A COMPUTATION OFFLOADING FOR IOT OF EDGE COMPUTING: A REINFORCEMENT LEARNING APPROACH BASED ON DDPG

2023
MASTER THESIS
COMPUTER ENGINEERING

Fatimah Najeh Abdullateef Al ZUABAIDI

Thesis Advisor
Assist. Prof. Dr. Nehad T. A. RAMAHA

A COMPUTATION OFFLOADING FOR IOT OF EDGE COMPUTING: A REINFORCEMENT LEARNING APPROACH BASED ON DDPG

Fatimah Najeh Abdullateef Al ZUABAIDI

Thesis Advisor
Assist. Prof. Dr. Nehad T. A. RAMAHA

T.C.
Karabuk University
Institute of Graduate Programs
Department of Computer Engineering
Prepared as
Master Thesis

KARABUK
October 2023

I certify that in my opinion the thesis submitted by Fatimah Najeh Abdullateef Al ZUABAIDI titled “A COMPUTATION OFFLOADING FOR IOT OF EDGE COMPUTING: A REINFORCEMENT LEARNING APPROACH BASED ON DDPG” is fully adequate in scope and quality as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Nehad T.A. RAMAHA	....
Thesis Advisor, Department of Computer Engineering	

This thesis is accepted by the examining committee with a unanimous vote in the Department of Computer Engineering as a Master of Science thesis. 19/10/2023

Examining Committee Members (Institutions)	Signature
Chairman : Assist. Prof. Dr. Nehad T. A. RAMAHA (KBU)	....
Member : Assist. Prof. Dr. İsa AVCI (KBU)	....
Member : Assist. Prof. Dr. Muhammet ÇAKMAK (SU)	....

The degree of Master of Science by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabuk University.

Assoc. Prof. Dr. Zeynep ÖZCAN	....
Director of the Institute of Graduate Programs	

“I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well.”

Fatimah Najeh Abdullateef Al ZUABAIDI

ABSTRACT

M. Sc. Thesis

A COMPUTATION OFFLOADING FOR IOT OF EDGE COMPUTING: A REINFORCEMENT LEARNING APPROACH BASED ON DDPG

Fatimah Najeh Abdullateef Al ZUABAIDI

Karabuk University
Institute of Graduate Programs
The Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Nehad T. A. RAMAHA
October 2023, 74 pages

The continuous proliferation and diversification of Internet of Things (IoT) devices have led to the emergence of computationally intensive and time-sensitive applications, including but not limited to object detection, smart homes, and smart grids. To address the computational limitations of IoT devices, the edge computing paradigm offers a solution by offloading resource-intensive tasks from IoT devices to more powerful edge nodes. Despite this, the edge computing architecture may introduce high latency, which proves unsuitable for IoT devices with constrained computing and storage capacities. Efforts have been made to enhance this scenario by deploying edge devices in proximity to IoT devices, providing low-latency computing resources. However, challenges persist, particularly when the edge server is inundated with offloading requests, potentially leading to incomplete task processing within the required timeframe.

This paper seeks to minimize the average task completion time in an IoT edge-computing environment by optimizing the task offloading ratio from IoT devices to the edge. This optimization is achieved through the utilization of Deep Deterministic Policy Gradient (DDPG), a form of Reinforcement Learning (RL) approach. Our approach involves implementing a dynamic task offloading decision mechanism on the edge, capable of determining the appropriate computational resources and resource allocation needed to complete a task. Additionally, we enhance the load-balancing process, ensuring fair distribution of resources among tasks, thereby reducing processing time and, consequently, response time. The results of our study illustrate that our dynamic task offloading decision mechanism significantly improves the overall completion time of tasks compared to conventional approaches.

Key Words : Edge computing, Task offloading, Reinforcement learning, Deep reinforcement learning, Offloading deviation, Load balancing.

