Intelligent Resource Allocation Scheme for Throughput Enhancement in 6G Networks: A Smart Hospital Perspective

Project report submitted to the Amrita Vishwa Vidyapeetham in partial fulfilment of the requirement for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Hrishika Dayan AM.EN.U4CSE20234
Krishnapriya V S AM.EN.U4CSE20243
Namita Suresh AM.EN.U4CSE20250
Srilakshmi S R AM.EN.U4CSE20269

under the guidance of

Dr.Simi Surendran

AMRITA SCHOOL OF COMPUTING
AMRITA VISHWA VIDYAPEETHAM
(Estd. U/S 3 of the UGC Act 1956)
AMRITAPURI CAMPUS
KOLLAM - 690525

MAY 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM
(Estd. U/S 3 of the UGC Act 1956)

Amritapuri Campus

Kollam -690525

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"Intelligent Resource Allocation Scheme for Throughput Enhancement in 6G Networks: A Smart Hospital Perspective"** submitted by **Hrishika Dayan(AM.EN.U4CSE20234), Krishnapriya V S(AM.EN.U4CSE20243), Namita Suresh(AM.EN.U4CSE20250) and Srilakshmi S R(AM.EN.U4CSE20269)**, in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering from Amrita Vishwa Vidyapeetham, is a bonafide record of the work carried out by them under my guidance and supervision at Amrita School of Computing, Amritapuri during Semester 8 of the academic year 2023-2024.

Dr.Simi Surendran
Project Guide

Project Coordinator

Dr. Swaminathan J
Chairperson
Dept. of Computer Science & Engineering

Reviewer

Place : Amritapuri
Date : 10 May 2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

(Estd. U/S 3 of the UGC Act 1956)

Amritapuri Campus

Kollam -690525

DECLARATION

We, **Hrishika Dayan(AM.EN.U4CSE20234), Krishnapriya V S(AM.EN.U4CSE20243), Namita Suresh(AM.EN.U4CSE20250) and Srilakshmi S R(AM.EN.U4CSE20269)** hereby declare that this project entitled **"Intelligent Resource Allocation Scheme for Throughput Enhancement in 6G Networks: A Smart Hospital Perspective"** is a record of the original work done by us under the guidance of **Dr. Simi Surendran**, Dept. of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, that this work has not formed the basis for any degree/diploma/associationship/fellowship or similar awards to any candidate in any university to the best of our knowledge.

Place : Amritapuri

Date : 10 May 2024

Signature of the student

Signature of the Project Guide

Acknowledgements

We would like to express our sincere gratitude to the Department of Computer Science and Engineering at Amrita School of Computing, Amrita Vishwa Vidyapeetham, for their unwavering support throughout this project. The department provided us with the essential infrastructure and resources that greatly facilitated our research endeavors.

A special note of thanks goes to our esteemed project guide, Dr. Simi Surendran. Her invaluable insights, expertise, and guidance were instrumental at every stage of our research. We are deeply grateful for her constant encouragement and support.

Finally, we acknowledge the invaluable contributions of all the participants involved in our study. Without their participation, the successful completion of this research would not have been possible.

Abstract

The advent of smart hospital technologies and the proliferation of the Internet of Things (IoT) has revolutionized healthcare, generating massive data demands and necessitating efficient resource allocation. This paper proposes an intelligent resource allocation scheme for 6G networks in smart hospitals, leveraging Deep Reinforcement Learning (DRL). We delve into the development of a DRL-powered approach to optimize network parameters and dynamically allocate resources like bandwidth and computing power that cater to the unique needs of various healthcare applications. This ensures that Quality of Service (QoS) requirements are met in diverse scenarios within smart hospitals. By prioritizing high-speed data transfer, improving energy efficiency, and optimizing network capacity, our research aims to foster efficient and responsive smart hospital environments in the 6G era. Ultimately, these endeavors empower healthcare professionals with the necessary tools to deliver high-quality care and robust patient monitoring, thereby advancing the landscape of healthcare delivery in the digital age. Furthermore, we compare the performance of Deep Deterministic Policy Gradient (DDPG) and Deep Q-Network (DQN) algorithms for the same use case. The proposed intelligent resource allocation scheme is evaluated based on learning capacity, throughput, and edge capability, providing insights into optimizing resource allocation within smart hospital environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the rapidly evolving landscape of healthcare, the integration of smart devices and technologies within hospitals has become commonplace. These devices, ranging from wearable health monitors to telemedicine applications, demand a sophisticated and efficient allocation of network resources to ensure seamless connectivity and optimal performance. This is essential for providing timely and effective healthcare services to patients, who often require continuous monitoring and support. The growing demand for data in healthcare, driven by the need for real-time monitoring, remote consultations, and data-driven decision-making, is further fueling the adoption of smart devices.

The integration of these smart devices and technologies into healthcare facilities is transforming the way healthcare is delivered to patients. However, this transformation comes with its own set of challenges, particularly in terms of efficiently managing network resources to ensure smooth connectiv-

ity and optimal performance of these devices. To overcome these challenges, a thorough grasp of the resource allocation needs in smart environments like hospitals and care homes is essential. Additionally, it demands the creation of adaptable and innovative solutions capable of meeting the dynamic demands of healthcare settings. The demand is being met by the development of advanced networking technologies such as 5G and the anticipated 6G networks, which promise faster speeds, lower latency, and higher bandwidth capacity. By efficiently managing network resources and leveraging advanced networking technologies, healthcare providers can ensure seamless connectivity and optimal performance of smart devices, ultimately enhancing the overall healthcare experience for individuals.

1.2 Research Objectives

Smart hospitals face dynamic needs due to evolving technologies, fluctuating numbers of connected devices, and rapidly changing patient care requirements. This dynamic environment necessitates real-time adjustments in resource allocation to ensure critical applications receive the necessary resources. Seamless interoperability among various devices and systems adds to the complexity, requiring flexible resource management strategies. To address these challenges, smart hospitals require adaptive resource allocation schemes capable of responding to changing conditions promptly.

The objective of this project is to devise a smart solution that accommodates the diverse needs of these devices, ensuring they operate seamlessly within the hospital network. The backbone of this digital transformation of hospitals lies in the capabilities of 5G and the evolving 6G cellular networks, with a specific focus on Enhanced Mobile Broadband (eMBB). This

fundamental service is essential for supporting mission-critical applications and services within smart hospital environments. This project uses Deep Reinforcement Learning (DRL) to dynamically allocate resources, catering to the unique demands of a complex and dynamic hospital ecosystem.

The persisting challenges in addressing the problem of intelligent resource allocation in smart hospitals using 6G networks are multifaceted. Firstly, smart hospitals consist of a diverse range of applications, devices, and services, each with unique resource requirements and QoS demands. Designing a resource allocation scheme that can effectively handle this complexity is a formidable challenge. Secondly, healthcare environments are dynamic and subject to fluctuations in demand. Resource allocation must adapt in real-time to varying requirements, ensuring critical applications receive the necessary resources while efficiently utilizing available network capacity.

As healthcare facilities embrace digital transformation, this project focuses on addressing challenges like network complexity and real-time adaptation to dynamic demands. DRL algorithms optimize network parameters, adapting in real time to changing conditions. The contribution of this work is to enhance resource allocation efficiency in smart hospitals by developing a deep reinforcement learning-based scheme that optimizes the allocation of bandwidth. To achieve the objective, a thorough analysis of the requirements for resource allocation in smart hospitals has been conducted, considering factors such as bandwidth and latency.

1.3 Organisation of the Report

The report is structured to provide a comprehensive understanding of efficient resource management in healthcare, particularly in the context of smart

hospitals. It begins with an Introduction that underscores the critical role of optimal resource allocation in supporting the dynamic and diverse needs of modern healthcare settings. Research Objectives are outlined to clarify the study's aims, emphasizing the significance and relevance of addressing resource allocation challenges in smart hospitals. The report then details the organization of subsequent chapters, providing a road-map for the reader.

The Problem Definition chapter elucidates the specific challenges in resource allocation faced by smart hospitals, focusing on factors such as the dynamic nature of healthcare environments and the diverse requirements of different applications and devices. The Related Work section reviews existing literature on resource allocation using RL, highlighting gaps and insights crucial for the study. It also examines research on 5G and 6G networks, emphasizing their potential to optimize resource allocation in smart hospitals. The Requirements chapter discusses the various software components and its specifications.

The Adaptive Resource Allocation in 6G Networks using DRL chapter comprises of 3 sub-sections. Firstly, the Architecture section discusses how the smart hospital architecture routes communication requests from mobile units to edge server. In resource-constrained scenarios, a DRL algorithm optimizes resource allocation, seamlessly migrating requests to the cloud for scalability, ensuring adaptability, and enhancing efficiency. Secondly, it covers the MDP formulation that consists of the State Space, Action Space, Reward Function and Discount Factor. The latter subsection covers the DQN and DDPG-based Resource Allocation (DRA) Algorithms and assesses its performance in adapting to dynamic environments, handling complex action spaces, and achieving optimal resource utilization.

