of the surrounding environment without any given input and output value and maximize the reward accordingly. Figure 2.14 illustrates the procedure of the learning methods, and Figure 2.15 represents the hierarchy of ML algorithms.

Supervised Learning: SL is the most common data-driven predictive model. The model reduces errors between the predicted value and the given result value as the training proceeds, resulting in the function representing the relation of data. A refined dataset consisting of input and output values is supplied to train the model. The quality of the dataset is essential since it has a significant impact on the SL algorithm's learning process. Hence, high-quality data collection and efficient management must be preceded to implement an optimal SL model. SL has two categories: regression and classification [153].

The regression technique predicts continuous responses such as changes or fluctuations in future values. It estimates target values based on provided datasets by deriving a functional expression that can represent the distribution of the data. Regression algorithms include Linear Regression, Support Vector Regression, Gaussian Process Regression, and Ensemble Methods.

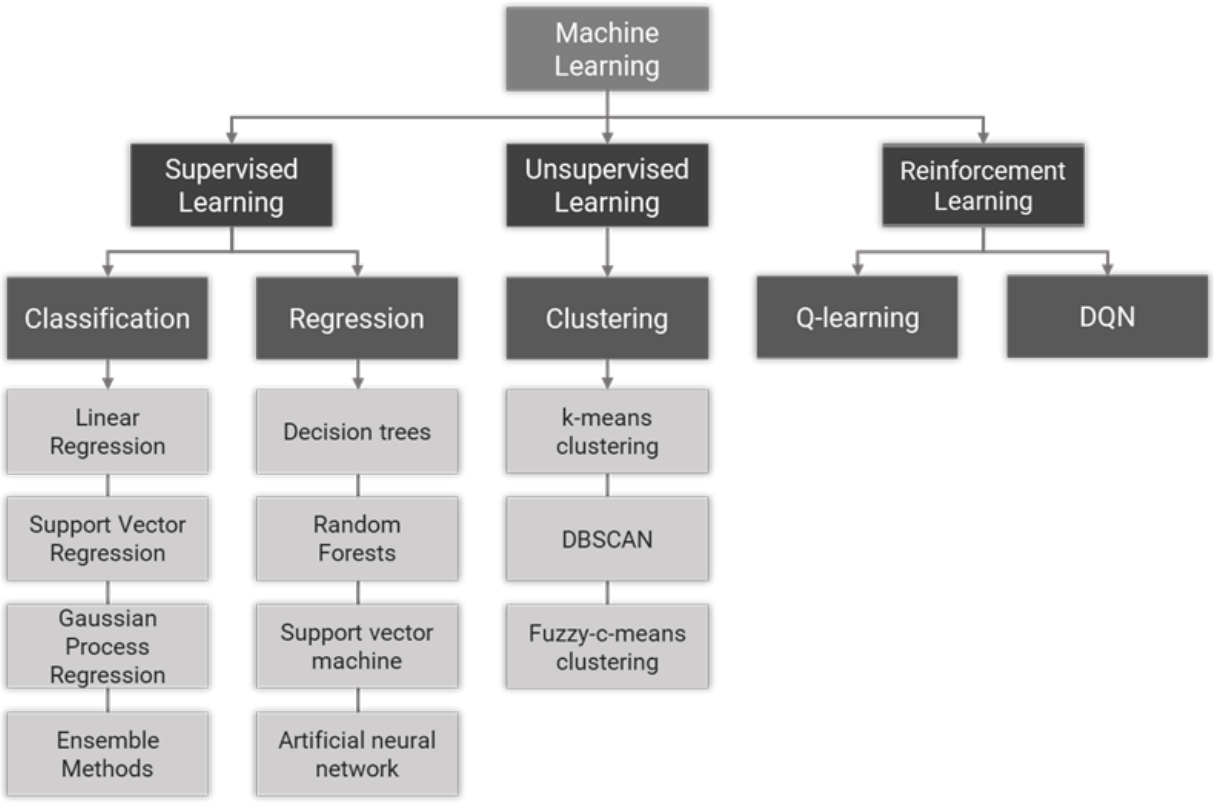


Figure 2.15: The hierarchy of ML algorithms.

In the case of classification, it can be applied when the data can be tagged, categorized, or classified into a specific group or class. The classification model is designed and trained to recognize the input data and generate rules for categorizing it. Decision trees [156], which can be visualized, are trained to reduce the uncertainty of each node. Random Forests generate multiple decision trees based on randomly assigning data with a single sample data. The algorithm improves accuracy by mitigating the overfitting problem of the Decision trees technique and ranks the importance of variables [69]. Support vector machine (SVM) is a technique that derives classification rules for a given sample group. It shows effective performance in the field of character recognition [77]. Artificial neural network (ANN) is a type of machine learning model that has become popular due to

the availability of the model for various areas, such as classification, clustering, pattern recognition, and prediction. ANN determines the optimal prediction through the neural networks on several layers connected to each node and the activation function [1].

Unsupervised Learning: Unsupervised learning is a learning method used to find hidden patterns or underlying structures from data without corresponding answers. Clustering analysis is the most general method to obtain hidden patterns or classifications in data through exploratory data analysis. There are several algorithms for performing clustering, such as k-means clustering, Density Based Spatial Clustering of Applications with Noise (DBSCAN), and Fuzzy-C-means clustering.

The k-means clustering classifies data items into k clusters, where k is randomly defined. The method starts setting k centroid points, which become reference points of the cluster groups. Even though the k-means algorithm is a simple way to define clusters, additional efforts are required to determine the k value. Due to the ambiguity of the k value, the deviation of outcomes is also non-negligible [114, 129].

DBSCAN is a density-based clustering algorithm that can distinguish unevenly shaped clusters from a large amount of data with noise and outliers. Unlike k-means clustering, in which groups use the distance between clusters, density-based clustering is a method of clustering high-density particles. DBSCAN has poor performance for datasets composed of clusters, which have different densities since it cannot set different distances ε and the minimum number of points for each cluster. In response to that, numerous enhanced DBSCAN algorithms tackle the density variation within the cluster [95, 157].

Fuzzy clustering is a type of clustering technique in which one data is allocated to more than one group, so fuzzy clustering is more reasonable than hard clustering. The Fuzzy-C-means clustering algorithm is a representative Fuzzy clustering algorithm. Fuzzy-C-means uses fuzzy partitioning, in which one data point is assigned to multiple groups based on a membership grade between 0 and 1. The membership value is determined by the distance between each cluster's center point and the data pointer. It is similar to the k-means algorithm, except that the allocation of clusters is replaced by the probability that

individual data points belong to each cluster. Besides, it has the same problem related to the initial parameters as the k-means algorithm because of its dependence on initial parameters [91, 151, 178].

Reinforcement Learning: In the RL algorithm, the agent continuously learns from the environment by trial-and-error learning. It tries to take the best action under the current state to maximize the reward value. The RL model can be designed as an MDP, which is composed of a set of states, a set of actions, and an immediate reward function. The interaction with a dynamic environment is the core factor in the RL model (shown in Figure 2.16). The agent not only exploits the stored values that are already experienced but also explores the future possible action to obtain a better reward [181].

An off-policy Temporal-Difference control algorithm, Q-learning, is a breakthrough approach in early RL [181]. The action-value function Q estimates the optimal action-value function for a given finite MDP. The relationship between the current state and action, and the next state and action is formed in the Bellman equation. The state-value function $v_\pi(s)$ for MDP is given by (3.1).

$$v_\pi(s) = \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s\Big], forall s \in S \tag{2.1}$$

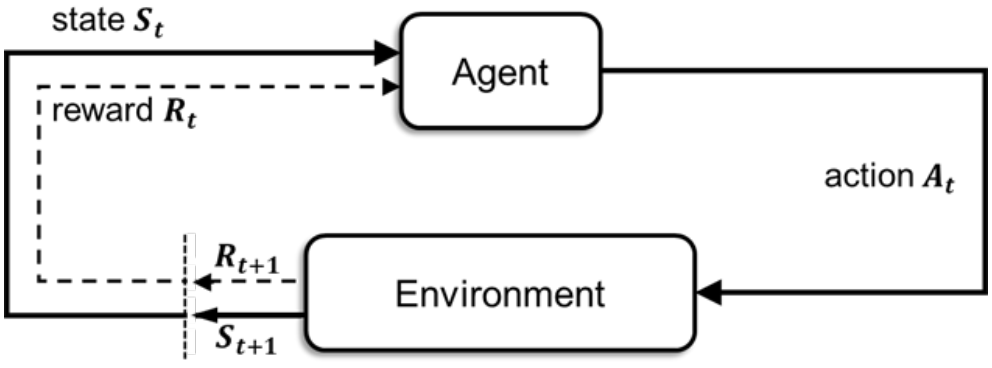


Figure 2.16: The agent–environment interaction in Reinforcement Learning.

where $\mathbb{E}_\pi[\cdot]$ means the expected value of a random variable given that the agent follows policy π, S_t is the current state, and R_t denotes the expected reward at timeslot t. The action-value function for policy π is represented as follows:

$$q_\pi(s,a) = \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\Big] \tag{2.2}$$

All Q-function values corresponding to states and actions are stored in the Q-table and used to determine possible actions for future states. The Q-learning algorithm is shown in algorithm 2.1.

Algorithm 2.1 Q-learning (off-policy TD control)

Algorithm parameters: learning rate $\alpha \in \{0, 1\}$, small $\epsilon > 0$

Initialize parameters
loop for each episode:
    Initialize state S
    while S is non-terminal **do**
        Choose A from S using policy derived from Q (e.g., $\epsilon-$ greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
        $S \leftarrow S'$
    end while
end loop

The Q-learning algorithm has restrictions in dealing with real-world problems due to the limitations of the Q-table. Deep Q-Network (DQN), which uses deep learning as an approximator to update the action-value function, is an excellent alternative to the problem of Q-learning. The algorithm improves the Q-Learning algorithm with supplementary techniques, the deep neural networks and the experience replay [135, 136].

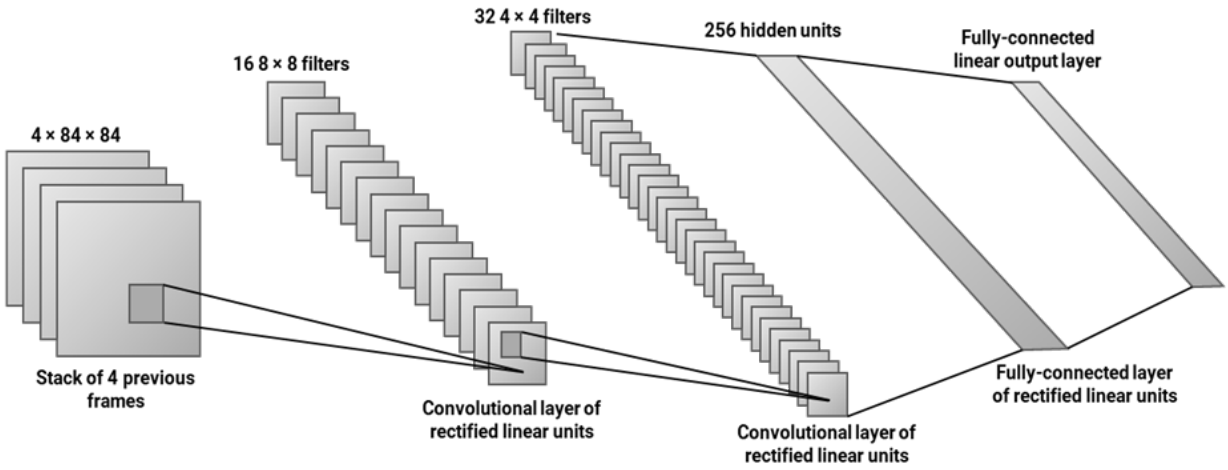


Figure 2.17: The architecture of the DQN model.

DQN was first introduced in the paper "Playing atari with deep reinforcement learning" [135]. The authors of the paper demonstrated that the agent of the DQN model could learn how to play a video game and achieve a great score without human intervention. They also exploited the experience replay, where the agent's experiences at each timeslot were stored in a replay memory. Based on the stored experiences, the agent takes action to exploit the experiences or explore a new action.

As shown in Figure 2.17, the proposed architecture has two convolutional layers to reduce the complexity of the input parameters and a fully connected layer for evaluating Q-values. By utilizing the neural network, the architecture can estimate Q-values through a single forward pass, taking into account all possible actions for a given state. Depending on the target problems, there are several ways to construct the architecture of the model with various neural network algorithms. Hence, the DQN method can provide flexible optimization solutions to complicated learning problems.

2.4.2 Deep Neural Network

As the usage of Deep Learning, which employs ANNs to mimic human thinking mechanisms, is prominent, various DNN models are being developed. The neural network algorithm has more hidden layers to build a more sophisticated decision model, and then the computational complexity increases significantly. Additionally, in some cases, the DNN model reached the incorrect optimal result instead of the actual optimal outcome. Various algorithms and techniques have been proposed to release the obstacles. Thus, we discuss not only the related DNN models, such as Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and LSTM but also the evolved DQN models, such as Double DQN and Dueling DQN.

Multi-Layer Perceptron

MLP improves the limitations of Perceptron with hidden layers so that DL can be used practically. In the MLP architecture, the first layer is the input layer connected to each node of the next hidden layer. The hidden layers consist of the weight values and biases and interact with the last output layer. Figure 2.18 shows the structure of MLP with two hidden layers.

MLP calculates an estimated value through a feed-forward process and back-propagates the error between the expected value and the actual value through the stochastic gradient descent (SGD) methods, such as Momentum, Nesterov accelerated gradient (NAG), Adaptive Gradient (AdaGrad), Root Mean Square Propagation (RMSProp), Adaptive Momentum Estimation (Adam), and AdaMax [115,180]. As a result, an optimization network is derived by adjusting the nodes' weight and bias values of each hidden layer.

MLP utilizes the activation function to define the output in the feed-forward step. The activation function is significant in preventing the gradient vanishing problem. Sigmoids function, Rectifier linear unit function (ReLU), and the hyperbolic tangent function (Tanh) are broadly applied as activation functions [158].

Convolutional Neural Network

CNN [5, 206] is a representative of DNN models for processing images or datasets with a grid pattern. CNN has become an overwhelming paradigm in computer vision tasks, achieving notable outcomes. In recent, it is, however, not limited to the field of image processing and has been used in various research fields. Besides, the algorithm is adopted as an approximator of DQN to reduce the complexity of the input parameters in the model. CNN is a complex structure network with two process steps, such as the convolutional and fully connected steps. The first phase extracts the flatten feature of the input tensor, while the second phase measures the expected value of the conveyed feature from the previous stage. Figure 2.19 is a whole framework of CNN.

The convolutional process is a core component of CNN, including kernel padding, ReLU

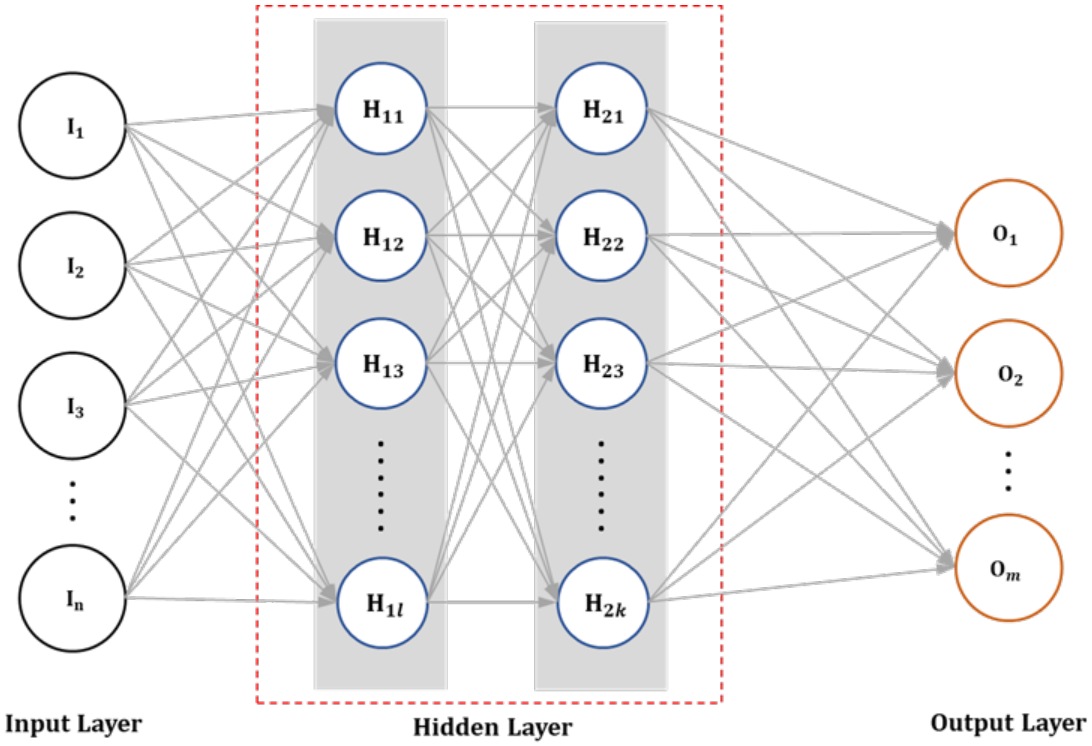


Figure 2.18: Multi-layer Perceptron structure.

activation, and the Max pooling. A small grid kernel is employed as a filter to extract the feature from input tensors, enabling an efficient cognitive process. Each value at the feature map is measured by an element-wise product of each kernel filter element and the input and a summation of all the product outputs. This capacity significantly diminishes the dimension and size of the input parameters to improve the efficiency of subsequent identification processes. The sample of the kernel padding calculation is described in Figure 2.20.

Another dominant segment of the convolutional sequence is the pooling layer, a down-sampling method to reduce the complexity of padding results. The most common type of pooling is a Max-pooling with a 2×2 fitter and a stride of 2. The padding result is separated into shooting-type sub-district, and only the maximum value of each region is extracted to form the pulling output, which is presented in Figure 2.21.