Science Code : 92517

ÖZET

Yüksek Lisans Tezi

UÇ BİLGİ İŞLEM İÇİN BİR HESAPLAMA BOŞALTMA: DDPG'YE DAYALI BİR TAKVİYE ÖĞRENME YAKLAŞIMI

Fatimah Najeh Abdullateef Al ZUABAIDI

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr Üyesi. Nehad T. A. RAMAHA
Ekim 2023, 74 pages

Nesne algılama, akıllı evler ve akıllı şebekeler gibi hesaplama açısından yoğun ve gecikmeye duyarlı uygulamalar, Nesnelerin İnterneti (IoT) cihazlarının üstel büyümesi ve çeşitliliği ile sürekli olarak ortaya çıkmaktadır. Hesaplama ağırlıklı görevleri IoT cihazlarından uç düğümlere aktarmak için uç bilişim paradigmasını uyarlayabiliriz, bu da IoT cihazlarının sınırlamasını daha güçlü kaynaklarla aşabilir. Bununla birlikte, uç bilişim mimarisi, sınırlı bilgi işlem ve depolama yeteneklerine sahip IoT cihazları için uygun olmayan yüksek gecikmeye neden olabilir. Uç bilişim, IoT cihazlarına düşük gecikmeyle bilgi işlem kaynakları sağlayabilen IoT cihazlarının yakınında bir uç cihaz dağıtarak bu durumu iyileştirmek için sunulmuştur. Bununla birlikte, uç sunucu, talepler yoğun şekilde geldiğinde cihazlardan yüklenen tüm görevleri gereken sürede tamamlayamayabilir. Bu makalede, bir tür Takviyeli Öğrenme (RL) yaklaşımı olan Derin Belirleyici Politika Gradyanı'na (DDPG) dayalı olarak IOT cihazlarından uca görev boşaltma oranını optimize ederek bir IoT uç bilişim ortamında görevlerin ortalama tamamlanma süresini en aza indirmeyi amaçlıyoruz. Bir görevi tamamlamak için birden fazla faktörü göz önünde bulundurarak işlenecek hesaplama kaynaklarının miktarını ve kaynak tahsisini belirleyebilen, uçta konuşlandırılmış dinamik bir görev boşaltma karar mekanizması öneriyoruz. Ayrıca, bu çalışmada, yük dengeleme sürecini iyileştiriyor ve kaynakları görevlere adil bir şekilde dağıtıyoruz; bu da işlem süresini ve dolayısıyla yanıt süresini azaltacaktır. Sonuçlar, dinamik görev boşaltma karar mekanizmamızın görevlerin genel tamamlanma süresini naif yaklaşımlara göre iyileştirebileceğini göstermektedir.

Anahtar Kelimeler : Kenar hesaplama, Görev devretme, Pekiştirmeli öğrenme, Derin pekiştirmeli öğrenme, Devretme sapması, Yük dengeleme.

Bilim Kodu :92517

ACKNOWLEDGMENT

Thanks, Allah, the Most Merciful, has granted me the opportunity to pursue a master's degree and conclude my thesis period. I am also grateful to my advisor, Assist. Prof. Dr. Nehad T.A. RAMAHA, for his assistance and direction in completing this thesis. I also sincerely thank the Karabuk University doctors, members, and colleagues; they played a crucial role in my academic journey. Finally, I am also grateful for the support of my family.

I respectfully present my thesis to Iraq, a country I adore, and Turkey, which has generously provided a home for our scholarly endeavours.

CONTENTS

Page

Page

LIST OF FIGURES

Page

LIST OF TABLES

<u>Page</u>

ABBREVIATIONS

EC	: Edge computing
DRL	: Deep Reinforcement Learning
NN	: Neural Network
IoT	: İnternet of Things
DDPG	: Deep Deterministic Policy Gradient
MDP	: Markov Decision Process
DQN	: Deep Q-Network
MEC	: Mobile Edge Computing
QL	: Q-learning

PART 1

RESEARCH OVERVIEW

1.1. INTRODUCTION

It's no secret that the Internet of Things (IoT) is on the rise, resulting in the generation of vast amounts of data for various purposes. The inception of IoT dates back to 1999 [2]. With the increasing demand for lower response times and better reliability, more services are being shifted from cloud computing to edge. Data is processed at the edge of networks with edge computing, thereby reducing response time and improving reliability[3]. In essence, IoT refers to a network of billions of devices that can detect, transmit, and transform data into a centralized system. Today, IoT devices and applications are deployed in various fields such as healthcare, smart city networks, intelligent transportation, and disaster management providing diverse forms of user assistance [4].