The Results and Analysis chapter evaluates the performance of the pro-

posed resource allo- cation algorithm using two reinforcement approaches: Deep Deterministic Policy Gradient (DDPG) and Deep Q-Network (DQN) based on cumulative reward acquired during the learning process, throughput, and edge capability. It also studies the effect of number of users and edge servers in the throughput and edge capability exhibited by this algorithm.

Finally, the Conclusion chapter synthesizes key findings, reflecting on the study's contributions and limitations. It discusses how the devised MDP framework for the smart hospital environment combined with the proposed DRA algorithm has shown encouraging outcomes in optimizing resource allocation.

Chapter 2

Problem Definition

The ever-growing adoption of Internet of Things (IoT) devices in smart hospitals presents a significant challenge for network infrastructure. These devices, such as wearable health monitors and sensor-equipped medical equipment, necessitate high-speed internet access to support a large number of devices and manage connections effectively. Smart hospitals are expected to house a multitude of low-power, low-data-rate devices, each requiring reliable connectivity. Traditional networks may struggle to handle such a high volume of connections efficiently. Efficient connection management is crucial for ensuring the smooth operation of these devices and the timely transmission of critical healthcare data. Current network architectures often lack the necessary flexibility to cater to the diverse needs of a smart hospital environment. While some applications prioritize high bandwidth for data transmission (e.g., remote consultations with high-resolution medical images), others might require ultra-reliable and low-latency communication for real-time patient monitoring or remote surgery control.

This project aims to address these challenges by proposing an intelligent resource allocation scheme specifically designed for smart hospitals leverag-

ing 6G networks. By utilizing Deep Reinforcement Learning (DRL), the proposed scheme can dynamically optimize network parameters and allocate resources like bandwidth and computing power to cater to the unique needs of various healthcare applications. This ensures that Quality of Service (QoS) requirements are met in diverse scenarios within smart hospitals. Our research focuses on comparing the performance of Deep Deterministic Policy Gradient (DDPG) and Deep Q-Network (DQN) algorithms for this use case. The ultimate goal is to empower healthcare professionals with the necessary tools to deliver high-quality care and robust patient monitoring, fostering a more efficient and responsive smart hospital environment in the 6G era.

Chapter 3

Related Work

Analysing the capabilities of 5G and 6G networks is essential for understanding their technological advancements and the new opportunities they bring. These networks offer higher data rates, lower latency, increased device connectivity, and improved energy efficiency. Salameh and El Tarhuni discuss the transition from 5G to 6G, emphasizing the need for faster and more reliable communication to support automation and smart cities[10]. Jiang et al. survey 6G mobile communication systems, anticipating the first 6G network around 2030, enabling new applications like holographic communications and pervasive intelligence. In the realm of 6G MEC, Sami et al. proposed IScaler, an intelligent resource scaling and service placement solution for 6G Mobile Edge Computing using Deep Reinforcement Learning(DRL)[4] [1].

Extensive application of DRL can be seen in cutting-edge technologies such as 6G mobile edge computing (MEC), Internet of Things (IoT), medical imaging, and robotics. Wei et al. review how DRL optimizes Mobile Edge Computing (MEC) for 6G networks, addressing challenges in dynamic environments[2]. Zhang and Zheng propose a DQN-based technique for task migration in MEC systems[7]. Lu et al. propose a dynamic service placement

framework, DSP-DRL, based on deep reinforcement learning (DRL) for mobile edge computing. Their approach optimizes total delay while considering constraints on physical resources and operational costs[8]. Rui et al. propose a service migration method for MEC using state adaptation and deep reinforcement learning to reduce service interruptions and mitigate the impact of network failures[11]. Chen et al. address dynamic task allocation in edge-cloud IoT systems using DRL, minimizing load forwarded to the cloud[13]. Liu et al. propose a DRL-based algorithm for service migration and resource allocation in edge IoT systems to minimize access delay, demonstrating effectiveness using a real-world dataset of Beijing cab trajectories[12]. Upadhyay et al. present a dynamic approach using Deep Q-Networks (DQN) to manage Unmanned Aerial Vehicles (UAVs) in crowded environments[15].

DRL has also been instrumental in optimizing computation offloading in MEC and improving ultra-reliable and low-latency communications in 6G networks. Chen et al. propose a deep reinforcement learning (RL) approach (ddpg) for computation offloading in mobile edge computing (MEC) that improves resource provisioning performance for mobile devices in complex network environments[6]. Chen et al. propose a temporal attentional deterministic policy gradient (TADPG) method for jointly optimizing computation offloading and resource allocation (JCORA) in mobile edge computing[14]. Hortelano et al. surveyed RL in edge computing for offloading tasks from devices with limited computing capabilities[5]. Chen and Wang propose a deep reinforcement learning approach (ddpg) for decentralized computation offloading in multi-user MEC that aims to minimize computation cost in terms of power consumption and buffering delay[9]. Liu et al.investigate machine learning's role in enhancing ultra-reliable and low-latency communications (URLLC) in 6G networks[3].

In medical imaging, DRL has been used for skin lesion classification and brain aneurysm segmentation, demonstrating its potential to enhance accuracy and efficiency in diagnosis[17][19]. Moreover, DRL has been applied to eye-gaze behaviour tracking, offering non-invasive methods for studying visual attention.Deepalakshmi and Amudha propose a deep convolutional reinforcement learning (DC-RL) model for predicting visual attention behaviour over dynamic scenes, aiming to overcome challenges in invasive eye tracking methods and enhance the understanding of gaze behaviour during video playback [18]. In the context of network management, Doke and Sangeeta suggest using deep reinforcement learning for network load balancing in data centres, aiming to improve efficiency and reduce costs in managing big data traffic[16]. Additionally, DRL has enabled dynamic path planning for mobile service robots in dynamic environments, illustrating its adaptability and effectiveness in real-world scenarios. Kumaar and Kochuvila developed a deep reinforcement learning-based path planner for mobile service robots in dynamic environments, utilizing the beta-decay transfer learning algorithm to adapt to environmental changes, achieving a high success rate and faster convergence compared to non-transfer learning agents[20].

Despite the promising results, there are several common themes and areas for future research. These include the need for more comprehensive evaluation in real-world scenarios, consideration of multi-objective learning approaches, addressing computational complexity, comparison of DRL models and ensuring robustness and reliability in different environments.

Chapter 4

Requirements

4.1 Software Requirements

Integrated Development Environment (IDE): Visual Studio Code (VS Code)

Visual Studio Code (VS Code) is recommended as the primary IDE for development. VS Code is a free, open-source, and cross-platform code editor that offers excellent Python support through extensions.

Python

The project relies on Python for core functionality. Python version 3 is required, with 3.7 or later recommended to ensure compatibility with the included libraries.

4.2 Python Libraries:

NumPy

A fundamental library for scientific computing in Python. It provides powerful array and matrix manipulation capabilities, essential for processing network resource allocation data.

SciPy

Builds upon NumPy, offering additional functionalities for advanced scientific computing, including optimization and integration. These functionalities is used for specific mathematical calculations and model optimization within the resource allocation scheme.

TensorFlow

TensorFlow is a popular open-source machine learning library. Used to develop and train the deep learning model for intelligent resource allocation. TensorFlow offers a comprehensive framework for various machine learning tasks.

Matplotlib

A popular Python library for creating static, animated, and interactive visualizations. Used in generating plots analysis.

Chapter 5

Adaptive Resource Allocation in 6G networks using DRL

The emergence of 6G networks signifies a prominent milestone in the telecommunications landscape, demanding advanced resource management strategies to effectively address the diverse requirements of applications amidst the dynamic nature of network conditions. In response to this exigency, DRL has emerged as a promising avenue for adaptive resource allocation. The subsequent sub-sections describe the proposed architecture of the resource allocation scheme in smart hospitals, the design of MDP, and the DRL algorithm used to obtain the optimal allocation of resources are described in the subsequent subsections. To meet these demands, a hierarchical approach is employed for resource allocation. This approach ensures that each application receives the necessary resources tailored to its requirements.

Moreover, the project incorporates a comprehensive comparison between two prominent DRL algorithms, Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG). The comparison delves into their efficacy

in optimizing resource allocation within 6G networks for smart hospitals. By evaluating factors such as convergence speed, stability, and adaptability to varying network conditions, the project aims to identify the most suitable algorithm for achieving optimal resource utilization and enhancing network performance. This comparative analysis forms a crucial aspect of the project's endeavor to develop an intelligent resource allocation scheme that meets the evolving demands of 6G networks in smart environments.

5.1 Architecture of Smart Hospital Resource Allocation

Smart hospitals, in their pursuit of 6G capabilities, have specific requirements and service needs that demand advanced communication systems. These requirements include seamless connectivity for telemedicine, wearables device data analysis, emergency response, emergency vehicle support, entertainment, and information services required for staff such as accessing electronic medical records.