When it comes to the fully connected layer, every node in each layer is connected to all nodes at the next layer. The flattened feature map from the convolutional stage is regenerated into a one-dimensional array through the hidden dense layers and the last

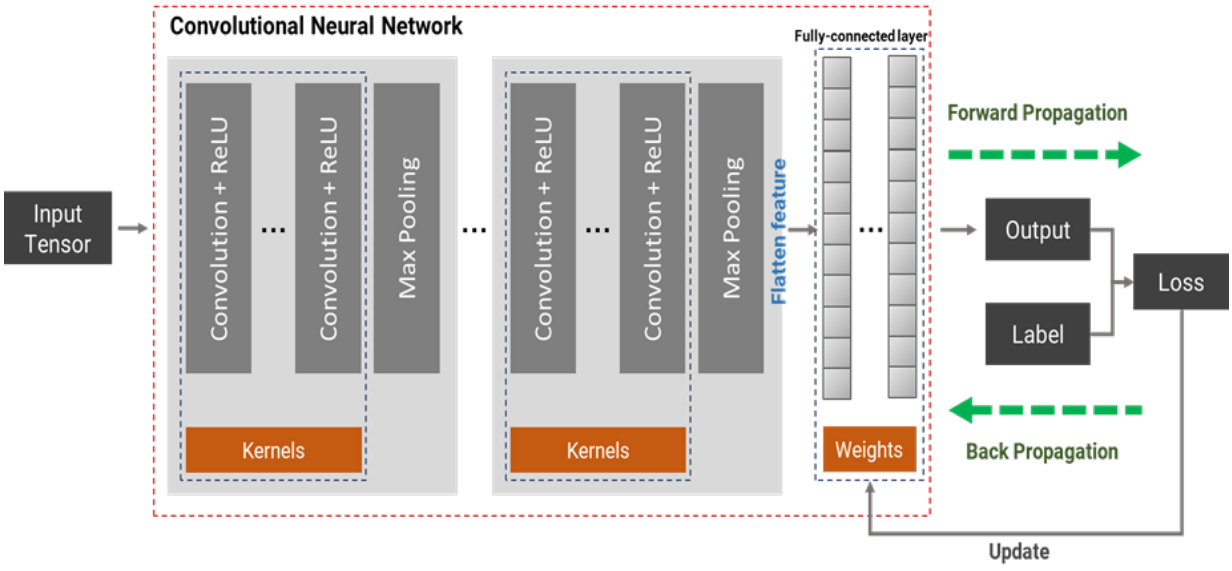


Figure 2.19: The overview of CNN framework and process.

activation function. The CNN model conducts the gradient descent with an error function through the forward propagation and the backpropagation to train the network. The kernel and weight values are adjusted by iteratively updating with the loss between the expected and the labeled values.

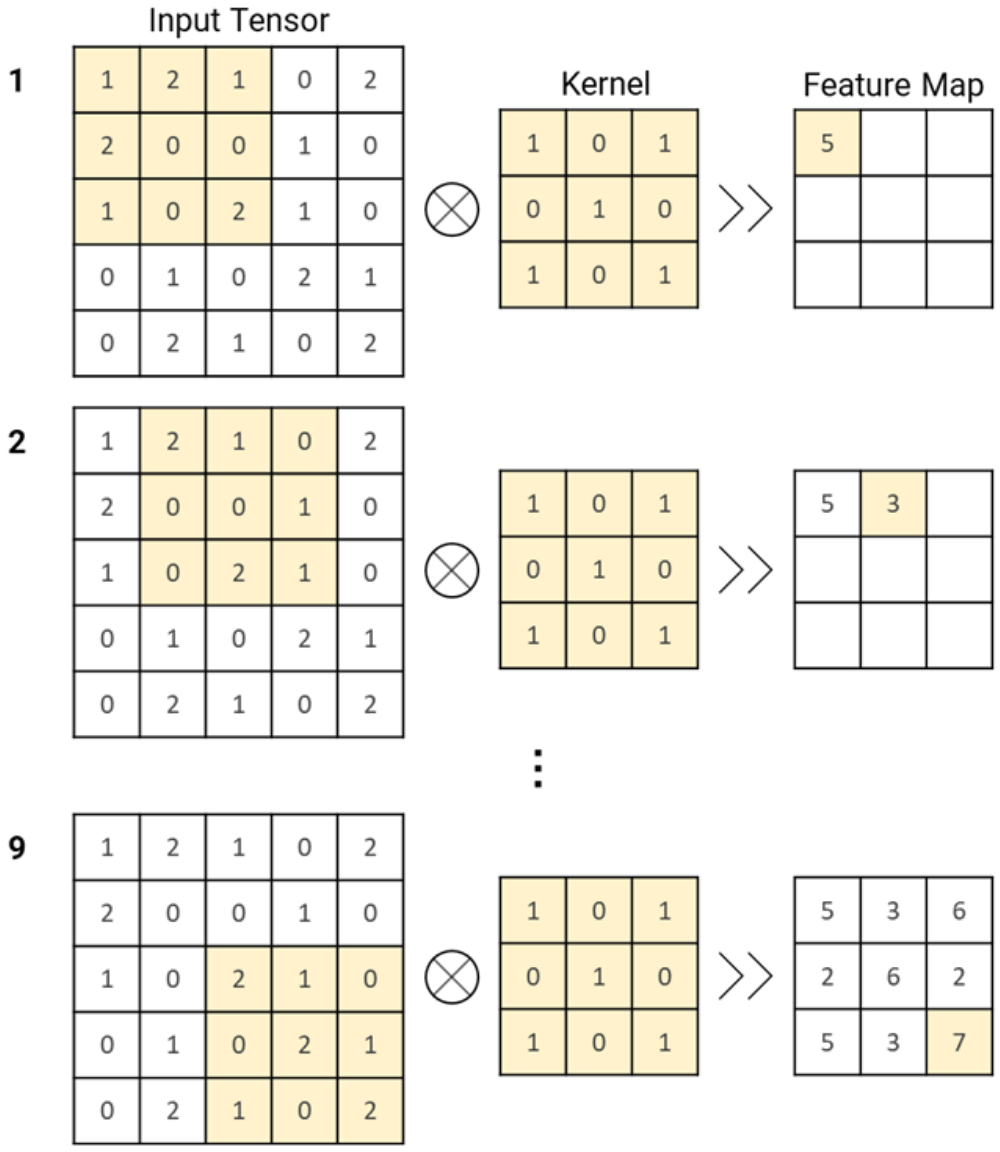


Figure 2.20: The sample of the kernel padding calculation.

Recurrent Neural Network

RNN [133, 142, 148] is a type of DNN that has strengths in processing sequence data or temporal data. The general DNN structure has a feed-forward neural network, whereas an RNN has a recurrent neural network structure, which transfers the output of the hidden node at the previous to the next node. RNN can efficiently estimate continuous data since the current node contains not only the information of the previous node but also the even earlier nodes' information. The RNN structure is displayed in Figure 2.22.

RNN applies a Backpropagation through time (BPTT) [138, 201], which considers both the current time step and the previous time steps when passing the gradient to the hidden layer weight. In the backpropagation process, BPTT has a problem: the computational complexity increases as the number of objective nodes increases [110]. In this case, Truncated BPTT [202], which utilizes BPTT by splitting the entire target nodes into a particular section, is an alternative.

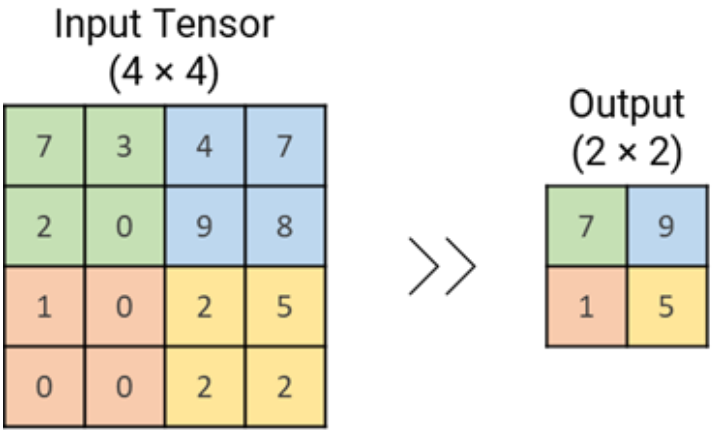


Figure 2.21: An example of max pooling operation.

Long Short-Term Memory

RNN has a severe drawback: as the number of nodes increases, a Long-Term Dependency issue occurs. In other words, as the number of steps for backpropagation increases, a gradient vanishing problem occurs, resulting in inefficient training performance. As a solution to this problem, S. Hochreiter and J. Schmidhuber introduced LSTM, a novel RNN-based model [110]. LSTM offers distinct concepts such as Memory Cells and three gates: Forget, Input, and Output. Figure 2.23 accounts for the recurrent architecture of LSTM.

Memory cell plays a crucial role in the LSTM algorithm to store hidden weight states so

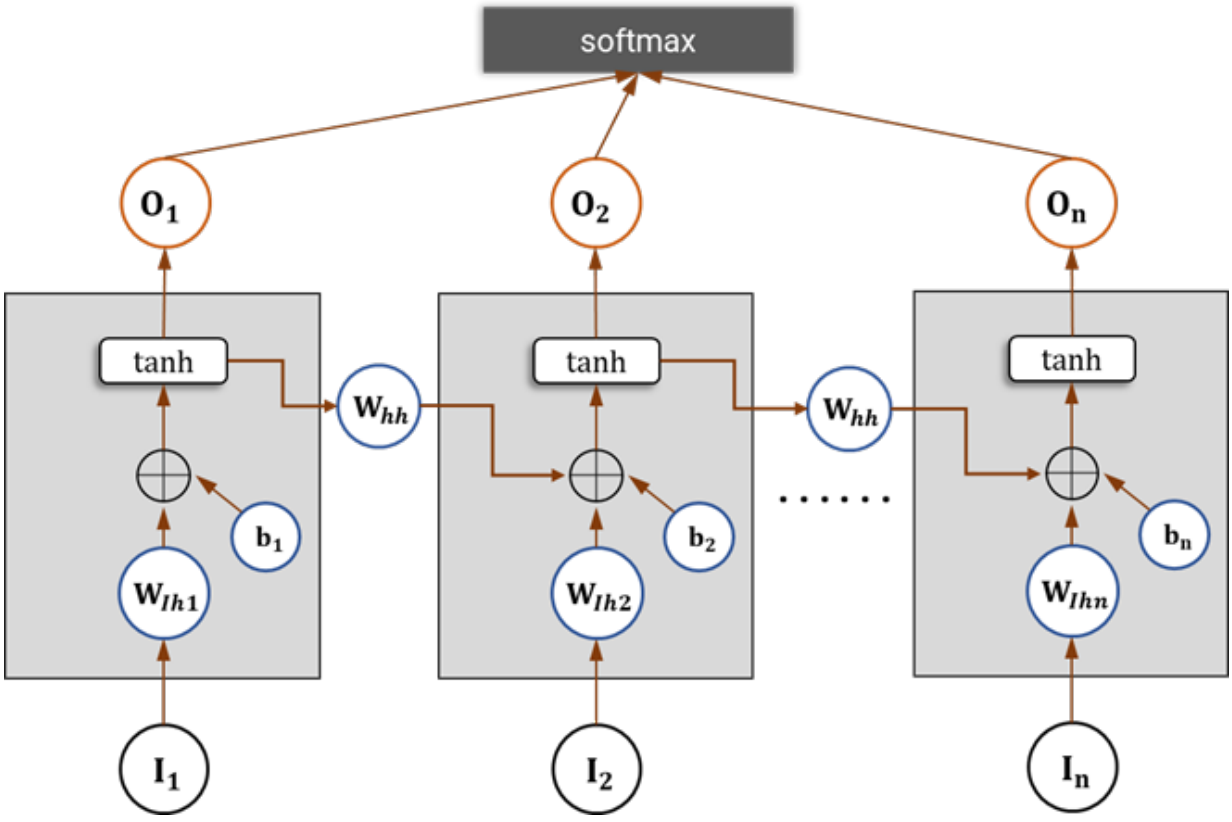


Figure 2.22: The Structure of RNN [142].

that the memory contents can be shared across the entire network nodes. The memory cell manages the state memory in conjunction with each gate and transfers it to the following node to mitigate the gradient descent problem [100, 110].

The Gate units are trained to make decisions related to the maintenance of certain information [100]. The first unit, the forget gate, performs an operation between the specific information transmitted from the preceding node and the current node's input and determines whether to store the result in the memory cell using the sigmoid function. In the next step, the input gate calculates new input-related information to reflect the memory cell. Two outcomes are generated by applying different weights and activation functions (Tanh and sigmoid function), respectively, based on the current input and the previous hidden state and are multiplied to append to the memory cell. Finally, the current block's

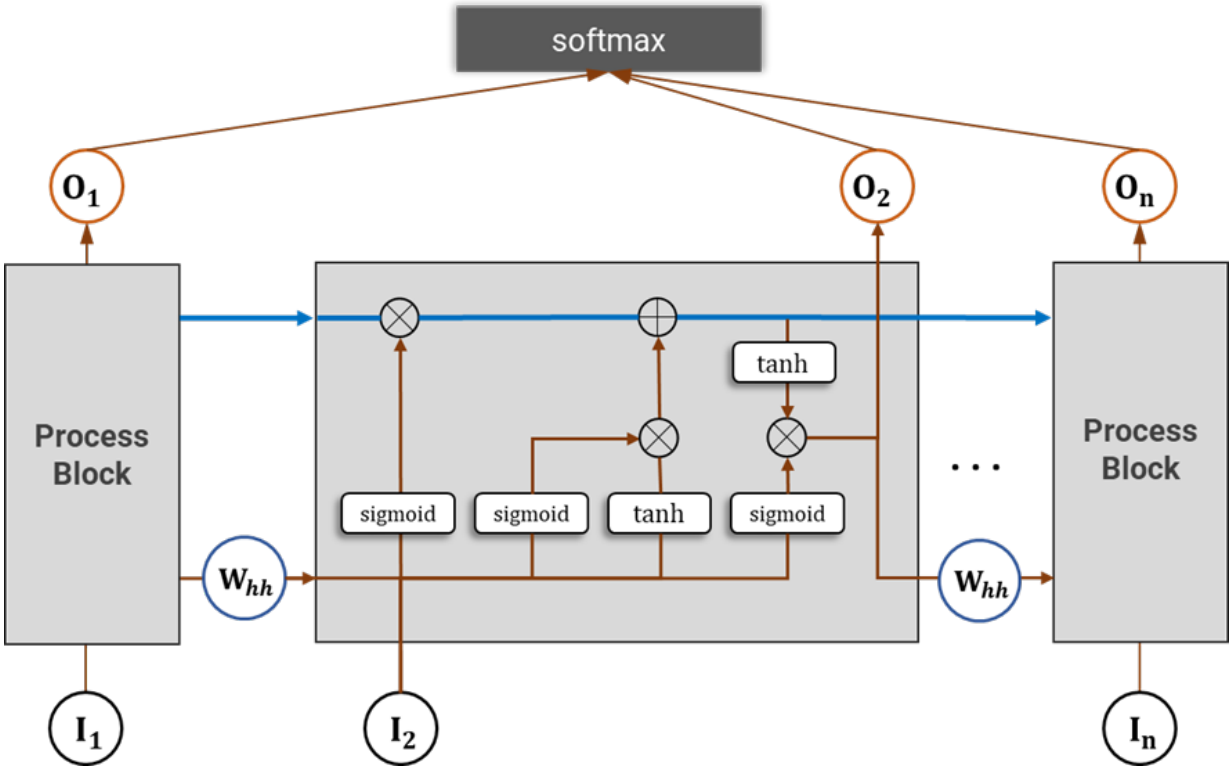


Figure 2.23: The recurrent architecture of LSTM [142].

output value, which is also the hidden state value transferred to the next, is formed at the Output Gate by multiplying the memory cell value applied Tanh and the new input value activated with the sigmoid function.

Enhanced DQN Models

As DQN, which has the capability to learn in response to the environment, is used in various fields, enhanced DQN algorithms have been proposed for better optimization efficiency or less computation time. Representative models are Double DQN and Dueling DQN [173].

In the general environment where the state space and size are enormous, the exploration frequency of DQN according to the epsilon-based algorithm decreases as the learning progresses, which derives local optimal and overestimated Q-values. An essential feature of the DQN model is the usage of a target Q-network and experience replay. The target network is the same as the online Q-network, and the online network is duplicated to the target network at specific steps without updating the loss every step. Accordingly, the action generated by the target Q-network even deteriorates the overestimation problem [173,186].

Double DQN is the same as the typical DQN [135, 136] in that it utilizes two Q-networks. However, it is different because it decomposes the max operation into the action selection and action evaluation [186]. Double Q-learning utilizes two Q-functions: the online Q-function and the target Q-function, denoted as Q^A and Q^B, respectively. Each Q-function uses a different set of experience samples from the other Q-function for the update operation. Even though both Q functions are updated with separate sets of experiences, those value functions are exploited to select an action. Algorithm 2.2 represents the detailed process of the Double Q-learning algorithm [107].

Both DQN and Double DQN have one deep learning phase with two Q-networks. On the other hand, Dueling DQN has two separated streams after the convolutional layer, the value function network for estimating the value of a given state and the advantage network to calculate a particular action (as shown in Figure 2.24). Eventually, the outcomes are produced by combining the results of both streams [173].

Algorithm 2.2 Double Q-learning

Initialize parameters
loop for each episode:
    Initialize state S
    Choose a, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe $r, s^{'}$
    Choose (e. g. random) either UPDATE(A) or UPDATE(B)
    if A **then**
        Define $a^* = arg\max_a Q^A(s^{'}, a)$
        $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)[r + \gamma Q^B(S^{'}, a^*) - Q^A(s,a)]$
    else:
        Define $b^* = arg\max_a Q^B(s^{'}, a)$
        $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)[r + \gamma Q^A(S^{'}, b^*) - Q^B(s,a)]$
    end if
    $S \leftarrow S^{'}$
end loop

Dueling DQN can recognize the state value, excluding the effect of each action on the state, which allows the agent to promptly define an appropriate action in a situation where the action is not affected, or similar actions are duplicated. In the last aggregation layer, simply summing the value network estimation and the advantage network value causes a lack of identifiability. Consequently, the stability of optimization can be improved by subtracting the average advantage of all actions possible, as shown in Equation 3.3 [200].