A vast number of distributed edge nodes with underutilized resources can be leveraged to reduce latency and bandwidth in IoT networks. Consequently, the application of IoT devices has gained considerable attention in recent years [5]. The purpose of IoT is to leverage edge processing to minimize latency. While IoT devices can collect large amounts of data, their capacities are limited. Cloud computing is commonly used for data analysis, but it results in longer delays than edge computing (EC). Edge processing, on the other hand, is an innovation that offers several advantages and can solve a wide range of issues[6].

The edge infrastructure of an organization offers a highly dynamic environment to numerous devices providing enhanced flexibility to users and supporting heterogeneous applications. Applications with sporadic traffic patterns necessitate significant battery life, memory capacity, computational resources, and prompt task execution [7]. Recently, edge computing has gained traction to achieve low-latency data processing by offloading computational resources from cloud data centers to edge devices. Additionally, the availability of reliable high-speed internet and communication technologies has fostered the proliferation of complex and computationally intensive (IoT) applications resulting in generating and processing massive amounts of data [8]. Typically, edge devices are more powerful than conventional IoT devices. This is because IoT devices are constrained by their limited resources making local processing and storage of large volumes of multimodal sensory data infeasible.

The proximity of (EC) services to (IoT) devices and data sources is crucial to reduce data transfer delay and minimize bandwidth consumption [9]. The proliferation of data-intensive applications in the (IoT) edge exacerbates the challenge of limited data speed [10]. The (IoT) can sense, actuate, and transmit information to a central system [11]. Currently, (IoT) devices and applications are utilized in a range of fields including logistics, retail, healthcare, smart city networks, intelligent transportation, and disaster management. The potential of (IoT) devices must be harnessed to provide sophisticated services [12]. Meanwhile, The number of (IoT) systems is growing exponentially [13].

They have already surpassed the world's population and are projected to reach 80 billion by 2030 [14]. This enormous growth necessitates platforms that can support the burgeoning number of IoT devices, organize and process the generated data effectively [15].

Considering devices that belong to the IoT, it is essential to handle data with short delays to guarantee fast user feedback. However, due to the limited power and computational capabilities of these devices, processing data at the edge is necessary. These requirements put constraints on the latency of edge processing [16].

To process this challenge, (EC) has emerged as a critical enabler for IoT. It brings cloud resources closer to the edge, supports real-time applications, and reduces the computing and network resources required for data processing in the cloud. By avoiding the delay of transferring data to distant clouds, EC can offer shorter latency to applications that require real-time processing. [17]. The process of edge registering is inherent as it involves a distinct convergence of EC target planning, wherein a multitude of cooperative edge devices and IoT devices collaborate to manage the information generated within the vicinity of the edge. The objective of this research is to utilize EC for processing IoT devices on the Internet or on a device situated at the edge of the network, which is closer to the data source.

Compared to cloud computing, EC has greater potential to accommodate more devices and facilitate IoT scenarios. Given these advantages, EC is considered a pivotal enabler for achieving ubiquity.

According to recent research, EC has demonstrated greater potential than cloud computing in accommodating an increasing number of devices and enabling the implementation of IoT scenarios. Consequently, EC represents a pivotal catalyst for the widespread adoption of ubiquitous computing.

1.2. PROBLEM STATEMENT

IoT refers to a network that connects devices and systems allowing users to access them from anywhere and, at any time [18]. IoT has been applied in fields such as home networks, home automation, manufacturing, security systems, healthcare, data management, analysis of military transportation systems, and sensor technology [19]. A frequent challenge in this context often involves data loading and unresponsive applications, which can negatively impact the user experience.

However, one significant challenge that arises with these applications is the delay, in transferring data, between devices and distant servers. This problem often leads to slow data loading and unresponsive application performance negatively impacting the user experience. To overcome these obstacles faced by IoT devices, the implementation of edge computing has emerged as a solution.

This computing pattern enhances the capabilities of user devices when it comes to running applications. Additionally, it provides security and faster performance compared to cloud computing [20, 21]. Consequently, processing happens with delay and lower power consumption leading to network performance [22]. This approach allows IoT devices to offload computing tasks to edge nodes that have stronger computing and storage skills. The proximity of an edge node to devices significantly reduces latency compared to cloud centric, where its processing nodes are central and far away from the IoT devices [18]. However, when edge nodes are faced with a volume of requests, an edge node might be challenged to complete tasks from IoT devices within the expected timeframe. Hence, it becomes crucial to deploy a cluster of edge nodes near users with priority given to selecting the node that's closest to achieve faster tasks first.