In this architecture (Fig. 5.1), when a mobile unit initiates a communication request, it is first routed to the base station. From there, the request is forwarded to the nearest edge server within the smart hospital network. At the edge server, a DRL algorithm is used to solve the MDP framework to determine the optimal resource allocation strategy based on the specific needs of the application. The edge server acts as a decentralized computing resource, efficiently handling local processing tasks. In cases where edge servers encounter challenges or resource constraints, a seamless migration strategy is implemented, transferring communication requests to the cloud server through the base station. This approach ensures scalability and flexi-

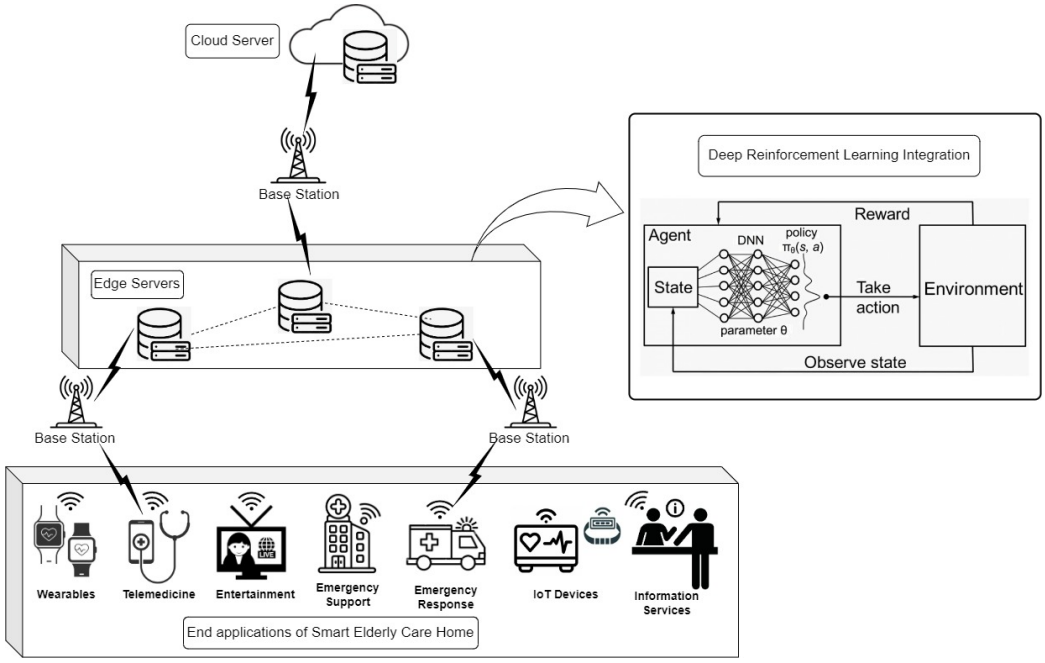


Figure 5.1: Smart Hospital Architecture Leveraging 6G Technology and Intelligent Resource Allocation Scheme

bility in resource allocation. The cloud server, with its higher computational and storage capabilities, serves as a supplementary resource, balancing the load between edge servers and the cloud. The integration of the DRL algorithm enhances adaptability and efficiency in resource allocation, supporting the dynamic and diverse needs of smart care applications.

5.2 MDP Model for Resource Allocation

Reinforcement Learning (RL) offers a powerful approach to resource allocation problems where the environment's dynamics are complex and may change over time. Unlike traditional methods, RL empowers an agent to learn through interacting with the environment. The agent receives rewards for beneficial decisions, as defined by a reward function reflecting allocation

efficiency. This learning is structured by modeling the environment as an MDP, which provides a framework for these interactions between an agent and its environment, thereby guiding the agent toward optimal resource allocation. In the context of resource allocation, an MDP formulation defines states representing the current situation, actions representing resource allocation decisions, transition probabilities capturing the dynamics of resource usage, and rewards reflecting the benefits of allocation decisions. By integrating RL within an MDP framework, resource allocation decisions can be made intelligently, considering the dynamic nature of the environment and maximizing long-term utility.

An MDP is defined by (S, A, R, P, γ), where S represents the set of states, A represents the set of actions, R represents the reward function and P represents the state transition probability matrix. The optimization objective of this MDP is to ensure an efficient allocation of resources to users and edge servers. To solve this issue we have designed an MDP using the following components:

5.2.1 State Space

The state space captures the relevant information about the system, including server resources, bandwidth, offloading targets, user locations, the amount of data transmitted, network load and CPU utilization of each server. It is represented as a 6-tuple

$$\{< \mathrm{CPU}_{\mathrm{util}}, P_{\mathrm{avail}}, B_{\mathrm{avail}}, D_{\mathrm{trans}}, N_{\mathrm{users}}, T_{\mathrm{off}}\}$$

where

$$\begin{aligned} \mathrm{CPU}_{\mathrm{util}} &: \text{CPU utilization of each edge server} \\ P_{\mathrm{avail}} &: \text{Available computing resources} \\ B_{\mathrm{avail}} &: \text{Available bandwidth of each edge server} \\ D_{\mathrm{trans}} &: \text{Data Transmitted} \\ N_{\mathrm{users}} &: \text{Current load of the network} \\ T_{\mathrm{off}} &: \text{Offloading target of each mobile user} \end{aligned}$$

- CPU utilization of each edge server refers to the percentage of the server's processing capacity that is currently being utilized. It provides insight into how efficiently the edge servers are handling computational tasks and can help in managing resource allocation and workload distribution.
- Available computing resources of each edge server refer to the computational capacity or processing power of each edge server.
- The Available bandwidth of each connection between edge servers indicates how much data can be transferred between edge servers for the purpose of offloading tasks or sharing computational resources.
- Data transmitted denotes the amount of data that is being transferred, possibly between mobile users and edge servers.
- The Current load of the network in terms of the number of connected users represents the number of users currently connected to the network. It indicates the level of activity and demand on the network

resources at a given time.

- Offloading target denotes where a mobile user's computational tasks or data processing should be performed, in which edge server.

5.2.2 Action Space

The action space represents the decisions that the system can make. In this case, it involves allocating computing resources, migration bandwidth, and determining the offloading target for each mobile user. The action space is represented as a three tuple

$$\{< A_b, A_p, O_t >\}$$

where

A_b : Bandwidth allocation for each user

A_p : Computing resource allocation for each user

O_t : Offloading target of each user

- "Allocating computing resources for each mobile user's task" refers to the process of assigning or dedicating computational capacity to the execution of tasks associated with a specific mobile user.
- "Allocating bandwidth for each mobile user's task" action involves deciding how much network bandwidth should be allocated to facilitate the transfer of data between the mobile user's device and edge server.
- "Offloading target of each mobile user" action determines where the

computational task of the user should be processed.

5.2.3 Reward Function

A reward function in RL acts as a critical guide for an agent, assigning numerical values to its actions or state transitions to encourage desired behavior. In the context of our project, the reward function is tailored to optimize the efficiency of data transmission processes. Focused on maximizing throughput, the function evaluates the success of data delivery by considering the difference between requested data and that which could not be transmitted. The algorithm's goal is to learn a policy that maximizes this throughput, striking a balance with the inherent capability of edge servers. By fine-tuning a weighting parameter μ, we aim to emphasize the importance of efficient data transmission while ensuring the optimal allocation of computational resources at the edge.

$$d_{\text{trans}} = d_{\text{req}} - d_{\neg\text{trans}} \tag{5.1}$$

where d_{trans} represents the amount of data transmitted, d_{req} denotes the total amount of data requested, and $d_{\neg\text{trans}}$ denotes the amount of data that could not be transmitted.

$$T_{\text{avg}} = \frac{\sum_{i=1}^{n} d_{\text{trans}}}{n} \tag{5.2}$$

where T_{avg} represents the average throughput, providing a measure of the overall efficiency or success of the data transmission process and n denotes

the total number of episodes.

$$R = \mu \cdot T_{\text{avg}} + (1 - \mu) \cdot E_c \tag{5.3}$$

where R represents the overall reward function and E_c denotes Edge capability, referring to the capacity or capability of the edge server. The reward function is a combination of two components: the average throughput, which emphasizes the system's efficiency in handling data transmission, and the edge capability, which accounts for the server's capacity. The weighting parameter μ allows for the adjustment of the influence of each component. When μ is closer to 1, the reward is more influenced by the average throughput, emphasizing the importance of efficient data transmission. Conversely, when μ is closer to 0, the reward is more influenced by the edge capability, prioritizing the capacity of the edge server.

5.2.4 Discount factor

The discount factor, γ, is a real value $\in [0, 1]$ that determines the importance of future rewards in the learning process. A value close to 1 indicates a higher consideration for future rewards, while a value close to 0 focuses more on immediate rewards. Here we considered the value of γ to be 0.9, indicating that our work gives more importance to future rewards compared to immediate rewards.

5.3 Algorithms

Resource allocation in 6G networks presents a complex challenge that necessitates dynamic decision-making to optimize network performance while

considering diverse requirements. This process often involves sequential decisions where actions at one-time step can impact future decisions. Reinforcement Learning (RL) models have garnered recognition for their efficacy in autonomous decision-making within dynamic and uncertain environments, learning optimal behaviors through interactions guided by a reward signal. In this study, our focus is on comparing two advanced RL techniques, the Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG) algorithms, both known for their adaptability to continuous action spaces and high-dimensional state spaces.