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + [A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a^{'}} A(s,a^{'};\theta,\alpha)] \tag{2.3}$$

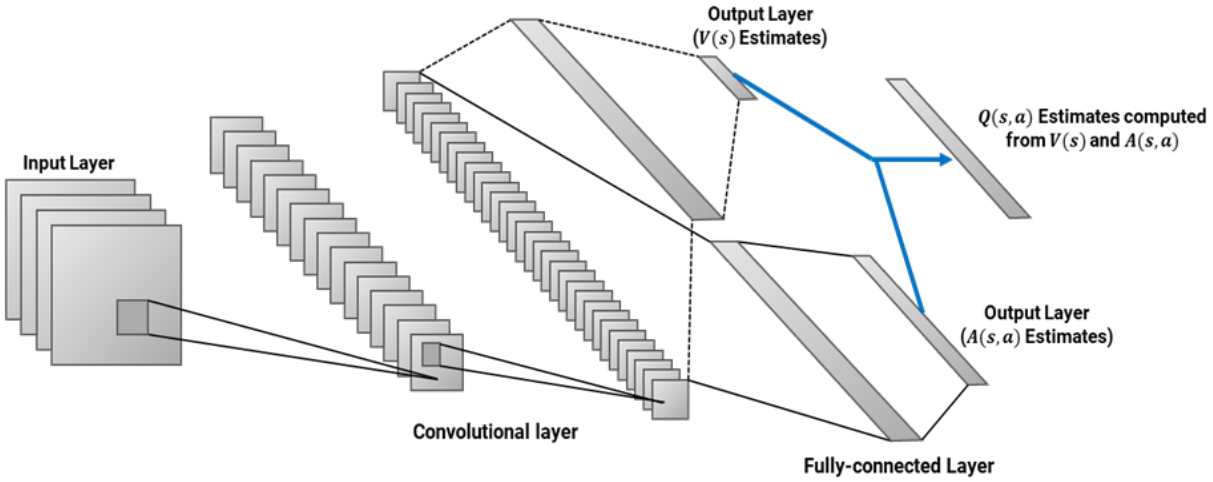


Figure 2.24: The architecture of Dueling Deep Q-Network.

2.4.3 AI applications for Service Migration

The progress of AI technology has brought substantial changes in optimization studies. Along with this vast advancement, many attempts have been applied to service migration problems in MEC. T. G. Rodrigues et al. [164] utilized a configuration phase to control the service delay elements, devising an analytical service delay model in MEC. They employed the Particle Swarm Optimization (PSO) algorithm to analyze the behavior of the fitness function at each Configuration Phase of the Model. The maximized scalability, derived accordingly, allows the model to reduce the process iterations and particles.

RL also played as an optimal solution for dealing with service migration problems. Z. Gao et al. in [97] developed a service migration model based on Q-learning and DQN to handle the complex environment that was affected by various constraints, such as path selection, VM placement, resource capacity, and link capacity. They took into consideration the necessary elements for the model, such as state, action, and reward, regarding the migration and communication costs. Similarly, a DRL-based method was devised by C. Zhang and Z. Zheng in [218] to cope with UEs with high-level mobility patterns. They organized a DQN model to enable the FMC controller to generate the optimal policy from

past experiences. The authors also formulated the reward function as a trade-off between QoS and the migration cost. Unlike MDP, the delivered task migration algorithm required no transition probability.

In our previous research [147], we have demonstrated the possibility of implementing realistic service migration scenarios using several environmental variables. Furthermore, we developed an extensive service migration model that responds to the environment based on DRL. It can respond more sophisticatedly to different UE mobility patterns by recognizing energy consumption as well as migration and communication costs. The advanced DQN algorithm also reduced high computational complexity with the multi-layered structure. Through simulations based on various conditions, the influence of the proposed model according to the situation change was additionally measured and analyzed.

Chapter 3

A novel Deep Reinforcement Learning based service migration model

In this chapter, we proposed an Extensive Service Migration (ESM) [147] model based on DRL to cope with the complex service migration environment in MEC. Compared to the existing models, the proposed model enables the controller to manage the migration process with a comprehensive perspective, which considers more environmental factors, including costs of both migration and transaction, and energy consumption. Although it brings computation complexity, we can implement the realistic simulation by utilizing more variables and the increased number of MEC servers.

3.1 System Model

We consider a time-slotted model, as $t \in \mathcal{T} = \{1, \ldots, T\}$, based on the FMC concepts, which has the advantages of applying the proposed agent control model since the FMC controller exists for managing distributed Data Center instances (i.e., MEC servers). The

FMC controller decides whether, when, and where to migrate the services to enable UEs to satisfy QoS. The MECs locations are denoted as $l_i \in \mathcal{L} = \{1, \ldots, L\}$ that the locations in $\mathcal{L}$ are 2-dimension vectors and the distance between l_1 and l_2 is calculated as the norm value of them, such as $\|l_1 - l_2\|$. Also, we define the user as $u(t)$ and the service locations as $h(t)$ at the timeslot t, respectively.

3.1.1 The migration and the transaction Cost function

We assume that the whole service is migrated when the action occurs at each timeslot, and we utilize the constant-plus-exponential cost functions from [197] and [196]. The function involves both the migration and transition cost, and its usefulness as an approximator was experimentally demonstrated. Note that state $s(t)$ represents the distance between the user and the MEC server at timeslot t, such that $s(t) = \big\|u(t) - h(t)\big\|$. The state after the action $a(s(t))$ is denoted as $s'(t)$, then $h'(t)$ is the new service location, where $\big\|h(t) - h'(t)\big\| = |s(t) - s'(t)|$, and $s'(t) = \big\|u(t) - h'(t)\big\|$. The total costs consist of the migration cost, the transition cost both between the connected MEC and the service located MEC and between the user and the connected MEC, which are denoted as $c_m(x)$, $c_{d_1}(x)$ and $c_{d_2}(x)$ respectively; the expressions are as follows:

$$c_m(x) = \begin{cases} 0, & x = 0 \\ \beta_c + \beta_l \mu^x, & x > 0 \end{cases} \tag{3.1}$$

$$c_{d_1}(x), c_{d_2}(x) = \begin{cases} 0, & x = 0 \\ \delta_c + \delta_l \theta^x, & x > 0 \end{cases} \tag{3.2}$$

Where β_c, β_l, δ_c, δ_l, μ, and θ are real-valued parameters, $x = |s(t) - s'(t)|$ for $c_m(x)$ and $c_{d_1}(x)$, and $x = s'(t)$ for $c_{d_2}(x)$. The parameters μ and θ set the weight of each distance to the costs, and their values can be influenced with the network topology and routing mechanism of the network. Besides, the parameters β_l and δ_l control the costs

proportionally. The expression of the one-timeslot cost is given by (3.3).

$$c_a(s(t)) = c_m(|s(t) - s^{'}(t)|) + c_{d_1}(|s(t) - s'(t)|) + c_{d_2}(s'(t)) \tag{3.3}$$

3.1.2 Energy Consumption function

We consider energy consumption on servers related to the migration between MEC servers. Even though it is not a significant problem since the server has always been plugging, we can use it as a control criterion to cope with the frequent migrations. We adopt the dynamic migration energy consumption method from [111] as the energy consumption formula for the service migration between MEC servers, which is given by (3.4).

$$E_a(S_n(t)) = \frac{\tau C \cdot \sigma(2^{R/W} - 1) \cdot (|s(t) - s^{'}(t)|^2)}{R \cdot \lambda^2} \tag{3.4}$$

where C, σ, R, W, and λ mean the service content size, the noise power, the transmission rate, the bandwidth, and the channel attenuation coefficient, respectively. $\tau \in [0\%, 100\%]$ is cited from the degree of service unexecuted service [127], which accounts for the running state of the service to be migrated.

3.2 PROBLEM SOLUTION

Based on the service migration model mentioned above, we formulate the DQN algorithm, which includes the Q-learning function and the DQN algorithm. The RL method allows the agent to deliver better estimations for the Q-value. The target problem of Q-learning is expressed in the MDP. In the RL algorithm, the agent can make a particular action when it encounters a specific environmental situation. The action results in a reward and a new state, where the agent can perform another action. With the recursive process, the agent obtains the optimal policy by determining the action, which maximizes the cumulative rewards (shown as Figure 3.1) [181].

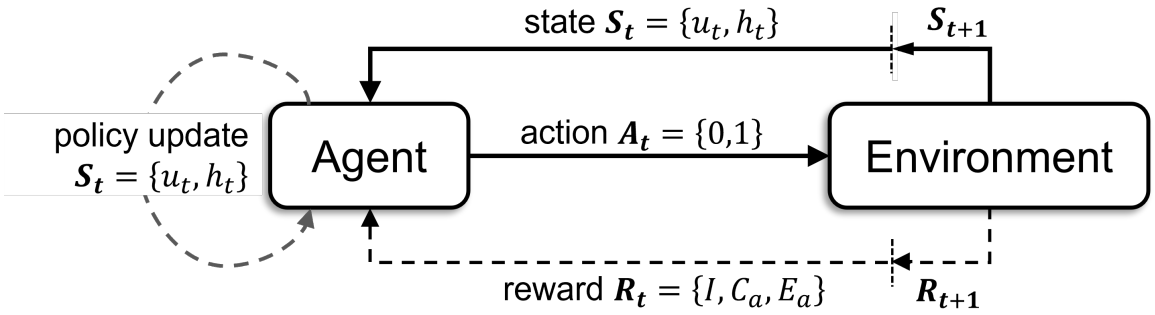


Figure 3.1: The agent–environment interaction with single user.

We define the necessary factors (state, action, and reward) as follows:

- **State:** We define the state as the distance between the UE and the MEC server in which the service is located. The state is denoted as $s(t) = \|u(t) - h(t)\|$ at timeslot t.

- **Action:** The agent can move the state $s(t)$ to the possible $s'(t)$ as taking the action $a(s(t))$. The action consists of the action set $\mathcal{A} = \{0, 1\}$, which means whether to migrate or not.

- **Reward:** The agent obtains the reward according to the action and the state. In this paper, we organize the object function as the cost and energy consumption. Aside from this, we define the reward function as the differentiation between an adequate large QoS value as I and the object function. With the equations (3.3) and (3.4), it is designed as follows:

$$R_a(s(t)) = I - [C_a(s(t)) + \rho \cdot Ea(s(t))] \tag{3.5}$$

  where, ρ is a coordination factor to adjust the scale difference between the cost and the energy consumption values.

3.2.1 Q-value function

The Q-value function represents the maximum value of the discounted total future rewards when the action a is taken while the system is at state s. As expressed in the equation (3.6), according to the Bellman equation, the Q-value function is the sum of the instantaneous reward obtained from executing action a, which moves the system from the current state s to the next state s', and the maximum expected future reward for being in the state s'.

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a') \tag{3.6}$$

Given the Q-value function, the agent chooses an optimal action for a given state according to the policy, which is represented as follows:

$$\pi(s) = \arg\max_{a} Q(s,a) \tag{3.7}$$

Q-learning can derive the optimal policy by repeatedly calculating the Q-value function and updating the network's weight vectors. The update procedure of Q-learning is simply expressed as below:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma \max_{a} Q(S_{t+1},a) - Q(S_t,A_t)] \tag{3.8}$$

3.2.2 Deep Q-network

In practice, applying the Q-value function is unrealistic because of the input size, i.e., the number of states. In other words, increasing the targeted states requires more computation time and power to find the optimal value by tracking all the situations caused by possible actions in a specific state; besides, it is impossible to complete. Hence, it is necessary to implement the approximation for estimating the Q-value [135]. We utilize the DQN, which includes the Deep Neural Network (DNN) stage as the approximation as well as

implements with the experience replay method. We take the states as input parameters and conduct the feed-forward and the back-propagation processes to obtain corresponding output Q-values (shown as Figure 3.2)

By utilizing the experience replay, we can prevent driving the DNN into the local minimum or diverging unexpectedly. We also adopt the off-policy control to separate the policy update into the behavior policy update and the target policy update. It allows the model to discover the optimal policy while following exploratory policy. During the process,

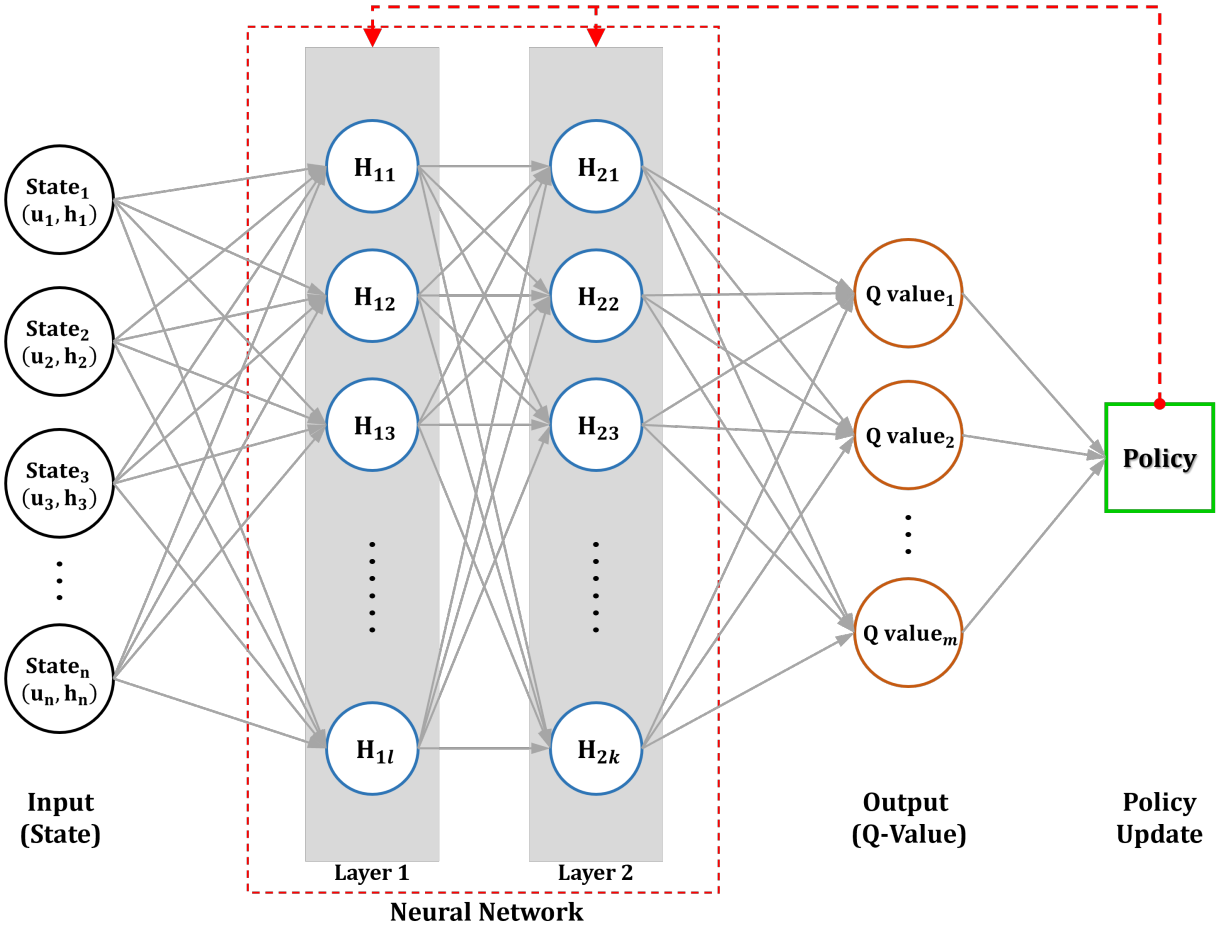


Figure 3.2: Architecture of deep Q-network.

samples of experience are drawn uniformly at random from the pool $\mathcal{D}_t = \{e_1, \ldots, e_t\}$ of stored samples, which are the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ at each step t. By conducting the update process with the loss function in Equation (3.9), we achieve the optimal policy for the service migration [136].

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-)\right) - Q(s, a; \theta_i)\right]^2 \tag{3.9}$$

Algorithm 3.1 represents the entire phases of the DQN process. The process begins with initializing the replay memory and the action Q-Network with random weight θ. The target Q-Network, which is a separate network to restrict leading to faltering and divergence of the policy, is also initialized as the same value as θ. According to an ϵ-greedy policy, actions are selected and executed during the learning loop, resulting in the reward and the next state. The happened experiences at each step are held in the dataset D, which is randomly used for Q-function value update. This technique allows us not only to acquire better data efficiency but also to obstacle correlations between the stored samples.

3.3 Performance Evaluation

In this section, we discuss numerical analysis to demonstrate the effectiveness of the proposed model. For the simulation, we assume that the user follows the random walk mobility model and MEC Servers are randomly deployed in a 100×100 grid area to implement the generic simulation environment. To validate the improvement of the proposal, we compare the model with other algorithms: No migration, Always migration, and Q-learning algorithm (as discussed later in Performance of Model analysis).

3.3.1 Parameter setting

The details of the parameter settings are provided in Table 3.1. The constant variables for the reward function are adopted from [196,197] and the energy function values are adopted

Algorithm 3.1 Deep Q-learning with experience replay

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

for *episode* $= 1, M$ **do**

  Initialize sequence $s_1 = \{u_1, h_1\}$

  preprocess sequence $\emptyset_1 = \emptyset(s_1)$

  for $t = 1, T$ **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \arg\max_a Q(\emptyset(s_t), a; \theta)$

    Execute action a_t in simulator

    Set $s_{t+1} = s_t, a_t, \{u_{t+1}, h_{t+1}\}$

    Preprocess $\emptyset_{t+1} = \emptyset(s_{t+1})$

    Store transition $(\emptyset_t, a_t, r_t, \emptyset_{t+1})$ in D

    Sample random mini-batch of transitions $(\emptyset_j, a_j, r_j, \emptyset_{j+1})$ from D

    Set $y_i = \begin{cases} r_j, \text{ if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\emptyset_{j+1}, a'; \theta^{-}), \text{ otherwise} \end{cases}$

    Perform a gradient descent step on $(y_j - Q(\emptyset_j, a_j; \theta))^2$ with respect to the network parameters θ

    Every C step reset $\hat{Q} = Q$

  end for

end for

from [111, 204, 207], which are logically and experimentally confirmed. We set $\delta_c = -0.4$, $\delta_l = 0.4$, and $\theta = 1.03$ for the transaction cost function parameters to capture the impacts of the variations in the distances between the user and the service located MEC server.