When it comes to edge computing, achieving network performance requires a dynamic smart, and fast resource allocation strategy. In edge networks using reinforcement learning techniques can greatly improve resource allocation. It goes beyond analyzing network data and also optimizes network operations while providing instructions to devices. This helps in making decisions about resource allocation precise and faster. Moreover, training reinforcement learning can significantly enhance the quality of services [20].

When relying solely on edge computing to perform tasks during peak loads on the edge server, the limited resources of the edge server can cause delays in completing tasks for devices. This is mainly because there are requests for offloading making it difficult to achieve tasks in efficient time. To tackle this challenge, the edge centric can intelligently allocate a portion of tasks to be managed directly on the edge centric itself which helps speed up the offloading procedure. As a result, we suggest a mechanism for making decisions about task offloading that optimizes the ratio of tasks being offloaded from the edge. This mechanism uses a customized approach based on Deep

Deterministic Policy Gradient (DDPG) that is tailored for different scenarios rather than adopting a one-size- strategy that fits all offloading tasks.

Currently, two primary resource allocation approaches in edge computing don't rely on a central cloud center [23]. One perspective revolves around users aiming to achieve optimal resource allocation by selecting suitable edge nodes for computational migration. This method assumes that all users are located within an area that encompasses edge computing networks and has access to nearby edge nodes. Users utilize game theory and operational research to determine the node and develop resource scheduling strategies [24]. The other perspective focuses on edge nodes highlighting the importance of organizing clusters of these nodes to optimize system performance. This approach suggests that collaboration among edge nodes is the key to master allocating resources efficiently [25] [26].

Dealing with control tasks that involve a range of possible actions poses a significant challenge when it comes to learning the most effective approach. This is true in (RL) where you have a large number of actions that can hinder performance. Known algorithms like DQN and Q learning among others have difficulty handling extensive action spaces. Unfortunately, encountering these action spaces is quite common. Moreover, DQN can face issues like training variability, and instability. Moreover, it struggles with handling continuous action spaces. In contrast, the DDPG algorithm proves to be efficient in addressing control tasks that involve continuous action spaces.

1.3. SIGNIFICANCE

Within the IoT environment, latency reduction is fundamental. In cloud-centric Internet of Things framework, data is handled on central servers which results in significant delay and latency within the IoT applications. Edge addresses this issue in the below style. Edge computing assigns computation, storage, and communication services from the cloud server to the network edge triggering increased availability and low latency [27]. IoT gadgets are resource-constrained and incapable of handling compute-intensive services. Incorporating edge computing, IoT and computation offloading offers an attainable solution concerning performance [27]. This combination guarantees faster response time, making real-time control and precise monitoring. Businesses that rely on instant decision-making, such as telemedicine and autonomous vehicles stand to benefit immensely from this progression.

Besides the above, data security is a vital concern to prioritize within the IoT environment. Centralized cloud centers have been the target of high-profile cyberattacks. Moreover, the presence of all the data in central servers increases data insecurity and breaches [28]. On the other hand, Edge computing progresses data security by decreasing the amount of data transmitted and handled in the cloud keeping sensitive data on client devices, and decreasing the risk of data compromise as posited by research[29]. This approach provides data processing, exchange in real time and minimizes the chance of data leaking amid the transit phase of information exchange between devices.

In addition to the above, efficiency is vital for the innovative progression of the Internet of Things and subsequently, the requirement for resource optimization. A study conducted by [30] on edge-centric IoT design highlights the centrality of resource allocation models especially, those utilizing reinforcement learning. These models aim to enhance the utility of edge nodes and users and so, optimizing the utilization of network resources and eventually progressing system performance. Such optimization becomes pivotal for guaranteeing the seamless operation of IoT frameworks mainly when resource limitations are a concern. The research underscores the significance of proficient resource allocation in edge-centric IoT systems associated with the prior study by [30], which divides the complex issue into reasonable components to enhance IoT network survivability.