DQN, an original model-based on-policy RL algorithm, operates in a discrete action space. On the other hand, DDPG extends this concept to the continuous action space, learning a deterministic policy. The general framework involves an agent within an environment with associated states. The agent interacts with the environment to take action, receiving rewards based on its actions. The goal is to maximize cumulative rewards through optimal actions. DDPG is an actor-critic algorithm comprising two entities: the actor selects actions for a given state, while the critic evaluates actions based on the state, generating Q-values to measure action effectiveness. DDPG utilizes four neural networks: a Q network, a deterministic policy network, and their target counterparts. This architecture, derived from the Advantage Actor-Critic (A2C) algorithm, stabilizes learning and emphasizes long-term rewards.

In this project, we propose a comparative study between DQN and DDPG-based Resource Allocation (DRA) Algorithms for smart hospital settings utilizing 6G networks.

Algorithm 1 DRA: DDPG based Resource Allocation

1. Initialization:
State $S = (CPU_{util}, P_{avail}, B_{avail}, D_{trans}, N_{users}, T_{off})$
Action $A = (A_b, A_p, O_t)$
Actor learning rate $LR_A = 0.0001$
Critic learning rate $LR_C = 0.0002$
Discount factor $\gamma = 0.9$
Target network update rate (soft update) $\tau = 0.01$
Memory capacity of Replay Buffer $B = 10000$
Batch size $N = 32$
Max no. of episode $M = 500$

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

2. Collect Experience:
Select action $a_t = (\mathrm{A}_b, A_p, O_t)$ from state s_t as per current policy
Execute action a_t, observe reward r_t and move to next state s_{t+1}
Store transition (s_t, a_t, r_t, s_{t+1}) in B

3. Update Critic:
Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from B
Set $y_i = r_i + Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
Update critic by maximizing the reward:

$$R = \mu \cdot T_{\text{avg}} + (1 - \mu) \cdot E_c \tag{5.4}$$

Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i} \tag{5.5}$$

4. Update Target Network:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1 - \tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1 - \tau)\theta^{\mu'}$$

end

The DRA algorithm leverages the strengths of both DQN and DDPG to learn and adapt optimal resource allocation strategies in dynamic hospital environments. As outlined in Algorithm 1, the DRA continuously refines its

resource allocation decisions based on feedback from the environment and reward signals, aiming to achieve efficient bandwidth and edge server services allocation for users.

The algorithm begins by defining the state S encompassing various factors like CPU utilization (CPU_{util}, available processing power P_{avail}, bandwidth availability B_{avail}, data transfer requirements D_{trans}, number of users N_{users}, and offloading threshold T_{off}. The action space A represents the allocation decisions, including bandwidth allocation A_b, processing power allocation A_p, and offloading decisions O_t. Additionally, hyper-parameters like learning rates for actor and critic networks, discount factor, target network update rate, the memory capacity of the replay buffer, batch size, and the maximum number of episodes are set. During each episode, the DRA interacts with the environment. It selects an action a_t based on the current state s_t using the actor network's policy. This action represents the allocation of bandwidth and processing power, along with potential offloading decisions to edge servers. The chosen action is then executed, and the observed reward r_t and the next state $s_(t+1)$ are recorded. This experience $s_t, a_t, r_t, s_(t+1)$ is stored in a replay buffer for future learning.

Periodically, the algorithm samples a minibatch of experiences from the replay buffer. The critic network is then updated by comparing the actual reward r_i with the expected future reward based on the target critic network's evaluation of the next state $s_(i+1)$. This process minimizes the temporal difference error and refines the critic's ability to assess the effectiveness of actions. Subsequently, the actor-network is updated using the sampled policy gradient. This encourages the actor to prioritize actions that lead to higher long-term rewards.

Algorithm 2 DRA: DQN based Resource Allocation

1. Initialization: State $S = (CPU_{util}, P_{avail}, B_{avail}, D_{trans}, N_{users}, T_{off})$
Action $A = (A_b, A_p, O_t)$
Learning rate $\alpha = 0.001$
Discount factor $\gamma = 0.9$
Exploration rate $\epsilon = 1.0$
Decay rate for exploration rate $\epsilon_{decay} = 0.995$
Memory capacity of Replay Buffer $B = 10000$
Batch size $N = 32$
Max no. of episode M= 500

for *episode* = 1 *to* M **do**

2. Collect Experience: Select action a_t =(A_b, A_p, O_t) from state s_t using ϵ-greedy policy
Execute action a_t, observe reward r_t and move to next state s_{t+1}
Store transition (s_t, a_t, r_t, s_{t+1}) in B
3. Update Q-Network: Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from B
Compute target Q-value:

$$r = \mu \cdot T_{\text{avg}} + (1 - \mu) \cdot E_c \tag{5.6}$$

$$Q_{target}(s, a) = r + \gamma \cdot \max_{a'} Q(s', a') \tag{5.7}$$

Update Q-network by minimizing the loss:

$$\text{Loss} = \frac{1}{N} \sum_{i} \left(Q(s_i, a_i) - Q_{target}(s_i, a_i)\right)^2 \tag{5.8}$$

4. Update Exploration Rate:

$$\epsilon \leftarrow \epsilon \times \epsilon_{decay};$$

end

To ensure the agent focuses on long-term rewards and avoids overfitting, the target networks (actor and critic) are slowly updated toward the main networks using a soft replacement parameter τ. This ensures the learning process prioritizes the target networks' evaluations, which are less influenced

by recent experiences. By iteratively executing these steps, the DRA progressively learns to allocate bandwidth, processing power, and offload tasks effectively, optimizing resource utilization and service delivery in the dynamic smart care home environment.

The DQN algorithm initializes the state S, action A, learning rate α, discount factor γ, exploration rate ϵ, memory capacity of the replay buffer B, batch size N, and the maximum number of episodes M. During each episode, the algorithm interacts with the environment by selecting actions based on an ϵ-greedy policy. This policy balances exploration and exploitation, allowing the algorithm to explore new actions while also exploiting learned knowledge.

Experiences, including states, actions, rewards, and next states, are stored in a replay buffer for learning. Periodically, the algorithm samples a mini-batch of experiences from the replay buffer to update the Q-network. The Q-network is updated by minimizing the loss between predicted Q-values and target Q-values computed using the Bellman equation. This update process improves the Q-network's ability to estimate the expected future rewards for different actions in various states.

Additionally, the exploration rate ϵ is updated over time to gradually reduce exploration and focus more on exploitation as the algorithm learns. This helps the algorithm converge towards optimal policies while ensuring sufficient exploration to discover new strategies.

In contrast to DDPG, which uses actor-critic architecture and continuous action spaces, DQN operates with a simpler Q-learning approach and discrete action spaces. The comparison between DQN and DDPG in resource allocation tasks would assess their performance in adapting to dynamic environments, handling complex action spaces, and achieving optimal resource utilization.

Chapter 6

Result and Analysis

In this section, we evaluate the performance of the proposed resource allocation algorithm using two reinforcement approaches: Deep Deterministic Policy Gradient (DDPG) and Deep Q-Network (DQN) based on cumulative reward acquired during the learning process, throughput, and edge capability. The experimental setup aimed to assess the impact of varying user and edge server configurations on the algorithm's learning process and the subsequent performance of the agent. Table 1 outlines the test cases considered in the experiment evaluation.

Table 6.1: Experimental design and evaluation criteria

Cases	**#Users**	**#Edge Servers**	**Evaluation Criteria**
Case 1	10	10	Balanced resource distribution
Case 2	15	10	Higher computing resource demands
Case 3	10	15	Demand exceeds resources availability

The experiment delved into assessing crucial aspects influencing the DRA

algorithm's performance, primarily focusing on tuning parameters in Table 2 and performance evaluation factors. Tuning parameters encompassed elements such as the number of steps taken by the agent within each episode, the actor and critic learning rates dictating the pace of parameter updates during training, and the overall duration measured in episodes for the learning process to unfold. This fine-tuning process is critical as it determines how quickly the model adapts to new information and refines its decision-making strategies. The discount factor parameter holds significance in reinforcement learning, as it defines the degree to which the algorithm considers future rewards in its decision-making process. By adjusting this factor, researchers can gauge the algorithm's focus on immediate rewards versus long-term cumulative rewards, thus balancing exploration and exploitation strategies.

Table 6.2: Tuning Parameters

Symbol	**Definition**	**Default Value**
L_{RA}	Actor learning rate	0.0001
LR_C	Critic learning rate	0.0002
γ	Discount factor	0.9
τ	Soft replacement	0.01
N	Batch size	32
E	Maximum no. of steps	1000
M	Maximum no. of episode	500

On the evaluation front, the metrics studied included reward analysis, illustrating how the reward curve evolved over episodes, and offering insights into the algorithm's learning dynamics. Throughput analysis scrutinized the trends in throughput during training, shedding light on resource utilization and efficiency. Additionally, the edge capability analysis delved into the model's adaptability concerning edge server capabilities, crucial in un-

derstanding the system's robustness and scalability. Complementing these metrics, statistical analysis involved computing and recording the mean, standard deviation, and range of rewards, providing a quantitative understanding of the algorithm's performance landscape.