Table 3.1: SIMULATION PARAMETER VALUES

Parameters	**Description**	**Value**
$\delta_c, \delta_l, \mu, \theta$	Constant values/ coefficient related to the constant-plus-exponential cost function	$-0.4, 0.4, 0.8, 1.03$
ϕ	Coordination factor	1×10^{-3}
C	Service content size (KB)	$2 \sim 20$
σ	Noise power	3
R	Data transmission rate (KB)	2
W	Bandwidth (MHz)	100
λ	Channel attenuation coefficient	0.2

The cost function configuration samples presented in [197] are adopted to approximate our transaction cost function appropriately. In regard to the migration cost, we consider $\mu = 0.8$, $-\beta_l \in [0.0, 2.0]$, and $\beta_c + \beta_l = 1$.

3.3.2 Performance of Model analysis

In the experiments, we train the proposed ESM with random single-user movement patterns to achieve the minimization of the total sum of the cost and the energy consumption related to the migration process. The trained model can select the appropriate action with the environment state according to the optimal policy. Under the no migration algorithm, the user starts and ends with the service on the same MEC sever such that the agent never lets the service migrate. The reward value of it is only affected by the transaction costs related to the distances between the user, the service located MEC, and the connected

MEC. When it comes to the Always migration algorithm, the agent allows the service to move to the closest MEC server as the user moves around the area. As a result, it can reduce transaction costs but increase migration costs and energy consumption. In terms of the Q-learning algorithm, the agent follows the policy based on Q-function and Q-table. It repeatedly estimates Q-value and updates the Q-table to find the optimal policy. The reward values based on the algorithm are close to the ESM results; however, it has a limitation on the number of states since it uses the Q-table that has restricted index values. Namely, it is unsuitable for the real circumstance, which has more state conditions or larger areas.

Figure 3.3 shows the total reward sums of the experimental group with the variation in the number of MEC servers. Our proposed migration model can achieve the highest reward sum value with each number of the MEC servers when comparing the others. We cannot observe a constant change trend in the results because we conducted the experiments using the fixed area. As the number of servers increases in the fixed area, the density of servers increases, so the distance between servers tends to decrease, which reduces migration costs and transaction costs related to distances while increasing the frequency of service migration. On the other hand, reducing the number of servers grows the distance between each server but reduces the number of migration targets, causing diminishing migration cost and energy consumption. Due to the correlation of these inclinations, the reward sum values do not present any tendency regardless of the number of servers. The results indicate that the policy generated by the ESM model can appropriately choose optimal actions by interacting with the occurred states.

3.3.3 Evaluation for Impact of variables

In the second experiment, we consider the variation of the total reward sum when the migration cost parameter $-\beta_l$ changes from 0 to 2 under the fixed number of MEC as 30 and the content size of 20. The migration cost function $c_m(x)$ follows the exponential function curve since it has parameters that satisfy $\beta_c + \beta_l = 1$. Therefore, as $-\beta_l$ increases

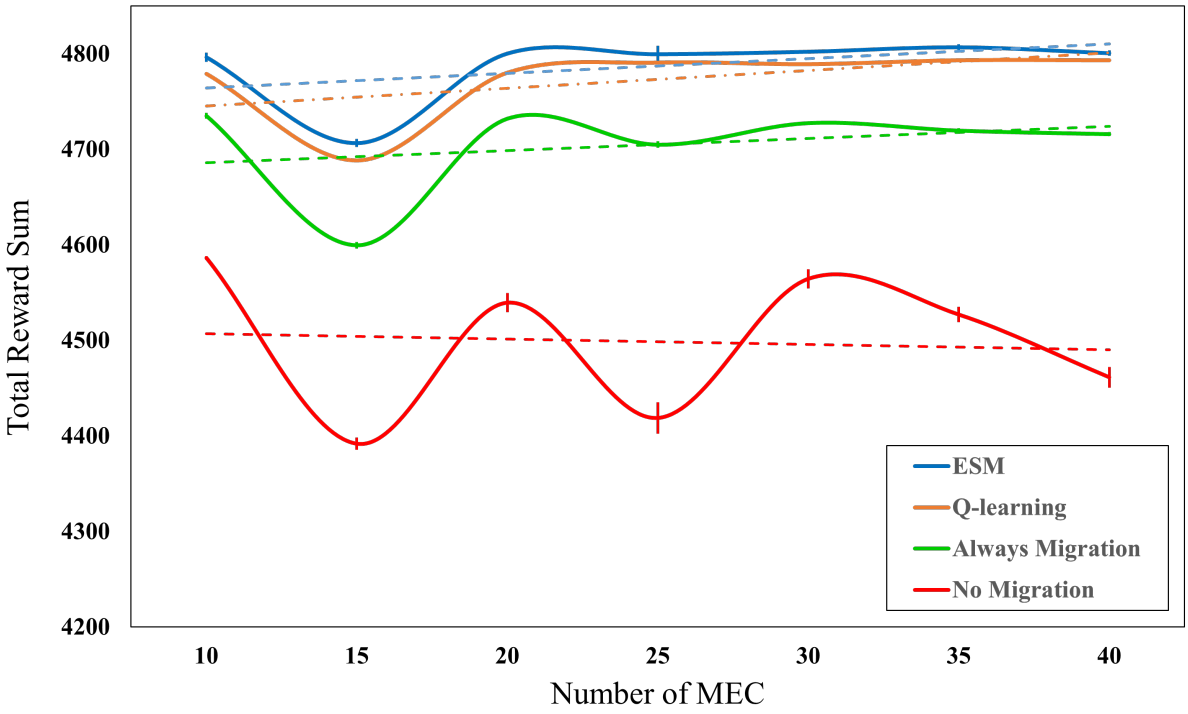


Figure 3.3: Total reward sum regarding the number of MECs when $\beta_l = 1$, and $C = 20$.

from 0 to 2, the results are gradually raised and eventually converge to specific values.

As shown in Figure 3.4, the results from ESM, Q-learning, and Always Migration, which involve the service migration, present a slight decrease according to $-\beta_l$ value is increased. In contrast, No Migration displays no fluctuation because it has no impact on migration. Although they also have trivial shifts in the result according to the $-\beta_l$ value, ESM and Q-learning obtain the large gaps with the other two algorithms on the reward sum value.

A similar result is observed in the third experiment set, which examines the influence of the content size on the reward values (shown as Figure 3.5). With a low content size, ESM, Q-learning, and Always Migration can achieve better rewards regarding migration energy consumption. In particular, Always Migration seems to be more affected by the

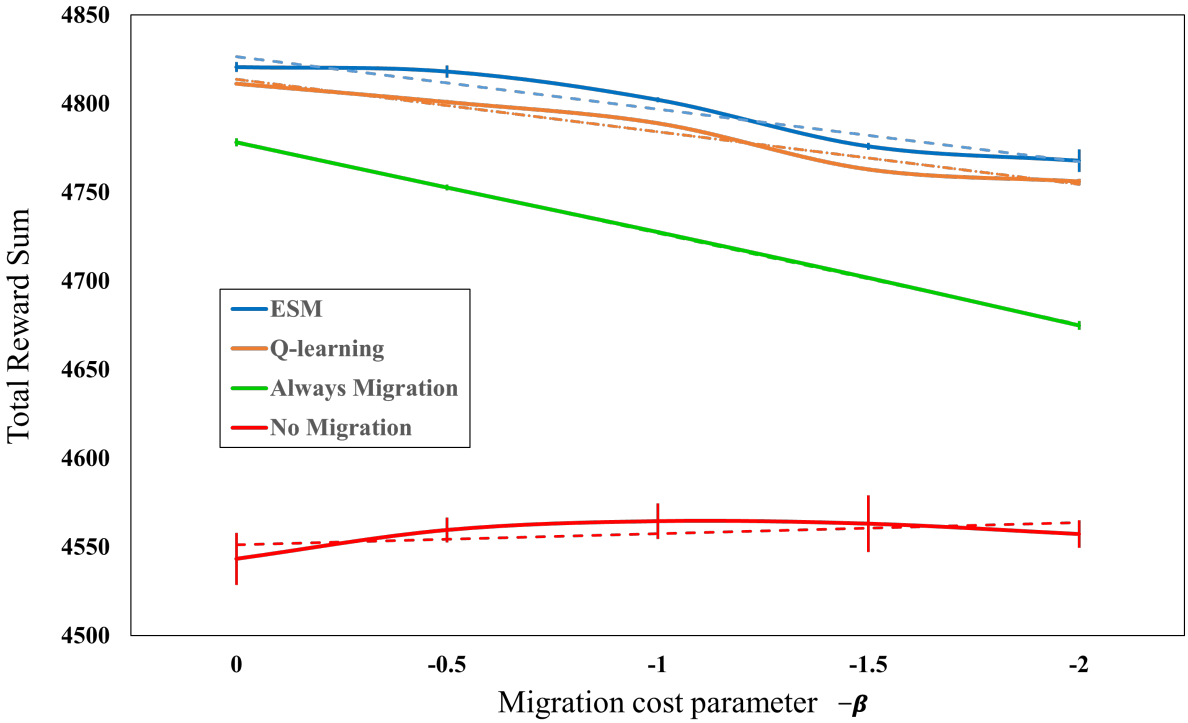


Figure 3.4: Total reward sum regarding the migration cost parameter $-\beta_l$ when MEC No. $= 30$, and $C = 20$.

parameter. Compared to the second experiment result, the algorithm has improved overall outcomes. Apart from that, ESM can achieve the best reward sum regardless of the content size. It indicates that the model is properly trained to obtain the optimal policy under the variations of the environmental factors.

3.3.4 Summary

The numerical simulations demonstrate that the proposed extensive service migration model exceeds existing algorithms in most cases. The suggested model outranks the conventional approaches, No-Migration and Always-Migration. Our proposal also improves the

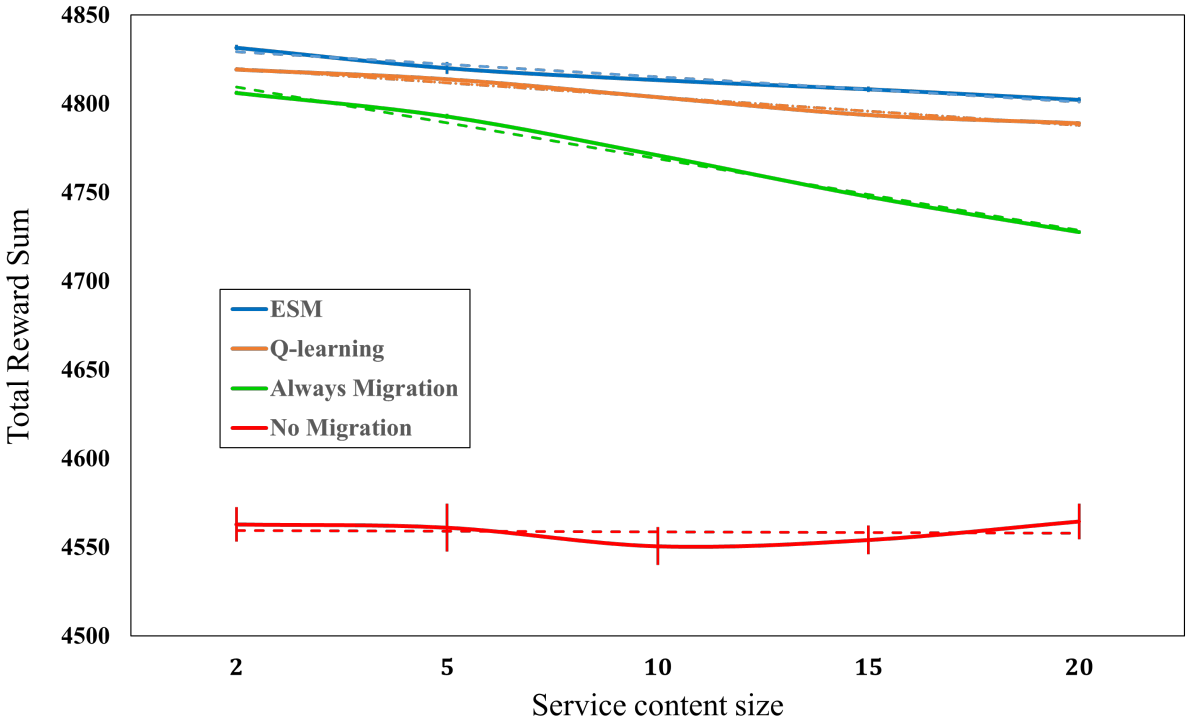


Figure 3.5: Total reward sum regarding the service content size C when MEC No. $= 30$, and $-\beta_l = 1$.

system's performance compared to the Q-learning method, which results in fairly improved results compared to the straightforward algorithms (No-Migration and Always-Migration) in the single-user case. However, due to the limitation of the q-table index, q-learning is limitedly applied to the actual service migration environment, which has oversized input parameters.

The application of the DRL-based extensible service migration model to the single-user mobility environment can be a bridgehead to propose a more improved service migration concept by using the DRL technology. Based on this, it is required to go one step further to develop a responsive migration algorithm that can substantially respond to the service

migration environment in real life.

Chapter 4

An innovative multi-user service migration model

The migration models' primary purpose is to maintain reliable QoS with low latency and reasonable cost. However, it is challenging to design a feasible service migration model in a situation where the number of mobile users increases exponentially through recent advancements in computing hardware and wireless communication technologies. Moreover, the complexity of user mobility patterns that are increasing day by day doubles the difficulty. In this chapter, we presented a novel multi-user service migration prototype based on DRL that deals with the complicated real-world service migration scenarios in MEC.

4.1 Service Migration Model

As illustrated in Figure 4.1, we consider the problem of service migration in a dynamic environment with multiple mobile users. We focus on establishing standards for making appropriate action judgments in response to surrounding situations. As a result, we have built a flexible service migration model that can take into account a wide range of dynamic environmental parameters.

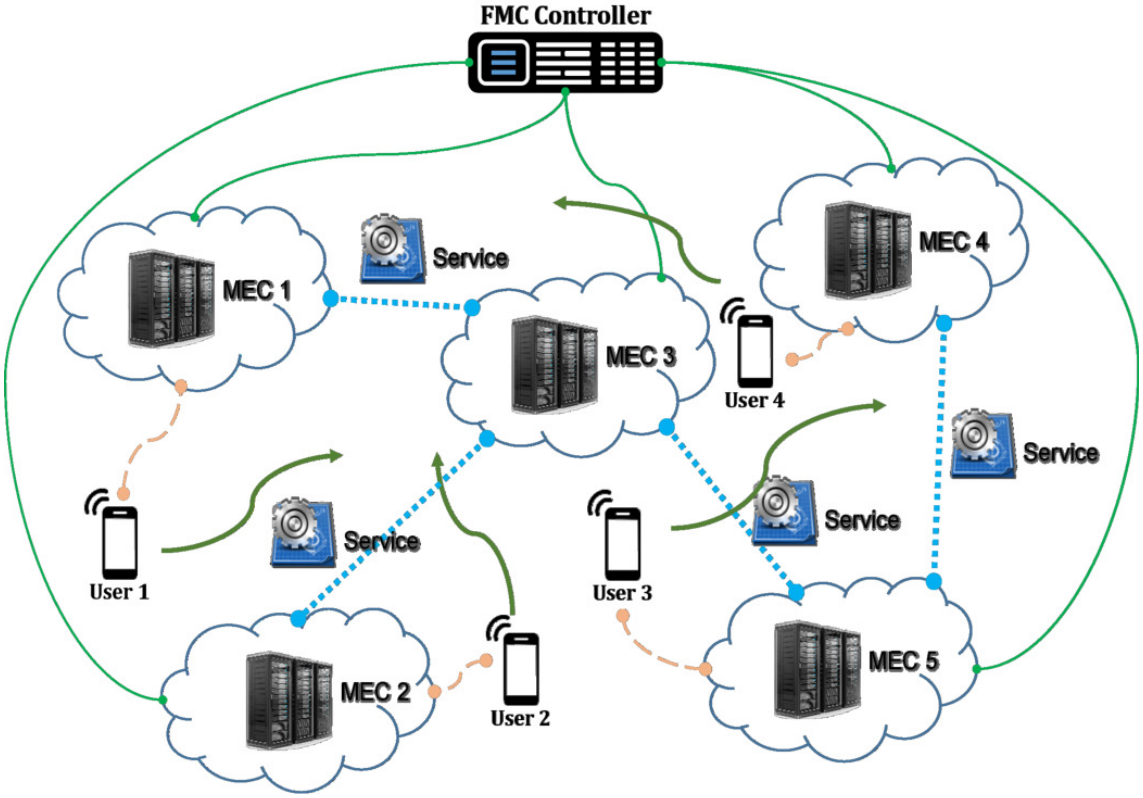


Figure 4.1: Interworked distributed cloud/mobile networks architecture.

We assume that the decisions to whether, when, and where to migrate the services to satisfy users' QoS, are made in a time-slotted fashion, $t \in \mathcal{T} = \{1, \ldots, T\}$, based on the the concept of FMC. We consider L different MEC servers in the system. The locations of servers, denoted as $l_i \in \mathcal{L}$ where $i \in \{1, \ldots, L\}$, are 2-dimension vectors and the distance between servers m and n, where $n, m \in \{1, 2, , \cdots L\}$, can be calculated as the norm value of them (i.e., $\|l_n - l_m\|$). Also, We consider N multiple users and illustrate the location of each user and its corresponding service location at timeslot t by $u_n(t)$ and $h_n^s(t)$, respectively, where, $n \in \mathcal{N} = \{1, \ldots, N\}$.