The IoT ecosystem is extending phenomenally, as documented by a report from [31]. With an ever-growing number of connected devices, versatility is necessary for any IoT design making scalability a vital process. Research conducted by [27] asserts the characteristic scalability of our proposed edge-centric approach. This approach has been designed to consistently adjust the increasing number of IoT devices making it a feasible solution that addresses the network and advanced demands. This versatility is achieved through the use of the DDPG algorithm which is a computational resource offloading algorithm, and efficient resource allocation. All of these collectively contribute to the framework's ability to accommodate the growing IoT ecosystem, guarantee an ideal performance that can consistently accommodate the surge in IoT gadgets, and offer a feasible mitigation for the ceaselessly advancing digital scene.

1.4. RESEARCH OBJECETIVE

The goal of this research is to make edge-IoT user environments more productive in the IoT through optimal resource allocation. This can be accomplished by strategically delegating a portion of the assigned tasks to edge servers benefiting the end devices. To achieve this objective, our research focuses on enhancing the efficiency of task offloading from IoT devices to the edge.

Several vital objectives drive this research project:

1.4.1. To Optimize User Utility

Our essential objective is to plan a resource allotment demonstration that places client involvement at the forefront. We aim to maximize the utility and fulfillment of end-users inside the IoT environment. This involves advancing an advanced resource allocation component that minimizes latency and guarantees that clients can get the resources they require effectively without delay. We proposed to improve the general IoT encounter by centering on user-centric optimization.

1.4.2. To Minimize Latency

Decreasing latency could be a pivotal angle of our extension. We focus on a significant decrease in information transmission and handling delays inside IoT applications. To attain this, we arrange to handle information as near as possible to its source minimizing the time it takes for data to navigate the organization. Typically, it is especially basic for applications requesting real-time information and for those of a critical nature mission. Our endeavors will include accomplishing low-latency communication to meet these requested necessities successfully.

1.4.3. To Enhance Data Security

Security is fundamental within the IoT ecosystem, particularly when managing sensitive information. Our objective is to improve information security by receiving edge computing standards. This approach decreases the introduction of delicate information through its travel to centralized node centers. By actualizing security measures at the edge, we aim to invigorate data security, guaranteeing its secrecy and judgment through its travel inside the IoT framework.

1.4.4. To Utilize Reinforcement Learning

we aim to optimize resource allocation powerfully guaranteeing that resources are designated effectively and viably. IoT environment and fine-tuned calculations are adjusted and advanced intelligently to the changing network conditions using RL by utilizing the DDPG algorithm as an energetic resource allocation methodology.

1.5. CONTRIBUTION

The anticipated contributions of this research encompass a comprehensive transformation in the IoT ecosystem. It envisions a future where IoT devices predominantly rely on edge computing and challenge the conventional cloud-centric architecture resulting in substantial performance improvements, reduced latency, and enhanced data security.

A key element of this research is the introduction of a resource allocation model for latency reduction, which will strategically distribute processing power, bandwidth, and other critical network resources among edge nodes and users [27]. This model prioritizes real-time data processing and low-latency communication making it particularly valuable for applications requiring instantaneous responses such as industrial automation and autonomous vehicles.

Additionally, this research plans to leverage the DDPG algorithm which is a form of reinforcement learning that addresses IoT resource allocation challenges [27]. This innovative approach will enable dynamic and adaptive resource allocation. Furthermore, it enhances IoT technology towards intelligent and self-optimizing networks.

To validate the proposed edge-centric IoT architecture and resource allocation model, the research will conduct extensive empirical experiments with varying configurations of nodes and users [32]. This empirical validation aims to provide actionable insights into feasibility, scalability, and practical applicability.

Another significant aspect is the emphasis on enhancing Data Security and Privacy within IoT ecosystems [32]. The proposed edge-centric approach minimizes data exposure during transit to centralized cloud centers, mitigates potential vulnerabilities associated with data breaches or cyberattacks, and addresses growing data privacy, concerns or regulatory requirements.

Furthermore, the research promotes Environmental Sustainability within the IoT landscape. By favoring energy-efficient edge computing over traditional cloud-centric approaches, it aligns with global efforts to reduce technology's carbon footprint making IoT more eco-friendly [32]. Adopting an edge-centric IoT architecture will diversify IoT Applications significantly enabling real-time applications across various domains including autonomous vehicles, healthcare, augmented reality, and industrial automation. This diversification creates new opportunities for innovation and economic growth.