6.0.1 Analysis 1: Evaluation of User and Edge Server Configurations on Model Learning

In this analysis of user and edge server configuration on model learning, we designed three experimental cases. In the first case, we explore a scenario with balanced resources, where the number of users and edge servers are equal. The second case investigates a situation with excess resources, where there are more edge servers than users. Finally, the third case examines a high-demand situation, where the number of users exceeds the available edge servers. By analyzing these different scenarios, we aim to gain a comprehensive understanding of how the system performs under various resource constraints.

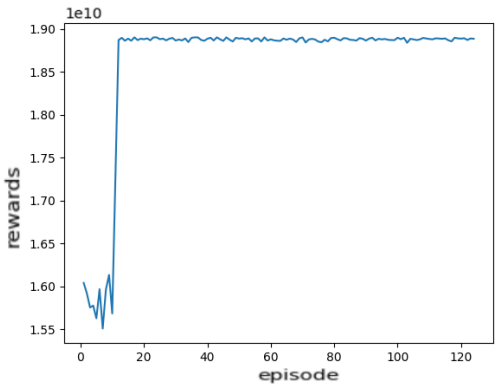
DDPG Algorithm Analysis

DDPG is an off-policy reinforcement learning algorithm that utilizes actor-critic architecture. Here, we evaluate how user-to-edge server ratios influence DDPG's learning performance across the three experimental cases:

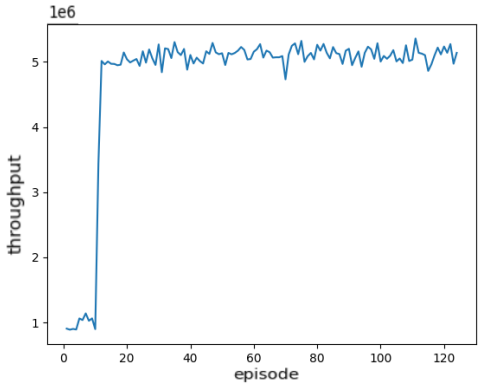
Case 1: Balanced Resource Distribution with Equal Number of Users and Edge Servers: In the scenario where the number of users equaled the number of edge servers, the system exhibited balanced resource distribution. The reward, throughput, and edge capability graphs displayed a compelling convergence to constant values over episodes, indicating not

only the algorithm's ability to learn but also a stable and efficient learning process. This equilibrium underscores the robustness of the model when faced with a scenario of optimal resource distribution. Fig. 6.1 presents a sample simulation result for a scenario with 10 users and 10 edge servers. Fig. 6.1(a) depicts the cumulative reward earned by the agent over training episodes. We observe that around episode 20, the reward curve converges, suggesting that the agent has learned an effective resource allocation policy. Fig. 6.1(b) and 6.1(c), respectively, illustrate the stability of throughput and edge capability throughout the training process. These results indicate that the system achieves stable performance in terms of both resource utilization and efficient utilization of edge server capabilities.

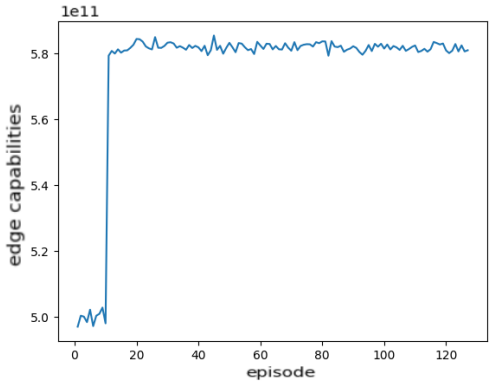
Case 2: Excess Resource Distribution with More Servers than Users: In the second case, we explore a scenario with more edge servers than users, representing an excess resource situation. Here, the system has more computational power available than what's currently demanded by the users. Fig. 6.2 showcases the performance of the DDPG algorithm with 10 users and 15 edge servers. Similar to Case 1, the cumulative reward graph in Fig. 6.2(a) exhibits convergence around episode 20 indicating that the agent effectively learns a resource allocation policy. Unlike Case 1, the edge capability graph in Fig. 6.2(c) converges at a significantly higher value. This is expected behavior as there are more servers available to handle tasks. With increased server capacity, the system can process more data and deliver improved service to users. Notably, the throughput graph in Fig. 6.2(b) maintains stability throughout the training process. This demonstrates that the system doesn't compromise on data transmission, even with excess resources. The system demonstrated improved performance, emphasizing its capability to utilize surplus resources effectively.



(a) Cumulative Reward vs Episode



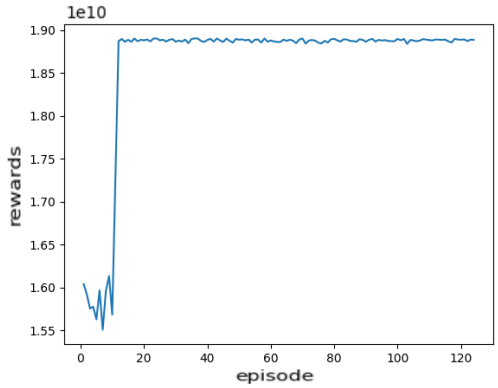
(b) Throughput vs Episode



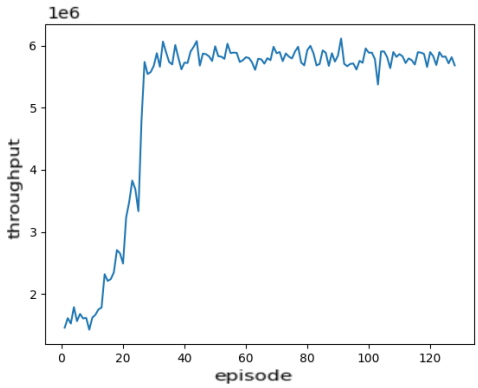
(c) Edge capabilities vs Episode

Figure 6.1: Performance analysis of user and edge server balanced configuration: (a) Cumulative reward converges around episode 20 (b) Throughput increasing as episode increases and gradually converges (c) Edge Capabilities increases as episode increases and converges at episode 20

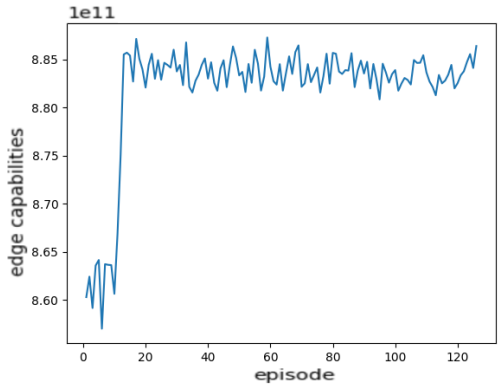
Case 3: Higher User Demand with More Users than Servers: The third case investigates a scenario where there are more users than edge servers, representing a situation with higher computational demand than available resources. Here, the system needs to efficiently allocate limited resources among a larger user base. Fig. 6.3 presents a sample simulation result for a scenario with 15 users and 10 edge servers. Compared to Case 1, the cumulative reward graph in Fig. 6.3(a) exhibits convergence at a slightly lower value. This suggests that while the agent adapts to the in-



(a) Cumulative Rewards vs Episode



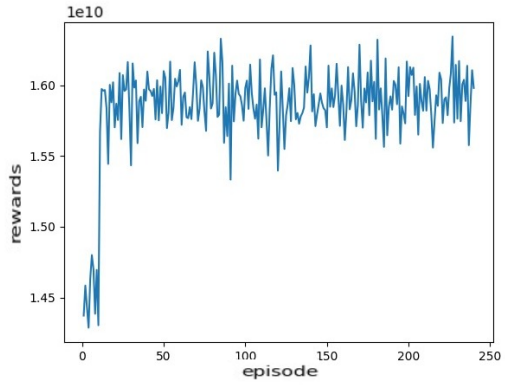
(b) Throughput vs Episode



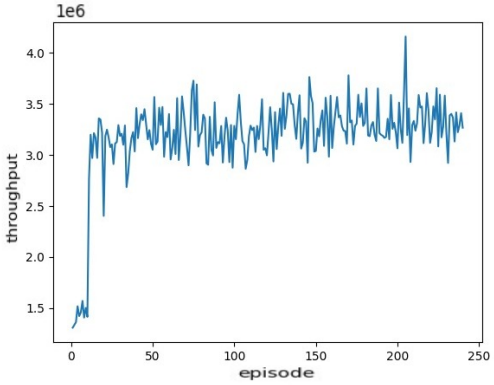
(c) Edge capabilities vs Episode

Figure 6.2: Performance analysis of user and edge server configuration under surplus resource conditions:(a) The cumulative reward converges approximately by episode 20. (b) Throughput shows a progressive increase with each episode and eventually stabilizes. (c) Edge capabilities show a steady rise with each episode until converging around episode 20.