Each timeslot consists of three steps: Move, Collect, and Action (as shown in Figure 4.2). At the Move step, each user u_n moves to the new location randomly, and then the

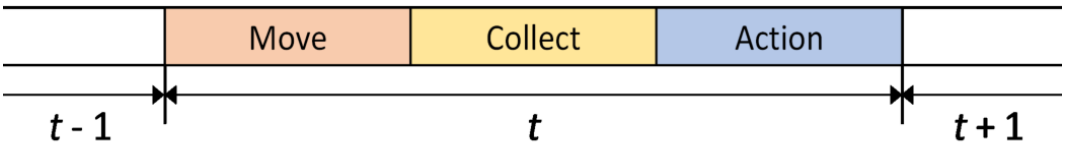


Figure 4.2: The timeslot structure of the proposed model.

agent collects all information of MECs and users at the Collect step. Lastly, each user executes the selected action with the current state based on the policy and obtains the reward value.

4.1.1 Migration / Transaction Cost

We assume that the whole service is migrated when the action occurs at each timeslot, and we utilize constant-plus-exponential cost function from [197], and [196] to model both migration and transition costs. The usefulness of the approximator was experimentally demonstrated in the aforementioned references. Note that each state is expressed as $s_n(t) = \{u_n(t), h_n^s(t), p_n(t)\}$. The parameter, $p_n(t)$, denotes the number of users connected to the same MEC server which is providing service to user n. The state after the action $a_n(s_n(t))$ is denoted as $s'_n(t)$. Then, $h'^s_n(t)$ is the new service location and the connected MEC location is determined and denoted as $h'^c_n(t)$ based on $u'_n(t)$. Furthermore, $p'_n(t)$ is the number of connected users to the new service location. The total costs consist of the migration cost, the transition cost both between the connected MEC and the service located MEC and between the user and the connected MEC, which are denoted as $c_n^m(x)$, $c_n^{d_1}(x)$ and $c_n^{d_2}(x)$ respectively; the expressions are as follows:

$$c_n^m(x) = \begin{cases} 0, & x = 0 \\ \beta_c + \beta_l \mu^x, & x > 0 \end{cases} \tag{4.1}$$

$$c_n^{d_1}(x), c_n^{d_2}(x) = \begin{cases} 0, & x = 0 \\ \delta_c + \delta_l \theta^x, & x > 0 \end{cases} \tag{4.2}$$

where β_c, β_l, δ_c, δ_l, μ, and θ are real-value parameters. The parameters μ and θ adjust the proportionality of distance to cost, and their values can be influenced by the network topology and routing mechanism of the network. Besides, the parameters β_l and δ_l are used to add extra control over tuning the relation between distance and cost. Inputs of $c_n^m(x)$, $c_n^{d_1}(x)$, and $c_n^{d_2}(x)$ are distances, $d_n^s(t)$, $d_n^h(t)$, and $d_n^c(t)$, where $d_n^s(t) = \|h_n^s(t) - h'^s_n(t)\|$, $d_n^h(t) = \|h'^s_n(t) - h'^c_n(t)\|$ and $d_n^c(t) = \|u'_n(t) - h'^c_n(t)\|$, respectively. The expression of the one-timeslot cost for a user is given by (4.3).

$$c_n^a(s_n(t)) = c_n^m(d_n^s(t)) + c_n^{d_1}(d_n^h(t)) + c_n^{d_2}(d_n^c(t)) \tag{4.3}$$

4.1.2 Energy Cost

In this part, we define the energy consumption of servers caused by the migration between MEC servers and the transmission between the user and the connected server. Although this is a minor problem since the server is constantly plugging, we can use it as a control criterion to cope with the frequent migrations. The researchers in [111] developed the dynamic migration energy consumption and demonstrated its performance. The method measures the energy consumption during the service migration, considering the content's size, noise, and the capacity of the transmission channels. Therefore, we adopt their proposed energy consumption formula for the service migration processes, which is expressed by (4.4).

$$E_n^a(S_n(t)) = \frac{\tau C \cdot \sigma(2^{R/W} - 1) \cdot (d_n^s(t)^2 + d_n^c(t)^2)}{R \cdot \lambda^2} \tag{4.4}$$

where C, σ, R, W, and λ denote the service content size, the noise power, the transmission rate, the bandwidth, and the channel attenuation coefficient, respectively. We further

employ the degree of service unexecuted service $\tau \in [0\%, 100\%]$, which is presented in the article [127], to approximate the running state of the service to be migrated.

4.1.3 Latency Cost

We utilize the constant delay Equation (4.5) to reflect the impact of the exceeded throughput in the MEC server.

$$L_n^a(s_n(t)) = \begin{cases} 0, & C_{all} \leq C_{limit} \\ \kappa, & C_{all} > C_{limit} \end{cases} \tag{4.5}$$

where κ means the delay penalty, C_{limit} indicates the total capacity of the MEC server, and C_{all} is the unexecuted service size of the users involved in the MEC server. Note that the entire server's demanded throughput is considered to calculate the reward value even though we add the number of users connected to the MEC server to the input state for simplicity. We assume that the number of connected users is sufficient as an indicator to represent the density of the server for training the neural network since the change of it probabilistically reveals the load on the server.

4.2 Problem Solution

In this section, we formulate an arithmetical method based on the DQN algorithm to implement the service mentioned above migration model. We define the Q-learning function as well as DQN-based algorithms, which employ the efficient DQN methodologies, such as the experience replay and Dueling-DQN (DDQN) [173]. Besides, we adopt two different DNN as an approximator: normal fully connected layers (FCL) and LSTM [110] with FCL. In the RL algorithm, the agent can respond to the ever-changing migration environment in a proactive manner. In other words, the agent can make a particular action when it encounters changes in the environment. As shown in Figure 4.3, a particular action results

in a reward and a new state. The agent then updates its policy to perform another action based on the received reward and the new state. With the recursive process, the agent obtains the optimal policy as determining the action, which maximizes the cumulative rewards [181].

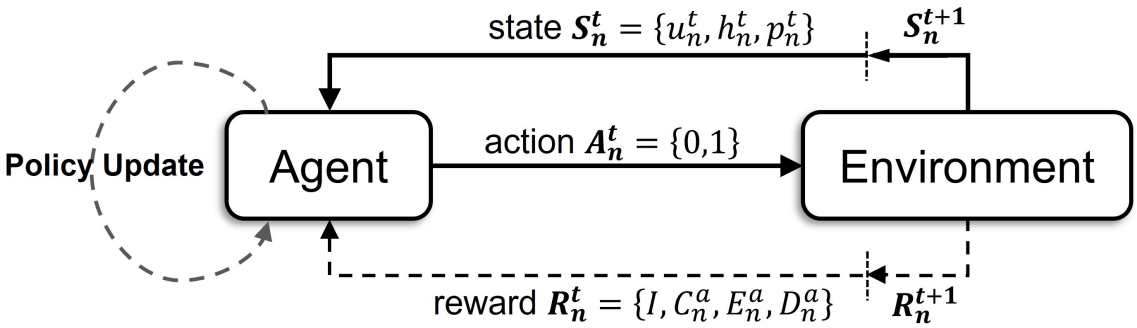


Figure 4.3: The agent–environment interaction with multi user.

We define the fundamental factors (state, action, and reward) as follows:

- **State:** The state is composed of two parts: the locations and the MEC server's capacity. We first define the state as the locations of both the UE and the MEC server in which the service is located (denoted as $u_n(t)$ and $h_n^s(t)$ at timeslot t). Furthermore, we add the number of users connected to the MEC server, $p_n(t)$, to the state to recognize the load associated with multi-user connections.

- **Action:** The agent can move from state $s_n(t)$ to the possible state $s'_n(t)$ by taking the action $a_n(s_n(t))$. The action belongs to the action set $\mathcal{A} = \{0, 1\}$, which means whether to migrate or not.

- **Reward:** The agent obtains the reward according to the action and the state. In this paper, we organize the objective function as the sum of the migration/transaction cost, the energy consumption, and the latency, which were discussed in section 4.1. Furthermore, we define the reward function as the difference between an adequate

large QoS value denoted as I and the object function. Therefore, from the equations (4.3), (4.4), and (4.5), the reward function can be represented by:

$$R_a(s(t)) = \sum_{i=1}^{n} \{I - [C_n^a(s_n(t)) + \rho \cdot E_n^a(s_n(t)) + L_n^a(s_n(t))]\} \tag{4.6}$$

where, ρ is a constant coefficient to adjust the scale difference between the cost and the energy consumption values.

4.2.1 Q-value function

Apart from the conventional Q-learning algorithms, which may suffer from a dimensional disaster in a real-world scenario, improved DQN algorithms have been introduced to reduce the computational complexity. We further adopt one of them, referred to as DDQN. In DDQN, the last layer is split into two parts, one of them to estimate the state-value function ($V(s)$) and the other one to estimate the advantage function ($A(s, a)$) for each action a. The last layer combines the outputs of both networks into a single output, which is an estimation of the Q-values. Therefore, DDQN can recognize the state value, excluding the effect of each action on the state, which allows the agent to promptly define an appropriate action in a situation where the action is not affected or similar actions are duplicated. This change is helpful because, in some cases, it is unnecessary to know the exact value of each action, so just learning the state-value function can be enough. The Q-function can be represented as below:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left\{ A(s, a; \theta, \alpha) - \frac{1}{|A|} \sum_{a} A(s, a; \theta, \alpha) \right\} \tag{4.7}$$

where $V(s)$ denotes the value of being at the state, and $A(s, a)$ represents the advantage value of taking action a while being in state s. θ denotes the weight vector of the neural network layers. α and β represent the parameter vectors of the advantage stream and the state value stream, respectively. In the last aggregation layer, simply summing the value

network estimation and the advantage network value causes a lack of identifiability. In other words, the backpropagation step may experience failure since $V(s)$ and $A(s,a)$ can not be obtained from a given $Q(s,a)$. Thus, the stability of optimization can be improved by subtracting the average advantage of all possible actions.

4.2.2 Deep Q-network

Since it is inevitable to define more detailed and diversified states which accurately capture all environmental parameters of a real-world service migration problem, it is unrealistic to directly apply the Q-learning algorithm due to the increased complexity. In a multi-user migration model, the complexity increases exponentially with the number of users. In other words, expanding the targeted states demands more computational effort and power to attain the optimal output by tracing all the circumstances induced by possible actions in a particular state. Consequently, it is helpful to devise an approximation approach to estimate the Q-value [135]. In this regard, DQN employs DNN as a function approximator to reduce the time and space complexity. In this paper, we apply basic FCL and LSTM with FCL for the DNN layers to implement the proposed model. We take the states as input parameters and conduct the feed-forward and the back-propagation processes to obtain corresponding output Q-values for each possible action.

LSTM, a kind of Recurrent Neural Network (RNN) algorithm with a recurrent neural network structure, can efficiently estimate continuous data since the current node contains the information for not only the last node but also all previous nodes. Besides, to solve the Long-Term-Dependency issue of RNN, LSTM introduces distinct concepts such as Memory Cells and three gates (i.e., forget, input, and output). Memory cells play a crucial role in the LSTM algorithm to store hidden weight states to share the memory contents across all network nodes. Moreover, memory cells manage the state memory in conjunction with each gate and transfer it to the following node to mitigate the gradient descent problem [100,110].

4.2.3 Service migration algorithm

Even though we can apply DNN to resolve the issue of complexity caused by excessive states and actions, DQN training may fall into local optimization or unexpectedly diverge due to the correlation of sequential states. To handle this problem, we apply experience replay, which stores and utilizes the agent's experience data in memory. We also employ the off-policy control to eliminate the tendency of Q-learning to overestimate the Q-values. Thus, we divide the policy update into the behavior policy update and the target policy update. This approach facilitates the model to learn the optimal policy while following exploratory policy. As shown in Algorithm 4.1, the DQN process first initializes the replay memory, and both the action Q-Network and the target Q-Network with random weight θ. At the Move step of each timeslot, all users change their locations based on the mobility pattern. Next, all users' states, the distance between the user n and the MEC server, and the number of users near the MEC server are collected to comprehensively consider the timeslot situations. Following that, each user selects and executes the action according to an ε-greedy policy during the action step, resulting in the reward and the subsequent state. At each timeslot, every user's experience $e_t = (s_t, a_t, r_t, s_{t+1})$ is stored in the dataset $\mathcal{D}_t = \{e_{t-D+1}, \ldots, e_t\}$, which will be used for training purposes. During the update process, samples of experience are picked uniformly at random from the batch $\mathcal{D}_t$ of the latest D experiences. We perform the update rule by minimizing the loss function to derive the network's weight at each training episode. The update process will continue until the algorithm converges and the optimal policy for the service migration is achieved [136].

Every C steps the weights of the action Q-Network are copied into the target Q-network. The target Q-Network enables the algorithm to restrain the policy from faltering and divergence. In the case of the DDQN, the algorithm can also use the same update process as in DQN algorithm to obtain the optimal policy since the only difference with the basic DQN is the additional layers after the approximation layers.

Algorithm 4.1 Deep Q-learning with experience replay

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

for *episode* $= 1, M$ **do**

  Initialize sequence $s_n^1 = \{u_n^1, h_n^1, p_n^1\}$

  preprocess sequence $\emptyset_1 = \emptyset(s_n^1)$

  for $t = 1, T$ **do**

    With probability ε select a random action a_n^t

    otherwise select $a_n^t = \arg\max_a Q(\emptyset(s_n^t), a; \theta)$

    Collect $s_n^{t+1} = \{u_n^{t+1}, h_n^{t+1}, p_n^{t+1}\}$

    Execute action a_n^t in simulator

    Observe reward r_n^t and state $\{u_n^{t+1}, h_n^{t+1}, p_n^{t+1}\}$

    Set $s_n^{t+1} = s_n^t, a_n^t, \{u_n^{t+1}, h_n^{t+1}, p_n^{t+1}\}$

    Preprocess $\emptyset_{t+1} = \emptyset(s_n^{t+1})$

    Store transition $(\emptyset_t, a_n^t, r_n^t, \emptyset_{t+1})$ in D

    Sample random mini-batch of transitions $(\emptyset_j, a_j, r_j, \emptyset_{j+1})$ from D

    Set $y_i = \begin{cases} r_j, \text{ if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\emptyset_{j+1}, a'; \theta^{-}), \text{ otherwise} \end{cases}$

    Perform a gradient descent step on $(y_j - Q(\emptyset_j, a_j; \theta))^2$ with respect to the network parameters θ

    Every C step reset $\hat{Q} = Q$

  end for

end for

4.3 Performance Evaluation

In this section, we use numerical simulations to study the performance of our proposed service migration algorithm. We assume that the user follows the random walk mobility model within the coverage area of multiple MEC Servers, which are randomly deployed in a 100×100 grid area, to achieve a realistic simulation environment. To validate the effectiveness of the proposal, we compare the results obtained from applying different algorithms: DQN, Dueling DQN (DDQN), DQN with LSTM (DLSTM), Q-learning, No-Migration, and Always-Migration. We apply the same parameter settings (3.3.1) that were used for the simulations of the single-user service migration model.

We train the proposed DQN models to reduce the total gain of the cost and the energy consumption during the migration process under random user mobility patterns. Given the current state of the environment, the trained model can take appropriate action based on the optimal policy. In terms of No-Migration algorithm, the agent lets the service remain at its original service located MEC, and the user should communicate to that server through the connected MEC. Therefore, the reward value is only affected by the transaction costs between the user and the connected MEC and between the service located MEC and the connected MEC. On the other hand, in Always-Migration algorithm, the model continuously allows the service to move to the closest MEC server as the user moves around the area. This mechanism will result in a reduction in the transaction cost in exchange for having higher migration costs and energy consumption.

Regarding the Q-learning algorithm, the agent complies with the policy based on Q-function and Q-table. It repeatedly estimates Q-value and updates the Q-table to achieve the optimal policy. It has a limitation on the number of states since it uses the Q-table with restricted index values. Particularly, the application of the conventional Q-learning in a real-world service migration scenario with too many state combinations is not adequate due to the large size of the Q-table. Additionally, we simulate the single-user service migration scenario from our previous work [147] with a modified energy cost function to make a more accurate comparison with the proposed multi-user migration model. We investigate the

variation of the total reward sum when the migration cost parameter $-\beta_l$ changes from 0.0 to 2.0 to examine the impact of the variable. The migration trend is determined according to the cost parameter $-\beta_l$. In other words, it is advantageous not to migrate under the larger $-\beta_l$ because it makes the migration cost high; otherwise, it is better to migrate [196]. For the energy consumption function, we also examine the effect of the service content size using different values ranging from 2(KB) to 20(KB).

4.3.1 Performance of Model

We execute two sets of simulations to evaluate the impact of both the number of users and MEC servers on the performance of the proposed algorithms. Table 4.1 compares the total reward sum of the object group of different algorithms when the number of users varies from 5 to 50. The DQN-based migration models always show higher reward sum values with each number of users than conventional methodologies. While No-migration algorithm shows a constant trend of reward sum despite changes in the number of users, the other experimental groups result in a slight decrease in the reward sum as the number of users increases. The reason is that, as the number of users increases, the probability of being affected by the server capacity limit raises, while No-Migration has a constant server capacity limit impact.

Furthermore, the complexity of DQN and Q-learning algorithms increases dramatically with the size of the state space. The state-space includes all possible combinations of users' locations and their corresponding service-located servers, which obviously grows exponentially with the number of users. The other reason for having a high complexity is that the agent has to estimate the reward value and determine the optimal action for each user, increasing the processing time proportional to the number of users.

Figure 4.4 provides a comparison for the total reward of the mentioned algorithms as the number of MEC increases for both multi-user and single-user scenarios. As seen in this figure, the proposed algorithms, DQN, DDQN, and DLSTM, outperform the conventional Q-learning, No-Migration, and Always-Migration considerably. Similar to the single-user

case, which presented an increasing tendency, the total reward sum also tends to rise in multi-user cases as the number of servers increases. The more servers are deployed, the agent has more options to distribute the required computation capacity for service processing, thereby facilitating a fine-grained migration policy that can minimize the delay penalty. It is also supported by the fact that No-Migration, which is not affected by the delay penalty, exhibits a steady pattern regardless of alterations in the number of servers. Always-Migration is uneven but presents a climbing slope.