Also, the research aims to set the stage for future directions and innovation within the IoT and edge computing domains. It will highlight potential research avenues such as optimizing edge node deployment strategies, enhancing security measures, and integrating emerging technologies like 5G and edge AI ensuring continuous progressive innovation in the field.

In conclusion, this research seeks to revolutionize the IoT landscape by introducing an edge-centric paradigm shift, optimizing resource allocation, leveraging reinforcement learning, conducting empirical validation, enhancing data security, promoting sustainability, diversifying applications, and paving the way for future innovations. These contributions collectively address the challenges within the IoT ecosystem while fostering a more efficient, secure, and sustainable future for IoT technology or applications.

1.6. RESEARCH SCOPE AND LIMITATIONS

In this section, the research investigates the transformative potential of edge-centric IoT engineering. Examining IoT gadget composition, asserting assignment models, fortification learning with DDPG, and adaptability contemplations survey idleness decrease and security suggestions. These all compare to edge-centric approaches making the way clearer for a maintainable IoT future as detailed below.

1.6.1. IoT Device Composition

Our study builds upon the assumption that all IoT devices include two essential components—edge nodes and users. These components are essential in forming the IoT landscape. Edge nodes, either physical gadgets or virtual substances, are deliberately conveyed inside the network [13]. Physical nodes act as basic data handling centers, whereas virtual nodes offer adaptability by powerfully optimizing resource allocation. This compositional understanding supports investigating resource allocation, latency reduction, security, and versatility in edge-centric IoT design.

1.6.2. Resource Allocation Model

This model is fastidiously created to prioritize client utility augmentation while minimizing latency considered as an essential concern in IoT engineering. The resource assignment model is integral and is the allotment of basic resources for the consistent operation of the IoT environment. It comprehensively addresses handling control, transmission capacity, and other arranged resource dissemination.

The allocation of processing control is central to optimizing information handling and decision-making at the edge hubs guaranteeing proficient real-time responses to client demands. Compelling transmission capacity allocation empowers quick and dependable information exchange between clients and edge hubs improving the overall client encounter [33]. Moreover, the model considers distributing other network resources such as memory and capacity to guarantee the all-encompassing optimization of the IoT environment. By adjusting these key resource allocations, the research points to attaining an ideal trade-off between client fulfillment and network efficiency which consists of 10 edge nodes and several users taking into account the priority of choosing the appropriate node that is close to the users which leads to the effectiveness and efficiency of the network.

1.6.3. Reinforcement Learning with DDPG

Central to our approach is the application of support learning and particularly saddling on the capabilities of the profound DDPG algorithm. This advanced learning strategy leads our research and empowers us to analyze its transformative potential in IoT environments methodically. The DDPG calculation enables the resource allocation framework with versatility and intelligence. Through a process of learning from intuition and input, it permits the model to dynamically refine its resource allocation strategies based on real-time conditions and client needs [34]. This versatile approach improves the effectiveness and responsiveness of the IoT environment. Eventually, this contributes to the optimization of asset utilization, latency decline, and generally framework performance.

1.6.4. Experimental Configurations

Inside our research, a significant aspect spins around conducting a series of tests. Each test is designed to shed light on the capabilities and performance of our proposed edge-centric IoT design plus resource allocation show. These tests envelop three particular setups:

10 Nodes and 10 Users: In this introductory situation, we make a generally small-scale IoT environment to set up a pattern for our system's execution. We pick up bits of knowledge into their essential productivity and idleness decrease capabilities by

analyzing how the design and resource allotment handle this humble number of nodes and clients.

10 Nodes with 20 Users: Building upon the primary arrangement, we present a broader client base while keeping up the same number of hubs. This setup permits us to investigate how the framework adapts to expanded client requests and gives profitable experiences to its versatility plus resource allocation adaptability.

10 Nodes with 30 Users: Within the third setup, we assisted in heightening the client number to 30 while keeping the hub check steady. This setup reveals a more complex and requesting environment with challenging the system's capacity to preserve compelling resource assignments. This setup results in inactivity diminishment within the confront of increased client action.