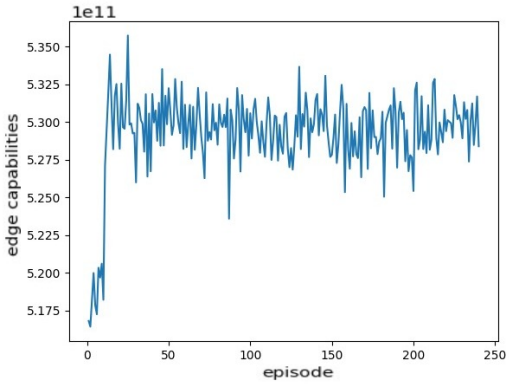
creased demand, the overall performance experienced a marginal reduction, indicating potential areas for optimization when confronted with increased computational needs. Interestingly, the edge capability in Fig. 6.3(c) reaches a stable value as servers are fully utilized. However, the maximum capability is lower than balanced case due to limited resources. Finally, Fig. 6.3(b) shows a slight decrease in throughput compared to balanced scenarios. This suggests the model prioritizes efficient resource allocation over maximizing throughput under higher demand.



(a) Cumulative reward vs Episode



(b) Throughput vs Episode



(c) Edge capabilities vs Episode

Figure 6.3: Performance analysis of user and edge server configuration under high user demand conditions (a) Cumulative reward reaches convergence around episode 25. (b) Throughput demonstrates an increase in growth with each episode, eventually converging. (c) Edge capabilities display continuous enhancement per episode until converging at episode 25.

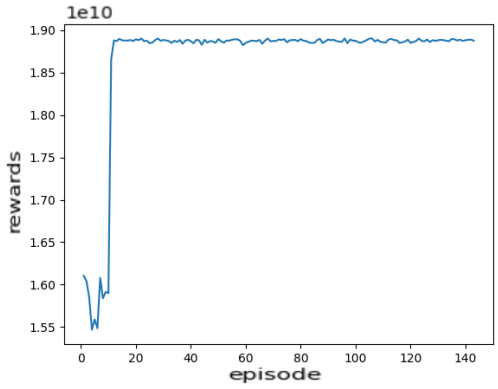
The observations from the above analysis showcase that the DDPG model demonstrates a strong ability to learn effective resource allocation policies, as evidenced by the convergence of cumulative reward graphs in all cases (Fig. 6.1(a), Fig. 6.2(a), and Fig. 6.3(a)). Additionally, Cases 1 (balanced) and 2 (excess resources) showcase stable performance in terms of throughput and edge capability utilization. The model adapts to varying resource constraints. Case 2 highlights the ability to leverage excess resources effectively (Fig. 6.2), while Case 3 demonstrates adaptation to situations with higher user

demand than available servers (Fig. 6.3). However, Case 3 also reveals a potential trade-off between handling increased demand and achieving optimal performance.

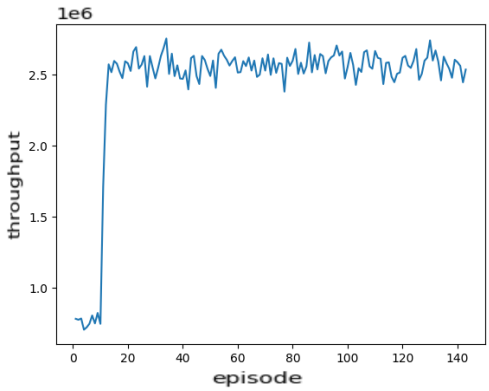
DQN Algorithm Results

Similarly, we analyze the performance of DQN, another popular reinforcement learning algorithm. We will evaluate DQN's learning efficiency under the same three experimental cases:

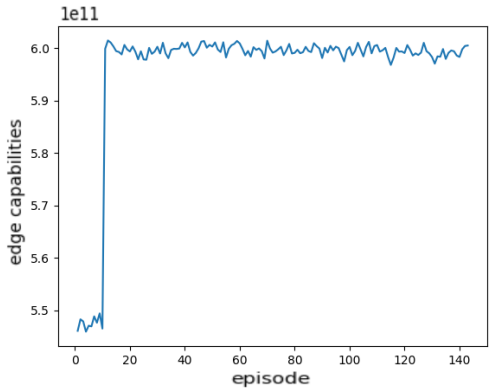
Case 1: Balanced Resource Distribution with Equal Number of Users and Edge Servers: DQN achieves balanced resource distribution in a scenario where the number of users equals the number of edge servers. This indicates the algorithm's ability to effectively learn and allocate tasks to edge servers, preventing overloading and maintaining their capacity. Fig. 6.4 presents a sample simulation result for a scenario with 10 users and 10 edge servers under the DQN algorithm. Fig. 6.4(a) illustrates DQN's reward graph. Here we observe a convergence around episode 10, suggesting that the agent has learned an effective resource allocation policy. This convergence signifies a stable and efficient learning process. Fig. 6.4(b), depicting DQN's throughput graph, shows a relatively stable throughput value of around 2.5 requests served per episode. This indicates that DQN is able to efficiently handle user requests while maintaining a healthy balance with resource utilization. Fig. 6.4(c) illustrates the stability of edge capability throughout the training process convergence around 6.0, suggesting that edge servers are operating at a moderate capacity, avoiding overload. This demonstrates DQN's ability to distribute tasks effectively and prevent resource exhaustion



(a) Cumulative Reward vs Episode



(b) Throughput vs Episode

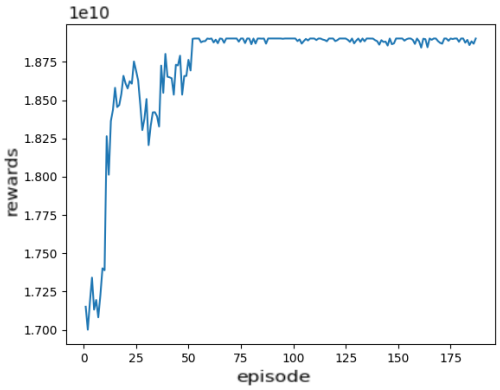


(c) Edge capabilities vs Episode

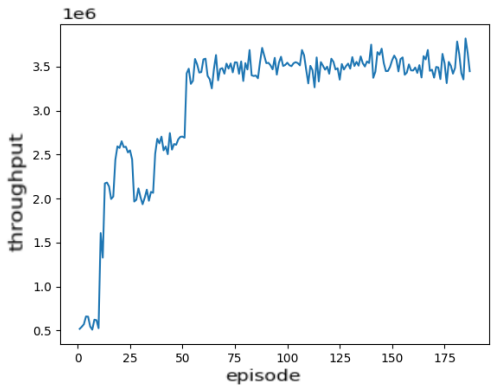
Figure 6.4: Performance analysis of user and edge server balanced configuration for DQN algorithm: (a) Cumulative reward converges around episode 10. (b) Throughput increases and stabilizes at the value of 2.5. (c) Edge capabilities show steady growth until convergence at episode 20.

Case 2: Excess Resource Distribution with More Servers than Users: Case 2 presents a scenario with more edge servers (15) than users (10), indicating an abundance of computational resources compared to the current user demand. Fig. 6.5 showcases the performance of the DQN algorithm in this case. The cumulative reward graph in Fig. 6.5(a) exhibits convergence around episode 60, suggesting that the agent has learned a resource allocation policy after 60 episodes that leverages the available resources. The throughput graph in Fig. 6.5(b) has a similar or slightly higher

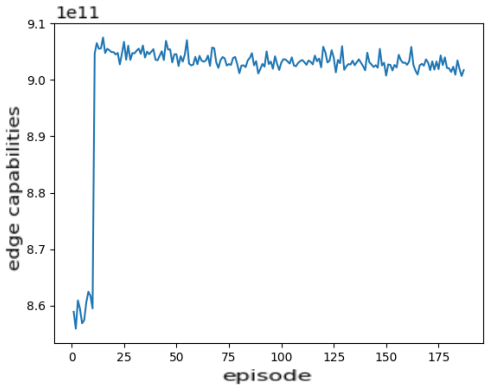
throughput compared to Case 1. This indicates that DQN prioritizes efficient resource utilization even with excess capacity. It does not significantly increase throughput beyond what's necessary to serve all users efficiently. The edge capability graph in Fig. 6.5(c) converges at a significantly higher value compared to case 1. This signifies that DQN avoids overloading servers, keeping them underutilized due to the abundance of resources.