4.3.2 Impact of variables

The third set of simulations evaluates the impact of the migration cost parameter $-\beta_l$ on the total reward sum results. In this part, we investigate the variation of the outcomes as $-\beta_l$ changes from 0.0 to 2.0 while the number of users, MEC servers, and the content size are set to 20, 30, and 20(KB), respectively. As depicted in Figure 4.5. a, the performance of the DQN models, Q-learning, and Always-Migration manifests a reduction when $-\beta_l$ value is raised. Therefore, it is advantageous not to migrate under the larger $-\beta_l$ because it makes the migration cost high. A more significant drop is shown in Always-Migration algorithm, which is highly affected by the migration cost in all actions as opposed to No-Migration, which represents no vacillation since it is not dependent on the migration cost. Although the DQN-related models represent a slight drop, they display better performance than Always-Migration and Q-learning algorithms as the optimal policy minimizes the impact of the migration cost parameter. This trend is more noticeable in the multi-user situation in comparison with the single-user case (shown in Figure 4.5. b).

In the last set of simulations, we examine the impact of the content size, which is a significant factor in estimating the energy consumption and the latency components of the reward values. Figure 4.6. a shows the total reward sum for different values of service content size ranging from 2(KB) to 20(KB). As shown in this figure, lower values for content size lead to a higher reward value for the DQN, Q-learning, and Always-Migration algorithms, which is consistent with intuition. In the case of Always-Migration

and Q-learning algorithm, this component seems to be more affected by the content size. Despite the reward diminution due to the increment in content size, the DQN-based models outperform the other algorithms to be compared, which indicates that the proposed DQN algorithms can manage to find the optimal action, regardless of the additional constraints that are introduced to the system model. The total reward sum versus content size for the case of single-user is illustrated in Figure 4.6. b. All migration-related algorithms show slightly declined reward sums with increasing the content size. However, in the No-Migration model, the reward for single-user is not affected by the migration, while the result in the multi-user circumstance varies according to the change in content size due to the factor related to the latency.

4.3.3 Summary

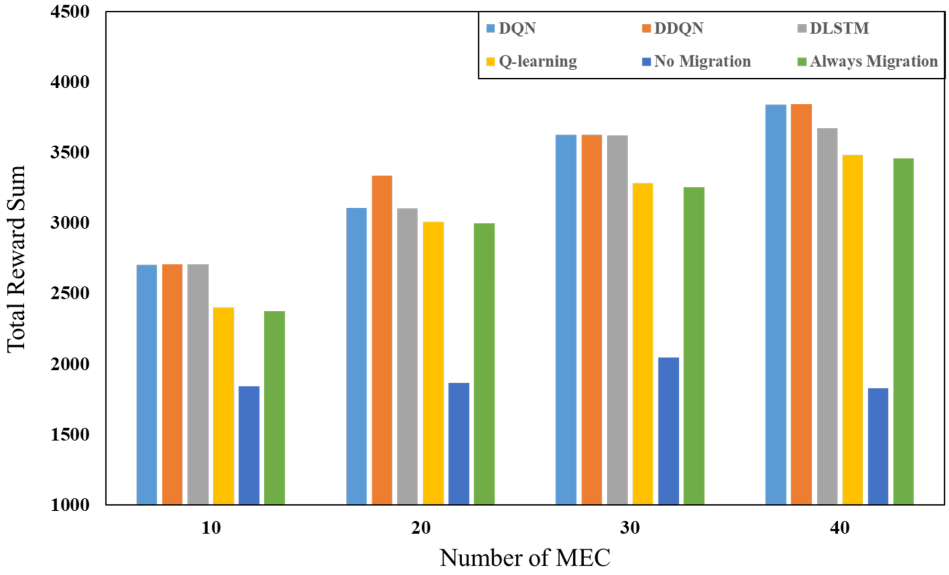
Based on the numerical results derived from the analytical simulations, we can observe that the proposed novel service migration models outperform the conventional algorithms in most circumstances. The proposals outweigh the straightforward methods, No-Migration and Always-Migration. The DQN-related algorithms also improve the performance of the system compared to the conventional Q-learning method. With the Q-learning method, we can get relatively more valuable results compared to the intuitive algorithm (Always-Migration) in the single-user case. However, in the case of the multi-user scenario, the total rewards are not remarkable because the server capacity factor is disregarded due to the state limitation of Q-learning.

Advanced DQN models such as DDQN and DLSTM generally show better performance results than the typical DQN algorithm in environments that handle complex action states such as game simulations [109]. However, in our simulation cases where the set of action-state pairs is relatively small, the two improved algorithms rather hinder the learning process, resulting in similar results to the standard DQN. To be specific, the DDQN algorithm can measure the value of a state without the action impact by calculating an estimate for value and action. It improves the algorithm efficiency by excluding the estimation process

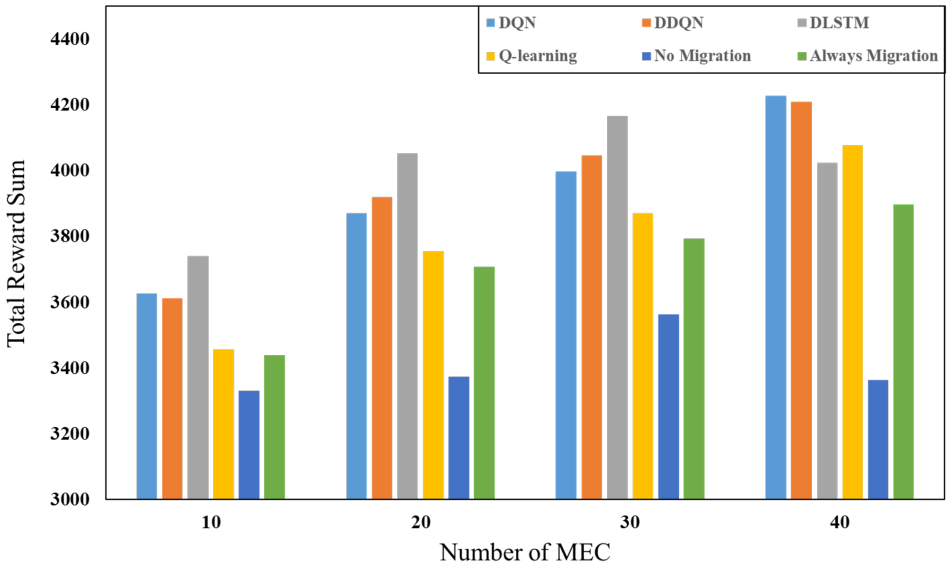
for the actions that do not affect the situation or are similarly reproduced. However, the separated layers cause inefficiency in the service migration environment since all states are affected by actions. LSTM-DQN also has little impact on the service migration model, which has less continuity between each value of the state set.

Table 4.1: Total reward sum for multi-user regarding the number of Users when MEC No.=30, $-\beta_l = 1$, and $C = 20$.

Algorithm	**Number of User**									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
No-Migration	2048.6	2032.7	2035.8	2045.9	2044.1	2045.7	2042.9	2045.1	2043.1	2041.2
Always-Migration	3643.8	3490.5	3362.6	3253.1	3158.6	3072.0	2994.0	2925.3	2863.8	2806.2
Q-learning	3670.1	3515.5	3391.5	3280.6	3180.4	3095.2	3016.9	2948.2	2886.7	2832.0
DQN	**3942.5**	3710.1	3700.1	**3624.6**	3526.2	3189.2	3110.0	3293.2	2978.6	**3178.1**
DDQN	3874.6	3715.1	3697.0	3623.1	**3526.5**	**3440.5**	3363.3	**3295.6**	**3233.5**	3177.5
DLSTM	3758.0	**3856.0**	**3733.0**	3620.3	3526.2	3440.0	**3364.2**	3294.6	3233.2	2925.8

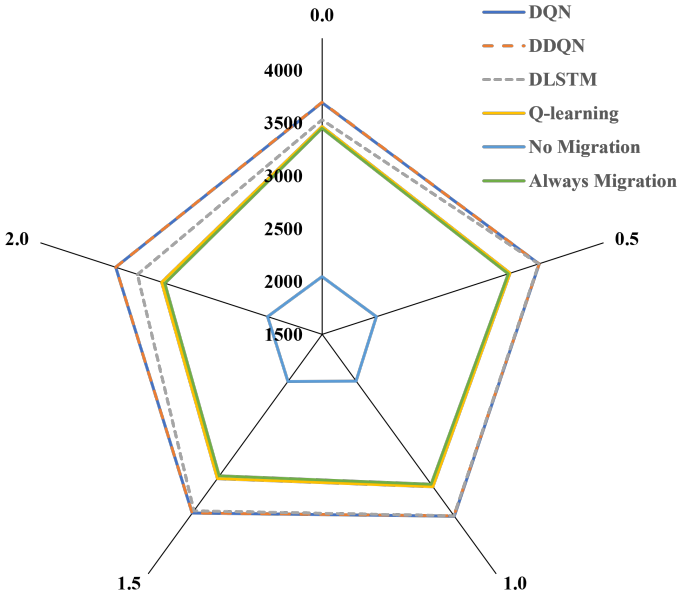


a) Multi-User

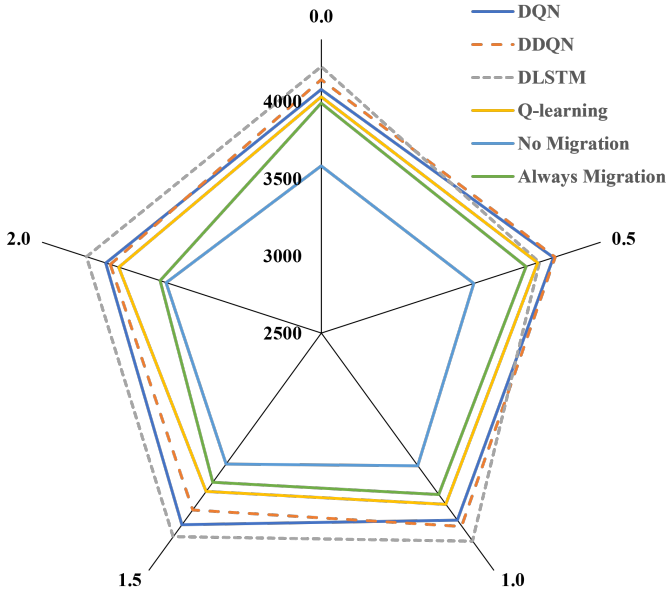


b) Single-User

Figure 4.4: Total reward sum regarding the number of MECs when User No. $= 20$, $-\beta_l = 1$, and $C = 20$.

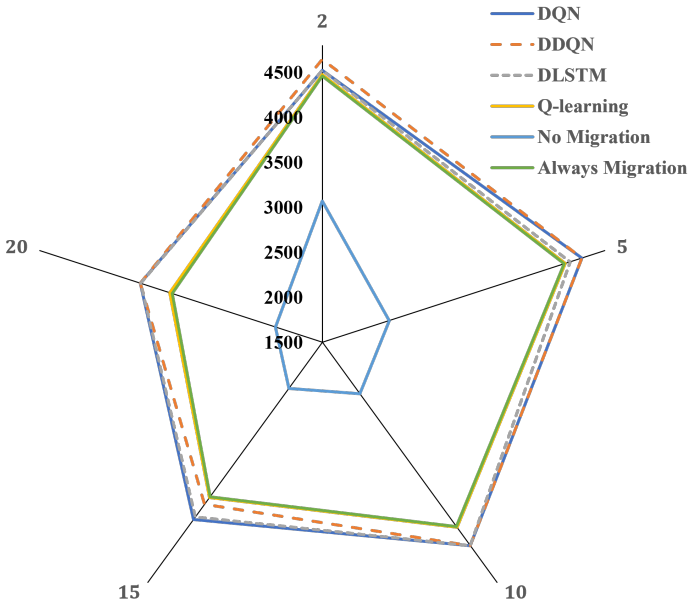


a) Multi-User

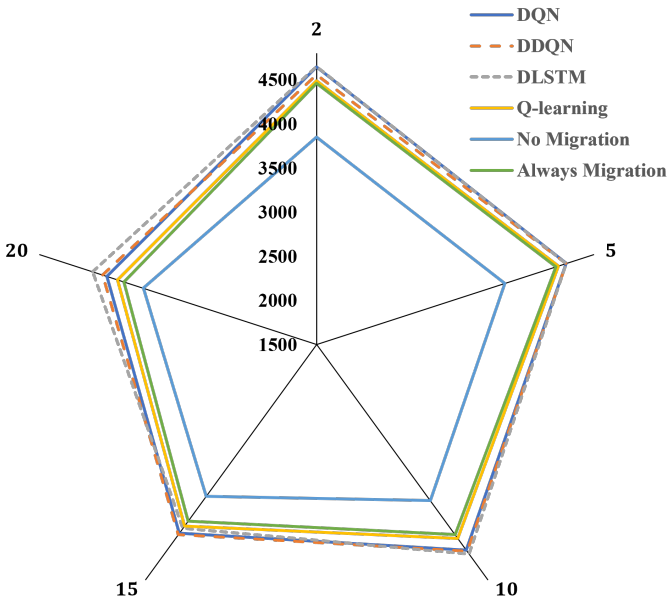


b) Single-User

Figure 4.5: Total reward sum for different migration cost parameter $-\beta_l$ when User No. $= 20$, MEC No. $= 30$, and $C = 20$.



a) Multi-User



b) Single-User

Figure 4.6: Total reward sum for different service content size C when User No. $= 20$, MEC No. $= 30$, and $-\beta_l = 1$.

Chapter 5

Conclusion and Future Work

This chapter summarizes our contributions and introduces open issues and future directions for obtaining improved service migration models.

5.1 Conclusion

Dynamic service migration is essential to sustaining QoS related to MEC in response to multiple user mobility patterns. Implementing a service migration model that can adapt unpredictable user mobility patterns with a static conventional optimization algorithm is unattainable. Fortunately, with the recent progress of DL technology, it has become achievable to develop a robust migration algorithm in MEC.

In this thesis, we first reviewed a comprehensive background of edge computing technology. Then, we investigated the achievements of related studies to understand the current status of service migration research. Moreover, we classified the types of machine learning and described the features and specific algorithms for each category. We also explained the details and structures of representative deep learning algorithms.

Based on the in-depth understanding of the MEC technology, we proposed an exten-

sive service migration model based on DRL to handle the service migration problem. In the proposed model, DQN is employed to solve the inevitable computation problem occurring from designing a service migration model that considers more aspects occurring in the practical environment. As demonstrating that the high-complexity computations can be handled using DQN as an approximator, we suggested the possibility of realizing the realistic service migration scenario based on the optimal policy under various determinants. In addition, we introduced a reward function based on numerous environmental variables and multi-user to support fine-grind decision-making. The enhanced DQN algorithms are applied to solve the computational complexity caused by considering multiple variables to model a real-world dynamic environment. Comparative simulations under several conditions were performed, and the numerical results were provided to illustrate that the proposed approaches surpassed the algorithms of the control group. In addition, the implications of the results were discussed through numerical analysis. Furthermore, we proved the potential of practicing realistic service migration scenarios in the real-world by demonstrating the effectiveness of the proposed DQN-based models. Finally, we presented open issues and challenges as well as future research directions related to the service migration in MEC.

5.2 Future Work

In line with the trend to engage DL technology in active research, several attempts to exploit DL in the service migration within MEC environments have proceeded. Although there have been many notable achievements, it is insufficient to reflect the real-world requirements related to task migration and user mobility in MEC. Some technical issues still impede the ongoing DL-based researches to control and manage the application executions at each MEC server. Therefore, we discuss the challenges and the direction for future work that can be further explored.

5.2.1 Deep Learning Model Design

As DL technology is widely used, the desire to improve DNN performance is also increasing. In the case of CNN, which is highly utilized among DNNs, improved neural networks such as AlexNet [117], VggNet [175], GoogleNet [182], and ResNeXts [205] are emerging. Furthermore, in the RL domain, novel DRL algorithms, such as Double-DQN, Dual-DQN, and Rainbow-DQN, have developed and shown enhanced performance as overcoming existing methods has proven in [109, 173]. Therefore, performance re-verification for the existing DL-based system problems should be performed by applying the newly improved algorithm. Also, comparative simulations between those algorithms should be conducted to determine which algorithm is appropriate according to the situation since new algorithms do not always produce better outcomes. A new opportunity can be derived by the intensified algorithms, which enable the design of creative scenarios that were formerly unreachable.

Regarding DQN, defining the state, action, and reward function are more critical, even though the RL algorithm used to find the optimal results is vital for building frameworks. The reward value function should be designed depending on the underlying factors that influence the variation of policies. For modeling a DRL framework for service migration, the optimal policy from the reward function based on only migration and communication costs is distinct from the case of additionally accepting energy consumption into the reward. Considering more determinants for identifying states, tasks, and reward factors leads to more sophisticated optimization models, which inevitably increases computational complexity. Consequently, we need to enhance DNN, which is utilized as an approximator to simplify complex states.

5.2.2 Real-world Scenario

As DRL technology develops, generating a responsive service model required in the MEC network environment became possible. Regardless, it is not sufficient to satisfy the re-

quirements in real life since studies related to the DRL-based service migration model have conditions and assumptions to simplify the processing.

Architectures, which can afford to interact with multiple users, are required to build a realistic service migration model. In other words, each user must simultaneously determine the optimal migration action based on what has occurred and collectively feedback the results to all users within the MEC network. To this end, it is fundamental to define state, action, and reward functions that can reflect the situation of multiple users and establish a sophisticated algorithm that can accumulate and process the entire user's information at each timeslot. Moreover, a structure of DNNs that can mitigate the complexity originated from the increased state size, and the computational amount should be investigated in parallel. In the paper [147], a model representing multiple environmental factors has been proposed to identify the migration situations more accurately. However, it needs to be developed to address multiple users.

Some papers [123,168,211] have proposed a multi-agent-based DRL model that handles network resources as another approach to support multiple users' circumstances. However, it is challenging to understand the state of the environment at the same time due to the inherent non-stationarity issues of multi-agent designs [108]. In addition, the complexity mentioned above is still a mission to reflect the scenario for the actual network physical layer. Therefore, it is inevitable to take robust measures to unravel these problems. Developing an improved DNN algorithm that simplifies many input variables and speeds up processing must be a useful sample for the measures.