(a) Cumulative Rewards vs Episode



(b) Throughput vs Episode



(c) Edge capabilities vs Episode

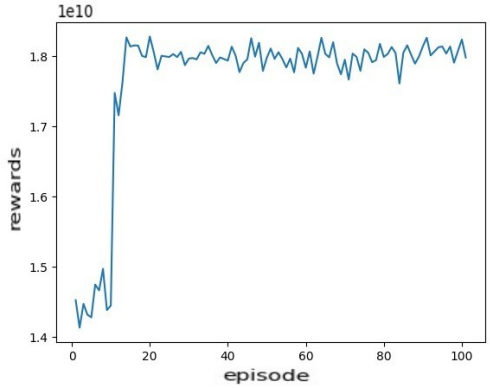
Figure 6.5: Performance analysis of user and edge server configuration under surplus resource conditions:(a) The reward graph converges around episode 60 (b) Throughput shows a progressive increase with each episode and eventually stabilizes at 3.5. (c) Edge capabilities show a steady rise with each episode until converging around episode 15.

Case 3: Higher User Demand with More Users than Servers: The third case investigates a scenario with higher user demand than avail-

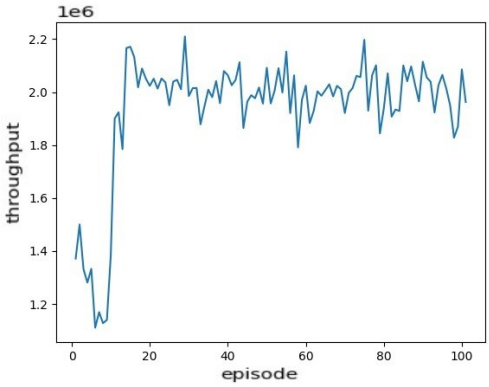
able resources. Here, the system needs to efficiently allocate limited resources among a larger user base (15 users and 10 edge servers). Fig. 6.6 presents a sample simulation result for this scenario. Compared to Case 1, the cumulative reward graph in Fig. 6.6(a) exhibits convergence around episode 20, but at a slightly lower value. This suggests that DQN adapts to the increased demand but experiences a minor performance reduction due to resource limitations. Even the DQN's throughput graph in Fig. 6.6(b) shows a slight decrease in throughput compared to the balanced scenario indicating that DQN prioritizes efficient resource allocation, potentially sacrificing some throughput to serve a larger user base with limited resources. The edge capability graph in Fig. 6.6(c) shows a stable value after convergence, potentially lower than the maximum capability observed in balanced cases. This signifies that servers are fully utilized due to high demand.

The analysis of DQN's performance across different user and edge server configurations reveals its capability to learn effective resource allocation strategies. Convergence of the cumulative reward graphs in all cases (Fig. 6.4(a), 6.5(a), and 6.6(a)) demonstrates this learning ability. DQN adapts its strategy based on resource availability. Cases 1 (balanced) and 2 (excess resources) showcase stable performance in terms of throughput and edge capability, with Case 2 highlighting efficient utilization of excess resources (Fig. 6.5(a)). In Case 3 (high user demand), DQN prioritizes efficient resource allocation, potentially leading to a slight decrease in throughput compared to balanced scenarios (Fig. 6.4(b)). This suggests a trade-off between handling increased demand and achieving peak performance under resource constraints.

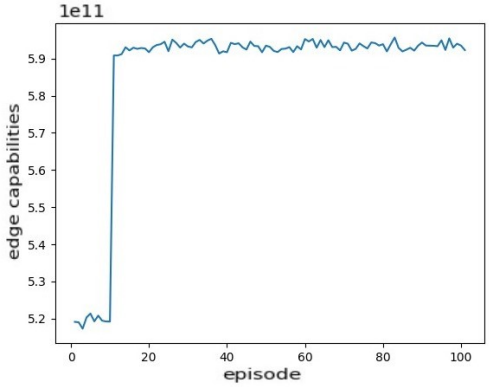
Both DDPG and DQN exhibited successful resource allocation in the



(a) Cumulative reward vs Episode



(b) Throughput vs Episode



(c) Edge capabilities vs Episode

Figure 6.6: Performance analysis of user and edge server configuration under high user demand conditions (a) The reward graph converges around episode 20 at a value near 1.8. (b) Throughput demonstrates an increase in growth with each episode, eventually converging near 2.0. (c) Edge capabilities display continuous enhancement per episode until converging at episode 10.

DRA algorithm across different user and edge server configurations. They achieved convergence in terms of cumulative reward, indicating their ability to learn effective allocation policies. DDPG demonstrates slightly faster convergence in specific scenarios and better resource utilization, while DQN prioritize user satisfaction in resource-constrained situations, potentially leading to higher rewards but pushing servers closer to their limits. Overall, both DDPG and DQN offer promising solutions for adaptive resource allocation in smart hospital environments.

6.0.2 Analysis 2: Evaluation of throughput and edge capability in dynamic environments considering varying user demands and resource capacity

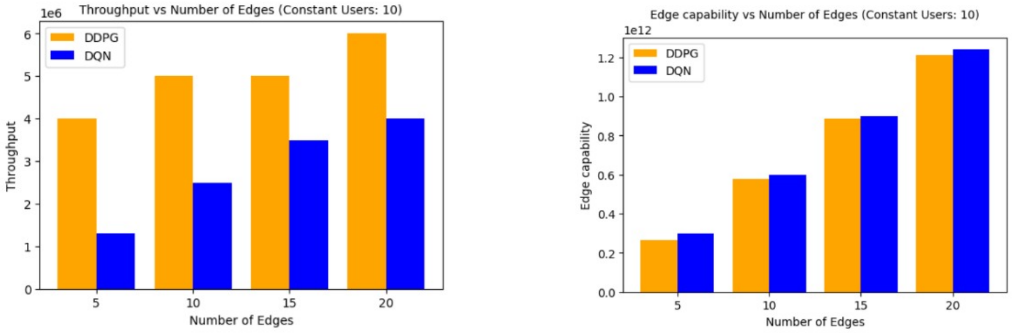
In Analysis 2, we explore the adaptability of the Resource Allocation model with both DQN and DDPG algorithms in dynamic environments. Edge computing systems often experience fluctuating user demands and resource availability due to factors like user behavior and varying workloads. To understand how the model handles these variations, we investigate two key scenarios, varying the number of edge servers with a constant number of users (Case 1) and vice versa (Case 2). In case 1, we fix the user base and observe the impact of adding more edge servers (edges) on throughput and edge capability. This helps us assess the model's ability to leverage increased resources to manage a consistent workload. Conversely, in case 2, we keep the number of edge servers constant and analyze how the model responds to a growing number of users. This sheds light on the model's capacity to handle rising user demands with limited resources. By analyzing these scenarios, we gain valuable insights into the scalability and efficiency of the model for resource allocation in edge computing environments that experience dynamic changes in both user demand and resource availability.

Case 1:Varying Number of Edges with Constant Users: This case explores the impact of varying edge server availability on the system's performance when the user base remains constant. We investigate how adding more edge servers (from 5 to 20) affects throughput and edge capability. The results for both DDPG and DQN reveal a clear trend: as the number of edges increases, both throughput and edge capability improves. Fig. 6.7 illustrates the analysis graphs of throughput and edge capability with vary-

ing resource capacity and constant user demand, comparing DDPG against DQN. As in Fig. 6.7(a), In DDPG, throughput exhibits a gradual increase, with a quantifiable jump observed between specific configurations. For example, increasing the number of edges from 5 to 10 leads to a 25% rise in throughput (from 4 Mbps to 5 Mbps). This trend continues, ultimately reaching a throughput of 6 Mbps with 20 available edges. Similarly, DQN shows an increase in throughput from 1.3 Mbps to 2.5 Mbps with an increase of approximately 92.3% ultimately reaching a throughput of 4 Mbps with 20 available edges. The rate of improvement differ based on its resource allocation strategies. With more edges available, DQN can distribute user requests more effectively, leading to better resource utilization and faster task processing. Additionally, the graph in Fig. 6.7(b) depicting edge capability against the number of edges showcases a consistent upward trend. As the number of edges increases from 5 to 20, edge capability of DDPG algorithm increases from 265 GB per edge to 1210 GB per edge and DQN algorithm increases from 300 GB to 1240 GB per edge. This further emphasizes the benefit of having more edge servers available, as it enhances the system's overall processing power and ability to handle user demands efficiently. The graph depicting edge capability for DQN showcases a similar trend to DDPG, signifying increased processing power and efficient utilization of servers as more resources become available.

In summary, both DDPG and DQN demonstrate effective resource utilization with increasing edge servers (constant user base). Throughput improvement suggests efficient workload distribution, highlighting the ability of both algorithms to leverage additional resources.