5.2.3 Security Dilemma

The network communications underlying the migration frameworks are continuously exposed to security and privacy threats. There is a possibility for misusing or manipulating transmitted context or data within the network model by unauthorized users. Although the need for secure network communication stimulates the development of countermeasures such as data packet encryption technology, vulnerabilities still exist due to the slow-

down caused by the encryption process or the techniques to disable encryption by augmented computing capacity. In particular, there are many security-threatening spots that can be struck by modern security attack techniques such as Advanced Persistent Threat (APT) [73], malware attacks [155], and Distributed Denial of Service (DDoS) attacks [176].

During the service migration process, the state and context of each UE and cloud server are necessary to share for the agent to be aware of the situation of the MEC environment. Hence, it is required to apply a method to exchange data securely and quickly. A more sophisticated optimization model can be achieved with more contextual information exchanged. Thus, the agent needs to collect more information from UEs and MECs. However, the security and privacy concerns make individual users hesitant to share user information. Accordingly, an incentive strategy is mandated to encourage users to share private information.

References

[1] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[2] Kaouther Abrougui, Azzedine Boukerche, and Richard Werner Nelem Pazzi. Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):717–735, 2011.

[3] Osama Abumansoor and Azzedine Boukerche. A secure cooperative approach for nonline-of-sight location verification in vanet. *IEEE Transactions on Vehicular Technology*, 61(1):275–285, 2011.

[4] Ejaz Ahmed and Mubashir Husain Rehmani. Mobile edge computing: Opportunities, solutions, and challenges. *Future Generation Computer Systems*, 70:59–63, 2017.

[5] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[6] Moayad Aloqaily, Ouns Bouachir, Azzedine Boukerche, and Ismaeel Al Ridhawi. Design guidelines for blockchain-assisted 5g-uav networks. *IEEE network*, 35(1):64–71, 2021.

[7] Shogo Ando and Akihiro Nakao. In-network cache simulations based on a youtube traffic analysis at the edge network. In *Proceedings of The Ninth International Conference on Future Internet Technologies*, pages 1–6, 2014.

[8] Athanasios Bamis, Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks. *Computer Networks*, 52(1):130–154, 2008.

[9] Rodolfo Bezerra Batista, Azzedine Boukerche, and Alba Cristina Magalhaes Alves de Melo. A parallel strategy for biological sequence alignment in restricted memory space. *Journal of Parallel and Distributed Computing*, 68(4):548–561, 2008.

[10] Zdenek Becvar, Matej Rohlik, Pavel Mach, Michal Vondra, Tomas Vanek, Miguel A Puente, and Felicia Lobillo. Distributed architecture of 5g mobile networks for efficient computation management in mobile edge computing. In *5G Radio Access Networks: Centralized RAN, Cloud-RAN, and Virtualization of Small Cells*, pages 29–50. CRC Press, 2017.

[11] Graeme Best, Wolfram Martens, and Robert Fitch. Path planning with spatiotemporal optimal stopping for stochastic mission monitoring. *IEEE Transactions on Robotics*, 33(3):629–646, 2017.

[12] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[13] Ouns Bouachir, Moayad Aloqaily, Lewis Tseng, and Azzedine Boukerche. Blockchain and fog computing for cyberphysical systems: The case of smart industry. *Computer*, 53(9):36–45, 2020.

[14] A Boukerch, Li Xu, and Khalil El-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, 30(11-12):2413–2427, 2007.

[15] Azzedine Boukerche. Performance comparison and analysis of ad hoc routing algorithms. In *Conference Proceedings of the 2001 IEEE International Performance, Computing, and Communications Conference (Cat. No. 01CH37210)*, pages 171–178. IEEE, 2001.

[16] Azzedine Boukerche. A simulation based study of on-demand routing protocols for ad hoc wireless networks. In *Proceedings. 34th Annual Simulation Symposium*, pages 85–92. IEEE, 2001.

[17] Azzedine Boukerche. Performance evaluation of routing protocols for ad hoc wireless networks. *Mobile networks and applications*, 9:333–342, 2004.

[18] Azzedine Boukerche. *Handbook of algorithms for wireless networking and mobile computing*. CRC Press, 2005.

[19] Azzedine Boukerche. *Algorithms and protocols for wireless and mobile ad hoc networks*. John Wiley & Sons, 2008.

[20] Azzedine Boukerche and Luciano Bononi. Simulation and modeling of wireless, mobile, and ad hoc networks. *Mobile ad hoc networking*, pages 373–409, 2004.

[21] Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. *Computer communications*, 29(4):477–489, 2006.

[22] Azzedine Boukerche, Xiuzhen Cheng, and Joseph Linus. Energy-aware data-centric routing in microsensor networks. In *Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 42–49, 2003.

[23] Azzedine Boukerche, Xuzhen Cheng, and Joseph Linus. A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks. *Wireless Networks*, 11(5):619–635, 2005.

[24] Azzedine Boukerche and Amir Darehshoorzadeh. Opportunistic routing in wireless networks: Models, algorithms, and classifications. *ACM Computing Surveys (CSUR)*, 47(2):1–36, 2014.

[25] Azzedine Boukerche and Sajal K Das. Dynamic load balancing strategies for conservative parallel simulations. In *Proceedings of the eleventh workshop on Parallel and distributed simulation*, pages 20–28, 1997.

[26] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. Analysis of a randomized congestion control scheme with dsdv routing in ad hoc wireless networks. *Journal of Parallel and Distributed Computing*, 61(7):967–995, 2001.

[27] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. Swimnet: a scalable parallel simulation testbed for wireless and mobile networks. *Wireless Networks*, 7:467–486, 2001.

[28] Azzedine Boukerche, Sajal K Das, Alessandro Fabbri, and Oktay Yildiz. Exploiting model independence for parallel pcs network simulation. In *Proceedings Thirteenth Workshop on Parallel and Distributed Simulation. PADS 99.(Cat. No. PR00155)*, pages 166–173. IEEE, 1999.

[29] Azzedine Boukerche and Caron Dzermajko. Performance evaluation of data distribution management strategies. *Concurrency and Computation: Practice and Experience*, 16(15):1545–1573, 2004.

[30] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. A novel solution for achieving anonymity in wireless ad hoc networks. In *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 30–38, 2004.

[31] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. Sdar: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks.

In *29th Annual IEEE International Conference on Local Computer Networks*, pages 618–624. IEEE, 2004.

[32] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks. *computer communications*, 28(10):1193–1203, 2005.

[33] Azzedine Boukerche and Xin Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad hoc networks*, 5(8):1303–1316, 2007.

[34] Azzedine Boukerche and Xin Fei. A voronoi approach for coverage protocols in wireless sensor networks. In *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, pages 5190–5194. IEEE, 2007.

[35] Azzedine Boukerche, Xin Fei, and Regina B Araujo. An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange. *Computer Communications*, 30(14-15):2708–2720, 2007.

[36] Azzedine Boukerche, Shichao Guan, and Robson Eduardo De Grande. A task-centric mobile cloud-based system to enable energy-aware efficient offloading. *IEEE Transactions on Sustainable Computing*, 3(4):248–261, 2018.

[37] Azzedine Boukerche, Shichao Guan, and Robson E. De Grande. Sustainable offloading in mobile cloud computing: Algorithmic design and implementation. *ACM Comput. Surv.*, 52(1), feb 2019.

[38] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A distributed algorithm for dynamic channel allocation. *Mobile Networks and Applications*, 7:115–126, 2002.

[39] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *IEEE transactions on Parallel and Distributed Systems*, 13(9):911–923, 2002.

[40] Azzedine Boukerche, Kathia Regina Lemos Jucá, Joao Bosco Sobral, and Mirela Sechi Moretti Annoni Notare. An artificial immune based intrusion detection model for computer and telecommunication systems. *Parallel Computing*, 30(5-6):629–646, 2004.

[41] Azzedine Boukerche and Xu Li. An agent-based trust and reputation management scheme for wireless sensor networks. In *GLOBECOM'05. IEEE Global Telecommunications Conference, 2005.*, volume 3, pages 5–pp. IEEE, 2005.

[42] Azzedine Boukerche, Renato B Machado, Kathia RL Jucá, João Bosco M Sobral, and Mirela SMA Notare. An agent based and biological inspired real-time intrusion detection and security model for computer network operations. *Computer Communications*, 30(13):2649–2660, 2007.

[43] Azzedine Boukerche, Anahit Martirosyan, and Richard Pazzi. An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks. *Mobile Networks and Applications*, 13:614–626, 2008.

[44] Azzedine Boukerche and Sotiris Nikoletseas. Protocols for data propagation in wireless sensor networks. *Wireless communications systems and networks*, pages 23–51, 2004.

[45] Azzedine Boukerche and Mirela Sechi M Annoni Notare. Behavior-based intrusion detection in mobile phone systems. *Journal of Parallel and Distributed Computing*, 62(9):1476–1490, 2002.

[46] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. Localization systems for wireless sensor networks. *IEEE wireless Communications*, 14(6):6–12, 2007.

[47] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. Secure localization algorithms for wireless sensor networks. *IEEE Communications Magazine*, 46(4):96–101, 2008.

[48] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. Vehicular ad hoc networks: A new challenge for localization-based systems. *Computer communications*, 31(12):2838–2849, 2008.

[49] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo Freire Nakamura, and Antonio AF Loureiro. Dv-loc: a scalable localization protocol using voronoi diagrams for wireless sensor networks. *IEEE Wireless Communications*, 16(2):50–55, 2009.

[50] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina B Araujo. Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol. In *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l*, pages 560–567. IEEE, 2005.

[51] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 157–164, 2004.

[52] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments. *Journal of Parallel and Distributed Computing*, 66(4):586–599, 2006.

[53] Azzedine Boukerche and Yonglin Ren. A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In *Proceedings of the 5th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 88–95, 2008.

[54] Azzedine Boukerche and Yonglin Ren. A trust-based security system for ubiquitous and pervasive computing environments. *Computer communications*, 31(18):4343–4351, 2008.

[55] Azzedine Boukerche and Yonglin Ren. A secure mobile healthcare system using trust-based multicast scheme. *IEEE Journal on Selected Areas in Communications*, 27(4):387–399, 2009.

[56] Azzedine Boukerche, Cristiano Rezende, and Richard W Pazzi. Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages. In *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, pages 1–6. IEEE, 2009.

[57] Azzedine Boukerche and E Robson. Vehicular cloud computing: Architectures, applications, and mobility. *Computer networks*, 135:171–189, 2018.

[58] Azzedine Boukerche and Steve Rogers. Performance of gzrp ad hoc routing protocol. *Journal of Interconnection Networks*, 2(01):31–48, 2001.

[59] Azzedine Boukerche and Amber Roy. Dynamic grid-based approach to data distribution management. *Journal of Parallel and Distributed Computing*, 62(3):366–392, 2002.

[60] Azzedine Boukerche, Amber Roy, and Neville Thomas. Dynamic grid-based multicast group assignment in data distribution management. In *Proceedings Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT 2000)*, pages 47–54. IEEE, 2000.

[61] Azzedine Boukerche and Samer Samarah. A novel algorithm for mining association rules in wireless ad hoc sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):865–877, 2008.

[62] Azzedine Boukerche, Yanjie Tao, and Peng Sun. Artificial intelligence-based vehicular traffic flow prediction methods for supporting intelligent transportation systems. *Computer networks*, 182:107484, 2020.

[63] Azzedine Boukerche and Carl Tropper. A static partitioning and mapping algorithm for conservative parallel simulations. In *Proceedings of the eighth workshop on Parallel and distributed simulation*, pages 164–172, 1994.

[64] Azzedine Boukerche and Carl Tropper. A distributed graph algorithm for the detection of local cycles and knots. *IEEE Transactions on Parallel and Distributed Systems*, 9(8):748–757, 1998.

[65] Azzedine Boukerche, Begumhan Turgut, Nevin Aydin, Mohammad Z Ahmad, Ladislau Bölöni, and Damla Turgut. Routing protocols in ad hoc networks: A survey. *Computer networks*, 55(13):3032–3080, 2011.

[66] Azzedine Boukerche and Damla Turgut. Secure time synchronization protocols for wireless sensor networks. *IEEE Wireless Communications*, 14(5):64–69, 2007.

[67] Azzedine Boukerche and Jiahao Wang. Machine learning-based traffic prediction models for intelligent transportation systems. *Computer Networks*, 181:107530, 2020.

[68] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. Outlier detection: Methods, models, and classification. *ACM Computing Surveys (CSUR)*, 53(3):1–37, 2020.

[69] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[70] Valeria Cardellini, Vittoria De Nitto Personé, Valerio Di Valerio, Francisco Facchinei, Vincenzo Grassi, Francesco Lo Presti, and Veronica Piccialli. A game-theoretic approach to computation offloading in mobile cloud computing. *Mathematical Programming*, 157(2):421–449, 2016.

[71] Clayson Celes, Fabricio A Silva, Azzedine Boukerche, Rossana Maria de Castro Andrade, and Antonio AF Loureiro. Improving vanet simulation with calibrated vehicular mobility traces. *IEEE Transactions on Mobile Computing*, 16(12):3376–3389, 2017.

[72] Alberto Ceselli, Marco Premoli, and Stefano Secci. Cloudlet network design optimization. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2015.

[73] Jiageng Chen, Chunhua Su, Kuo-Hui Yeh, and Moti Yung. Special issue on advanced persistent threat. *Future Generation Computer Systems*, 79:243–246, 2018.

[74] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24(5):2795–2808, 2016.

[75] Sarah Clinch, Jan Harkes, Adrian Friday, Nigel Davies, and Mahadev Satyanarayanan. How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users. In *2012 IEEE international conference on pervasive computing and communications*, pages 122–127. IEEE, 2012.

[76] Sergio Correia, Azzedine Boukerche, and Rodolfo I Meneguette. An architecture for hierarchical software-defined vehicular networks. *IEEE Communications Magazine*, 55(7):80–86, 2017.

[77] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[78] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Gedar: Geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks. In *2014 IEEE International Conference on communications (ICC)*, pages 251–256. IEEE, 2014.

[79] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Geographic and opportunistic routing for underwater sensor networks. *IEEE Transactions on Computers*, 65(2):548–561, 2015.

[80] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. A novel void node recovery paradigm for long-term underwater sensor networks. *Ad Hoc Networks*, 34:144–156, 2015.

[81] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Design guidelines for opportunistic routing in underwater networks. *IEEE Communications Magazine*, 54(2):40–48, 2016.

[82] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Underwater wireless sensor networks: A new challenge for topology control–based systems. *ACM Computing Surveys (CSUR)*, 51(1):1–36, 2018.

[83] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62, 2010.

[84] Felipe Cunha, Leandro Villas, Azzedine Boukerche, Guilherme Maia, Aline Viana, Raquel AF Mini, and Antonio AF Loureiro. Data communication in vanets: Protocols, applications and challenges. *Ad Hoc Networks*, 44:90–103, 2016.

[85] Felipe Domingos Da Cunha, Azzedine Boukerche, Leandro Villas, Aline Carneiro Viana, and Antonio AF Loureiro. Data communication in vanets: a survey, challenges and applications. 2014.

[86] Amir Darehshoorzadeh and Azzedine Boukerche. Underwater sensor networks: A new challenge for opportunistic routing protocols. *IEEE Communications Magazine*, 53(11):98–107, 2015.

[87] Horacio Antonio Braga Fernandes De Oliveira, Azzedine Boukerche, Eduardo Freire Nakamura, and Antonio Alfredo Ferreira Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Transactions on Computers*, 58(5):677–691, 2008.

[88] Shuiguang Deng, Longtao Huang, Javid Taheri, and Albert Y Zomaya. Computation offloading for service workflow in mobile cloud computing. *IEEE transactions on parallel and distributed systems*, 26(12):3317–3329, 2014.

[89] Savio Dimatteo, Pan Hui, Bo Han, and Victor OK Li. Cellular traffic offloading through wifi networks. In *2011 IEEE eighth international conference on mobile ad-hoc and sensor systems*, pages 192–201. IEEE, 2011.

[90] Fahad R Dogar, Peter Steenkiste, and Konstantina Papagiannaki. Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 107–122, January 2010.

[91] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[92] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols. In *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*, pages 18–27, 2006.

[93] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 165–172, 2006.

[94] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. A distributed fault identification protocol for wireless and mobile ad hoc networks. *Journal of parallel and distributed computing*, 68(3):321–335, 2008.

[95] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings*

of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, page 226–231. AAAI Press, 1996.

[96] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84–106, 2013. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[97] Zhipeng Gao, Qidong Jiao, Kaile Xiao, Qian Wang, Zijia Mo, and Yang Yang. Deep reinforcement learning based service migration strategy for edge computing. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 116–1165, 2019.

[98] Marisol García-Valls, Abhishek Dubey, and Vicent Botti. Introducing the new paradigm of social dispersed computing: Applications, technologies and challenges. *Journal of Systems Architecture*, 91:83–102, 2018.

[99] Sahil Garg, Kuljeet Kaur, Shalini Batra, Georges Kaddoum, Neeraj Kumar, and Azzedine Boukerche. A multi-stage anomaly detection scheme for augmenting the security in iot-enabled applications. *Future Generation Computer Systems*, 104:105–118, 2020.

[100] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[101] Baraq Ghaleb, Ahmed Y Al-Dubai, Elias Ekonomou, Ayoub Alsarhan, Youssef Nasser, Lewis M Mackenzie, and Azzedine Boukerche. A survey of limitations and enhancements of the ipv6 routing protocol for low-power and lossy networks: A focus on core operations. *IEEE Communications Surveys & Tutorials*, 21(2):1607–1635, 2018.

[102] Shichao Guan and Azzedine Boukerche. Design and implementation of offloading and resource management techniques in a mobile cloud environment. In *Proceedings of the 17th ACM International Symposium on Mobility Management and Wireless Access*, pages 97–102, 2019.

[103] Shichao Guan and Azzedine Boukerche. A mec-based distributed offloading model for ubiquitous and time-constraint offloading. In *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8. IEEE, 2019.

[104] Xinjie Guan and Baek-Young Choi. Push or pull? toward optimal content delivery using cloud storage. *J. Netw. Comput. Appl.*, 40(C):234–243, apr 2014.

[105] Kiryong Ha, Padmanabhan Pillai, Wolfgang Richter, Yoshihisa Abe, and Mahadev Satyanarayanan. Just-in-time provisioning for cyber foraging. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 153–166, 2013.

[106] Hadi Habibzadeh, Tolga Soyata, Burak Kantarci, Azzedine Boukerche, and Cem Kaptan. Sensing, communication and security planes: A new challenge for a smart city system design. *Computer Networks*, 144:163–200, 2018.

[107] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.

[108] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity, 2019.

[109] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[110] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[111] Jintian Hu, Gaocai Wang, Xiaotong Xu, and Yuting Lu. Study on dynamic service migration strategy with energy optimization in mobile edge computing. *Mobile Information Systems*, 2019, 2019.

[112] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[113] Krittin Intharawijitr, Katsuyoshi Iida, and Hiroyuki Koga. Analysis of fog model considering computing and communication latency in 5g cellular networks. In *2016 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*, pages 1–4. IEEE, 2016.

[114] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[115] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[116] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *2012 Proceedings IEEE Infocom*, pages 945–953. IEEE, 2012.

[117] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[118] Adlen Ksentini, Tarik Taleb, and Min Chen. A markov decision process-based service migration procedure for follow me cloud. In *2014 IEEE International Conference on Communications (ICC)*, pages 1350–1354, 2014.

[119] Rafael Lazimy. Mixed-integer quadratic programming. *Mathematical Programming*, 22(1):332–349, 1982.

[120] Yujin Li and Wenye Wang. The unheralded power of cloudlet computing in the vicinity of mobile devices. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 4994–4999. IEEE, 2013.

[121] Jingchu Liu, Tao Zhao, Sheng Zhou, Yu Cheng, and Zhisheng Niu. Concert: a cloud-based architecture for next-generation cellular systems. *IEEE Wireless Communications*, 21(6):14–22, 2014.

[122] Kaiyang Liu, Jun Peng, Heng Li, Xiaoyong Zhang, and Weirong Liu. Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing. *Future Generation Computer Systems*, 64:1–14, 2016.

[123] Xiaolan Liu, Jiadong Yu, Zhiyong Feng, and Yue Gao. Multi-agent reinforcement learning for resource allocation in iot networks with edge computing. *China Communications*, 17(9):220–236, 2020.

[124] Felicia Lobillo, Zdenek Becvar, Miguel Angel Puente, Pavel Mach, Francesco Lo Presti, Fabrizio Gambetti, Mariana Goldhamer, Josep Vidal, Anggoro K Widiawan, and Emilio Calvanesse. An architecture for mobile computation offloading on cloud-enabled lte small cells. In *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6. IEEE, 2014.

[125] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, 19(3):1628–1656, 2017.

[126] Andrew Machen, Shiqiang Wang, Kin K. Leung, Bong Jun Ko, and Theodoros Salonidis. Migrating running applications across mobile edge clouds: Poster. MobiCom '16, page 435–436, New York, NY, USA, 2016. Association for Computing Machinery.

[127] Andrew Machen, Shiqiang Wang, Kin K Leung, Bong Jun Ko, and Theodoros Salonidis. Migrating running applications across mobile edge clouds: poster. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 435–436, 2016.

[128] Andrew Machen, Shiqiang Wang, Kin K Leung, Bong Jun Ko, and Theodoros Salonidis. Live service migration in mobile edge clouds. *IEEE Wireless Communications*, 25(1):140–147, 2017.

[129] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. *Theoretical Computer Science*, 442:13–21, 2012. Special Issue on the Workshop on Algorithms and Computation (WALCOM 2009).

[130] Abdelhamid Mammeri, Azzedine Boukerche, and Zongzhi Tang. A real-time lane marking localization, tracking and communication system. *Computer Communications*, 73:132–143, 2016.

[131] Anahit Martirosyan, Azzedine Boukerche, and Richard Pazzi. A taxonomy of cluster-based routing protocols for wireless sensor networks. In *2008 International Symposium on Parallel Architectures, Algorithms, and Networks (i-span 2008)*, pages 247–253. IEEE, 2008.

[132] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[133] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.

[134] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[135] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[136] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[137] Manuel Eugenio Morocho-Cayamcela, Haeyoung Lee, and Wansu Lim. Machine learning for 5g/b5g mobile and wireless communications: Potential, limitations, and future directions. *IEEE Access*, 7:137184–137206, 2019.

[138] Michael C Mozer. A focused backpropagation algorithm for temporal. *Backpropagation: Theory, architectures, and applications*, 137, 1995.

[139] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, and Vikas Kumar. Security and privacy in fog computing: Challenges. *IEEE Access*, 5:19293–19304, 2017.

[140] Apollinaire Nadembega, Abdelhakim Hafid, and Tarik Taleb. A destination and mobility path prediction scheme for mobile networks. *IEEE Transactions on Vehicular Technology*, 64(6):2577–2590, 2015.

[141] Apollinaire Nadembega, Abdelhakim Senhaji Hafid, and Ronald Brisebois. Mobility prediction model-based service migration procedure for follow me cloud to support qos and qoe. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016.

[142] Christopher Olah. Understanding lstm networks, 2015.

[143] Horacio ABF Oliveira, Azzedine Boukerche, Eduardo F Nakamura, and Antonio AF Loureiro. Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm. *Performance Evaluation*, 66(3-5):209–222, 2009.

[144] Horacio ABF Oliveira, Eduardo F Nakamura, Antonio AF Loureiro, and Azzedine Boukerche. Error analysis of localization systems for sensor networks. In *Proceedings*

of the 13th annual ACM international workshop on Geographic information systems, pages 71–78, 2005.

[145] Renê Oliveira, Carlos Montez, Azzedine Boukerche, and Michelle S Wangham. Reliable data dissemination protocol for vanet traffic safety applications. *Ad Hoc Networks*, 63:30–44, 2017.

[146] Tao Ouyang, Zhi Zhou, and Xu Chen. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. *IEEE Journal on Selected Areas in Communications*, 36(10):2333–2345, 2018.

[147] Sung Woon Park, Azzedine Boukerche, and Shichao Guan. A novel deep reinforcement learning based service migration model for mobile edge computing. In *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8, 2020.

[148] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks, 2014.

[149] Richard W Pazzi and Azzedine Boukerche. Propane: A progressive panorama streaming protocol to support interactive 3d virtual environment exploration on graphics-constrained devices. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(1):1–22, 2014.

[150] Richard WN Pazzi and Azzedine Boukerche. Mobile data collector strategy for delay-sensitive applications over wireless sensor networks. *Computer Communications*, 31(5):1028–1039, 2008.

[151] Wang Peizhuang. Pattern recognition with fuzzy objective function algorithms (james c. bezdek). *SIAM Rev.*, 25(3):442, jul 1983.

[152] Alberto Del Pia, Santanu S Dey, and Marco Molinaro. Mixed-integer quadratic programming is in np. *Mathematical Programming*, 162(1):225–240, 2017.

[153] D. Praveen Kumar, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu. Machine learning algorithms for wireless sensor networks: A survey. *Information Fusion*, 49:1–25, 2019.

[154] Lingjun Pu, Xu Chen, Jingdong Xu, and Xiaoming Fu. D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration. *IEEE Journal on Selected Areas in Communications*, 34(12):3887–3901, 2016.

[155] Attia Qamar, Ahmad Karim, and Victor Chang. Mobile malware attacks: Review, taxonomy and future directions. *Future Generation Computer Systems*, 97:887–909, 2019.

[156] J. R Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[157] Anant Ram, Ashish Sharma, Anand S. Jalal, Ankur Agrawal, and Raghuraj Singh. An enhanced density based spatial clustering of applications with noise. In *2009 IEEE International Advance Computing Conference*, pages 1475–1478, 2009.

[158] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

[159] Yonglin Ren and Azzedine Boukerche. Modeling and managing the trust for wireless and mobile ad hoc networks. In *2008 IEEE International Conference on Communications*, pages 2129–2133. IEEE, 2008.

[160] Yonglin Ren, Richard Werner, Nelem Pazzi, and Azzedine Boukerche. Monitoring patients via a secure and mobile healthcare system. *IEEE Wireless Communications*, 17(1):59–65, 2010.

[161] Cristiano Rezende, Azzedine Boukerche, Heitor S Ramos, and Antonio AF Loureiro. A reactive and scalable unicast solution for video streaming over vanets. *IEEE Transactions on Computers*, 64(3):614–626, 2014.

[162] Cristiano Rezende, Abdelhamid Mammeri, Azzedine Boukerche, and Antonio AF Loureiro. A receiver-based video dissemination solution for vehicular networks with content transmissions decoupled from relay node selection. *Ad Hoc Networks*, 17:1–17, 2014.

[163] E Robson and Azzedine Boukerche. Dynamic balancing of communication and computation load for hla-based simulations on large-scale distributed systems. *Journal of Parallel and Distributed Computing*, 71(1):40–52, 2011.

[164] Tiago Gama Rodrigues, Katsuya Suto, Hiroki Nishiyama, Nei Kato, and Katsuhiro Temma. Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration. *IEEE Transactions on Computers*, 67(9):1287–1300, 2018.

[165] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[166] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[167] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Transactions on Vehicular Technology*, 60(2):656–663, 2010.

[168] Mohamed Sana, Antonio De Domenico, Wei Yu, Yves Lostanlen, and Emilio Calvanese Strinati. Multi-agent reinforcement learning for adaptive user association in dynamic mmwave networks. *IEEE Transactions on Wireless Communications*, 19(10):6520–6534, 2020.

[169] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[170] Mahadev Satyanarayanan, Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, Wenlu Hu, and Brandon Amos. Edge analytics in the internet of things. *IEEE Pervasive Computing*, 14(2):24–31, 2015.

[171] Stefan Schmid, Roger Wattenhofer, and Azzedine Boukerche. Modeling sensor networks. *Algorithms and Protocols for Wireless Sensor Networks*, 62:77, 2008.

[172] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[173] Mohit Sewak. *Deep Q Network (DQN), Double DQN, and Dueling DQN*, pages 95–108. Springer Singapore, Singapore, 2019.

[174] Fabricio A Silva, Azzedine Boukerche, Thais RM Braga Silva, Linnyer B Ruiz, Eduardo Cerqueira, and Antonio AF Loureiro. Vehicular networks: A new challenge for content-delivery-based applications. *ACM Computing Surveys (CSUR)*, 49(1):1–29, 2016.

[175] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[176] Gaurav Somani, Manoj Singh Gaur, Dheeraj Sanghi, Mauro Conti, and Rajkumar Buyya. Ddos attacks in cloud computing: Issues, taxonomy, and future directions. *Computer Communications*, 107:30–48, 2017.

[177] Ivan Stojmenovic and Sheng Wen. The fog computing paradigm: Scenarios and security issues. In *2014 federated conference on computer science and information systems*, pages 1–8. IEEE, 2014.

[178] R Suganya and R Shanthi. Fuzzy c-means algorithm-a review. *International Journal of Scientific and Research Publications*, 2(11):1, 2012.

[179] Xiang Sun and Nirwan Ansari. Primal: Profit maximization avatar placement for mobile edge computing. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016.

[180] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[181] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction.* Adaptive Computation and Machine Learning series. MIT Press, 2018.

[182] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[183] Tarik Taleb and Adlen Ksentini. Follow me cloud: interworking federated clouds and distributed mobile networks. *IEEE Network*, 27(5):12–19, 2013.

[184] Tarik Taleb, Adlen Ksentini, and Pantelis A Frangoudis. Follow-me cloud: When cloud services follow mobile users. *IEEE Transactions on Cloud Computing*, 7(2):369–382, 2016.

[185] Mati B Terefe, Heezin Lee, Nojung Heo, Geoffrey C Fox, and Sangyoon Oh. Energy-efficient multisite offloading policy using markov decision process for mobile cloud computing. *Pervasive and Mobile Computing*, 27:75–89, 2016.

[186] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016.

[187] Pierre van Moerbeke. Optimal stopping and free boundary problems. *The Rocky Mountain Journal of Mathematics*, 4(3):539–578, 1974.

[188] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, MCS '12, page 29–36, New York, NY, USA, 2012. Association for Computing Machinery.

[189] Leandro A Villas, Azzedine Boukerche, Horacio ABF De Oliveira, Regina B De Araujo, and Antonio AF Loureiro. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, 12:69–85, 2014.

[190] Leandro A Villas, Azzedine Boukerche, Daniel L Guidoni, Horacio ABF De Oliveira, Regina Borges De Araujo, and Antonio AF Loureiro. An energy-aware spatio-temporal correlation mechanism to perform efficient data collection in wireless sensor networks. *Computer Communications*, 36(9):1054–1066, 2013.

[191] Leandro Aparecido Villas, Azzedine Boukerche, Guilherme Maia, Richard Werner Pazzi, and Antonio AF Loureiro. Drive: An efficient and robust data dissemination protocol for highway and urban vehicular ad hoc networks. *Computer Networks*, 75:381–394, 2014.

[192] Leandro Aparecido Villas, Azzedine Boukerche, Heitor Soares Ramos, Horacio AB Fernandes De Oliveira, Regina Borges de Araujo, and Antonio Alfredo Ferreira Loureiro. Drina: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. *IEEE Transactions on Computers*, 62(4):676–689, 2012.

[193] Kaiqiang Wang, Minwei Shen, Junguk Cho, Arijit Banerjee, Jacobus Van der Merwe, and Kirk Webb. Mobiscud: A fast moving personal cloud in the mobile network. In

Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges, pages 19–24, 2015.

[194] Shangguang Wang, Jinliang Xu, Ning Zhang, and Yujiong Liu. A survey on service migration in mobile edge computing. *IEEE Access*, 6:23511–23528, 2018.

[195] Shiqiang Wang, Rahul Urgaonkar, Ting He, Murtaza Zafer, Kevin Chan, and Kin K. Leung. Mobility-induced service migration in mobile micro-clouds. In *2014 IEEE Military Communications Conference*, pages 835–840, 2014.

[196] Shiqiang Wang, Rahul Urgaonkar, Murtaza Zafer, Ting He, Kevin Chan, and Kin K Leung. Supplementary materials for dynamic service migration in mobile edge-clouds.

[197] Shiqiang Wang, Rahul Urgaonkar, Murtaza Zafer, Ting He, Kevin Chan, and Kin K. Leung. Dynamic service migration in mobile edge-clouds. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2015.

[198] Xiumin Wang, Jin Wang, Xin Wang, and Xiaoming Chen. Energy and delay tradeoff for application offloading in mobile cloud computing. *IEEE Systems Journal*, 11(2):858–867, 2017.

[199] Yichuan Wang, He Zhu, Xinhong Hei, Yue Kong, Wenjiang Ji, and Lei Zhu. An energy saving based on task migration for mobile edge computing. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):133, May 2019.

[200] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR.

[201] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[202] Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 12 1990.

[203] Hongjia Wu, Gaocai Wang, Xiaotong Xu, and Jintian Hu. A cache placement strategy for energy savings in ccn. In *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 788–795, 2018.

[204] Na Xia, Mei Tang, Jian-guo Jiang, Dun Li, and Hao-wei Qian. Energy efficient data transmission mechanism in wireless sensor networks. In *2008 International Symposium on Computer Science and Computational Technology*, volume 1, pages 216–219. IEEE, 2008.

[205] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[206] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4):611–629, 2018.

[207] Yingjie Yang, Xin Chen, Ying Chen, and Zhuo Li. Green-oriented offloading and resource allocation by reinforcement learning in mec. In *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 378–382, 2019.

[208] Hong Yao, Changmin Bai, Deze Zeng, Qingzhong Liang, and Yuanyuan Fan. Migrate or not? exploring virtual machine migration in roadside cloudlet-based vehicular cloud. *Concurrency and Computation: Practice and Experience*, 27(18):5780–5792, 2015.

[209] Ibrar Yaqoob, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, Muhammad Imran, and Sghaier Guizani. Mobile ad hoc cloud: A survey. *Wireless Communications and Mobile Computing*, 16(16):2572–2589, 2016.

[210] Changsheng You, Kaibin Huang, and Hyukjin Chae. Energy efficient mobile cloud computing powered by wireless energy transfer. *IEEE Journal on Selected Areas in Communications*, 34(5):1757–1771, 2016.

[211] Xinyu You, Xuanjie Li, Yuedong Xu, Hui Feng, Jin Zhao, and Huaicheng Yan. Toward packet routing with fully distributed multiagent deep reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–14, 2020.

[212] Maram Bani Younes and Azzedine Boukerche. Intelligent traffic light controlling algorithms using vehicular networks. *IEEE transactions on vehicular technology*, 65(8):5887–5899, 2015.

[213] Maram Bani Younes and Azzedine Boukerche. A performance evaluation of an efficient traffic congestion detection protocol (ecode) for intelligent transportation systems. *Ad Hoc Networks*, 24:317–336, 2015.

[214] Maram Bani Younes and Azzedine Boukerche. An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems. *Wireless Networks*, 24:2451–2463, 2018.

[215] Rong Yu, Yan Zhang, Huimin Wu, Periklis Chatzimisios, and Shengli Xie. Virtual machine live migration for pervasive services in cloud-assisted vehicular networks. In *2013 8th International Conference on Communications and Networking in China (CHINACOM)*, pages 540–545, 2013.

[216] Deze Zeng, Lin Gu, Song Guo, Zixue Cheng, and Shui Yu. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Transactions on Computers*, 65(12):3702–3712, 2016.

[217] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys Tutorials*, 21(3):2224–2287, 2019.

[218] Cheng Zhang and Zixuan Zheng. Task migration for mobile edge computing using deep reinforcement learning. *Future Generation Computer Systems*, 96:111–118, 2019.

[219] Yang Zhang, Dusit Niyato, and Ping Wang. Offloading in mobile cloudlet systems with intermittent connectivity. *IEEE Transactions on Mobile Computing*, 14(12):2516–2529, 2015.

[220] Zhenxia Zhang, Azzedine Boukerche, and Richard Pazzi. A novel multi-hop clustering scheme for vehicular ad-hoc networks. In *Proceedings of the 9th ACM international symposium on Mobility management and wireless access*, pages 19–26, 2011.

[221] Zhenxia Zhang, Richard W Pazzi, and Azzedine Boukerche. A mobility management scheme for wireless mesh networks based on a hybrid routing protocol. *Computer Networks*, 54(4):558–572, 2010.