Case 2:Varying Number of Users with Constant Edges:



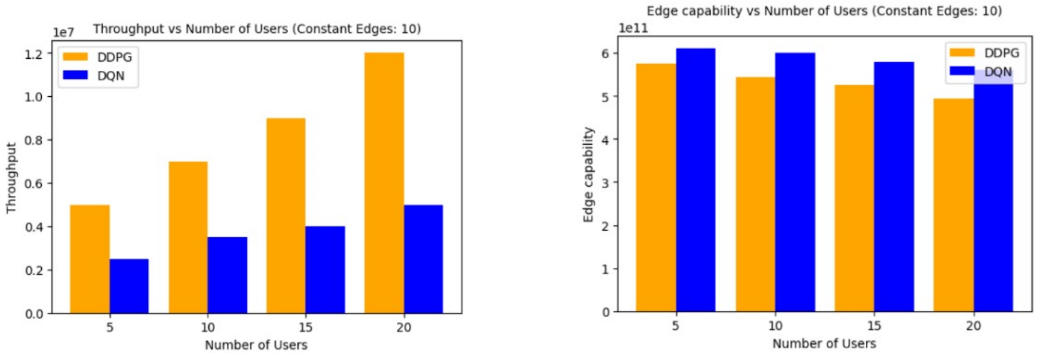
(a) Throughput vs No of edges

(b) Edge Capability vs No of edges

Figure 6.7: Analysis of throughput and edge capability with varying resource capacity and constant user demands: (a) Throughput increases with increased number of edge servers (b) Edge capability increases with increased number of edge servers

In Case 2, we investigated the impact of fluctuating user demands on the system's performance with a constant number of 10 edge servers. Here, the analysis focused on how the DRA model adapts to a growing user base (from 10 to 25 users) and its effect on both throughput and edge capability for both DDPG and DQN approaches. Fig. 6.8 depicts the analysis result of throughput and edge capability with varying user demands and fixed resource capacity. Here, we expect a decline in throughput as more users compete for limited processing power. Contrary to our initial expectation, In In Fig. 6.8(a), DDPG's throughput actually increases with a growing user base (from 10 to 25 users). However, the rate of increase is much smaller compared to Case 1 where the number of users remained constant. This suggests that DDPG can handle a larger volume of data (throughput) as the number of users increases, but at a diminishing rate as resources become strained. DQN's throughput also shows a slight increase with a growing user base. However, the increase is minimal compared to DDPG. This suggests that DQN prioritizes efficient allocation over faster processing, leading to a slower improvement in throughput even under moderate user demand. For

edge capability as in In Fig. 6.8(b), both DDPG and DQN show a different trend compared to Case 1. In this case, with a fixed number of edges (10), the edge capability starts at a higher value but shows a consistent decline as the number of users increases. This can be explained by the growing strain on individual servers as they handle requests from more users. While the overall throughput increases, this comes at the cost of increased workload on each server, leading to a decrease in individual edge capability.



(a) Throughput vs No of users

(b) Edge Capability vs No of users

Figure 6.8: Analysis of throughput and edge capability with varying user demands and fixed resource capacity: (a) Throughput gradually decreases with increased number of edge servers (b) Edge capability gradually decreases with increased number of edge servers

In conclusion, by analyzing the impact of varying user demands and resource availability on throughput and edge capability, this section revealed valuable insights into the strengths and weaknesses of DDPG and DQN algorithms within the Resource Allocation model. While both algorithms demonstrate effective resource utilization with additional edge servers (Case 1), their performance under increasing user demands (Case 2) exposes key differences. DDPG prioritizes throughput, handling a larger volume of data as user numbers increase, but at the cost of individual server capability. Conversely, DQN prioritizes efficient resource allocation, leading to slower throughput growth

but maintaining steadier edge capability. The choice between DDPG and DQN depends on the specific needs of the system, with DDPG better suited for maximizing throughput and DQN preferable for maintaining individual server performance under resource constraints.

Chapter 7

Conclusion

The devised Markov Decision Process (MDP) framework for the smart hospital environment, combined with the implementation of the Deep Deterministic Policy Gradient (DDPG) algorithm, has shown encouraging outcomes in optimizing resource allocation. Extensive evaluations conducted across diverse configurations of users and edge servers have showcased the model's capacity to converge effectively in the reward curve, indicating its effectiveness in learning and adapting to dynamic hospital scenarios. One notable aspect of the implemented DDPG algorithm is its dedication to fair resource allocation, addressing the critical need for equitable distribution of computing resources within the hospital environment. This ensures that all tasks, especially those of critical nature, receive appropriate priority, contributing to a more responsive and efficient healthcare system.

The primary goal of the smart hospital system was to enhance resource efficiency, ensuring critical tasks and services receive the necessary computing power for timely and efficient processing. The graphical representation of throughput and edge capability against episodes provides a clear visualiza-

tion of the positive impact on system performance, showcasing the model's ability to adapt and optimize resource allocation over time. Moreover, the comparison between the DQN and DDPG algorithms reveals insightful findings. While DQN excels in certain scenarios, particularly those with discrete action spaces, DDPG demonstrates superior performance in continuous action spaces and high-dimensional state spaces, making it more suitable for complex resource allocation tasks in dynamic environments.

However, it is crucial to acknowledge the inherent limitations, notably the paramount importance of data security measures in the smart hospital context. As we progress towards a more interconnected and data-driven healthcare landscape, future iterations of the system must prioritize robust security protocols to safeguard sensitive patient information and maintain trust in the system.

References

[1] Sami, Hani, Hadi Otrok, Jamal Bentahar, and Azzam Mourad. "AI-based resource provisioning of IoE services in 6G: A deep reinforcement learning approach." IEEE Transactions on Network and Service Management 18, no. 3 (2021): 3527-3540.

[2] Wei, Peng, Kun Guo, Ye Li, Jue Wang, Wei Feng, Shi Jin, Ning Ge, and Ying-Chang Liang. "Reinforcement learning-empowered mobile edge computing for 6G edge intelligence." IEEE Access 10 (2022): 65156-65192.

[3] Liu, Yan, Yansha Deng, Arumugam Nallanathan, and Jinhong Yuan. "Machine Learning for 6G Enhanced Ultra-Reliable and Low-Latency Services." IEEE Wireless Communications 30, no. 2 (2023): 48-54.

[4] Jiang, Wei, Bin Han, Mohammad Asif Habibi, and Hans Dieter Schotten. "The road towards 6G: A comprehensive survey." IEEE Open Journal of the Communications Society 2 (2021): 334-366.

[5] Hortelano, Diego, et al. "A comprehensive survey on reinforcement-learning-based computation offloading techniques in Edge Computing Systems." Journal of Network and Computer Applications 216 (2023): 103669.

[6] Chen, Miaojiang, et al. "Deep reinforcement learning for computation offloading in mobile edge computing environment." Computer Communications 175 (2021): 1-12.

[7] Zhang, Cheng, and Zixuan Zheng. "Task migration for mobile edge computing using deep reinforcement learning." Future Generation Computer Systems 96 (2019): 111-118.

[8] Lu, Shuaibing, et al. "A Dynamic Service Placement Based on Deep Reinforcement Learning in Mobile Edge Computing." Network 2.1 (2022): 106-122.

[9] Chen, Zhao, and Xiaodong Wang. "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach." EURASIP Journal on Wireless Communications and Networking 2020.1 (2020): 1-21.

[10] Salameh, Ahmed I., and Mohamed El Tarhuni. "From 5G to 6G—challenges, technologies, and applications." Future Internet 14.4 (2022): 117.

[11] Rui, LanLan, et al. "Service migration in multi-access edge computing: A joint state adaptation and reinforcement learning mechanism." Journal of Network and Computer Applications 183 (2021): 103058.

[12] Liu, Fangzheng, et al. "Joint Service Migration and Resource Allocation in Edge IoT System Based on Deep Reinforcement Learning." IEEE Internet of Things Journal (2023).

[13] Chen, Yan, et al. "Dynamic task allocation and service migration in edge-cloud iot system based on deep reinforcement learning." IEEE Internet of Things Journal 9.18 (2022): 16742-16757.

[14] Chen, Juan, et al. "A DRL agent for jointly optimizing computation offloading and resource allocation in MEC." IEEE Internet of Things Journal 8.24 (2021): 17508-17524.

[15] Upadhyay, Yash, et al. "A Dynamic Approach for Handling UAV in Crowded Environment Using Deep Q-Networks." 2023 4th IEEE Global Conference for Advancement in Technology (GCAT). IEEE, 2023.

[16] Doke, Ashwini R., and K. Sangeeta. "Deep reinforcement learning based load balancing policy for balancing network traffic in datacenter environment." 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT). IEEE, 2018.

[17] Kumar, R. Prasanna, et al. "Attention-Guided Residual Network for Skin Lesion Classification Using Deep Reinforcement Learning." 2023 International Conference on Integrated Intelligence and Communication Systems (ICIICS). IEEE, 2023.

[18] Deepalakshmi, R., and J. Amudha. "A Reinforcement Learning based Eye-Gaze Behavior Tracking." 2021 2nd Global Conference for Advancement in Technology (GCAT). IEEE, 2021.

[19] Sha, Akhbar, et al. "Deep Reinforcement Learning for Brain Aneurysm Segmentation in 3D TOF MRA Images: A Comparative Study using 3D U-Net, 3D ResNet, and LSTM Networks." 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS). IEEE, 2023.

[20] Kumaar, AA Nippun, and Sreeja Kochuvila. "Mobile Service Robot Path Planning using Deep Reinforcement Learning." IEEE Access (2023).

Appendix A

Source code

Github Link to our Project Source Code