PART 2

LITERUTURE REVIEW

Reinforcement learning has recently become a popular research area in edge computing for offloading reflecting the growing interest in edge computing. This distributed computing paradigm enables computation to be performed closer to the data source or end-users thus, reducing network latency and enhancing application performance. One technique employed in this context is offloading which allows tasks or components of tasks to be executed remotely, in either the cloud or edge, to improve overall system performance. The present chapter seeks to reinforce the problem statement and research objectives of this project by offering a comprehensive review of pertinent literature on three key topics: the Internet of Things (IoT), reinforcement learning (RL) and task offloading. Commencing with a concise overview of IoT, this section lays the groundwork for the ensuing discussion.

2.1. OVERVIEW OF INTERNET OF THINGS (IOT)

IoT, also known as a "global internet," constitutes a network of countless device connections and marks a groundbreaking stride in the domain of internet technology. Essentially, it involves an expansion of the internet beyond the digital realm integrating the physical world with unprecedented efficiency and ease. In practical terms, this means that everyday objects can engage in complex data exchange and communication. Thus, facilitating the creation of a truly interconnected world. [35]. Using IoT, connected physical objects are capable of collecting and exchanging information with other devices without requiring human intervention. Furthermore, these devices can be remotely controlled and leverage their internet connectivity. The diversity of IoT devices is extensive encompassing various applications such as smart electricity meters, televisions, appliances, homes, cities, healthcare, transportation, agriculture, security systems, and more, all of which hold significant potential [36].

Data transfer has been essential due to the increase in IoT devices, highlighting the importance of secure and efficient communication channels.

The advent of IoT technologies has propelled the widespread adoption of smart devices leading to a surge in computation-intense and latency-sensitive requests that require real-time reply capabilities [37]. The volume of data generated by IoT devices has increased significantly leading to challenges in processing this data due to the limited battery life of these devices, despite the availability of state-of-the-art computational and storage facilities. Consequently, efficient processing of the high demands of such applications has become a pressing concern for researchers and practitioners alike [38-40].

Furthermore, the scope of IoT devices extends beyond traditional electronic devices like sensors, smartphones, and other smart devices to encompass living beings such as animals, food, and plants. [41]. In their work, emphasize the importance of the needs of different objects within the IoT ecosystem including living beings like animals, food, and plants providing specialized features such as wireless, wired technology, or computational resources. There are several devices with varying capabilities and needs within the IoT ecosystem. Moreover, the priority of a task can differ across devices even within the same IoT application. This presents a significant hurdle in meeting the Quality of Service (QoS) benchmarks of the application.

2.2. EDGE COMPUTING

The ever-increasing demand for time-sensitive applications and the rapidly growing data volume have made the proximity of cloud resources to end devices a critical factor. To address this challenge, edge computing has emerged as a feasible solution to augment user experience by providing cloud centric resources nearer to the network's edge and the end devices. The fundamental process of offloading gadget tasks to a close edge has the potential to reduce the need for transmission to and from the cloud thus, mitigating bandwidth consumption in backhaul and cloud networks [42]. Edge processing offers multiple advantages over traditional cloud computing paradigms including lesser latency, enhanced energy effectiveness, and flexible computing for computation-intense requirements [43].

The advent of edge computing has proven to be a pivotal technology in augmenting the effectiveness of resource-intensive operations. This technology encompasses a multitude of methods including Fog, Mobile-Edge, and Cloudlet Computing. All of these function as intermediaries between endpoint devices and cloud servers, thereby furnishing computational and storage resources with negligible latency [44]. Nonetheless, devising effective task-offloading strategies in edge environments with limited resources remains a significant challenge that current research seeks to overcome [45].

Recently, there has been a growing interest in leveraging edge offloading for applications requiring low latency. This paper proposed a three-layer offloading architecture to optimize processing delays by distributing tasks across various layers based on communication and computing costs [46]. In a comparable vein, the task offloading quandary was tackled by a stochastic optimization lens, whereby they adeptly employed techniques of stochastic optimization to convert an unpredictable issue into a deterministic optimization issue. This approach reduces of energy consumption in transmission and guarantees low queueing latency [47].

To facilitate task offloading and resource allocation, we developed a blockchain-based framework known as Edge ABS, which utilizes smart contracts for task offloading and resource allocation. The Edge User Allocation (EUA) problem was also explored by[48], which aims to minimize latency and energy consumption in edge computing environments.

In recent years, the issue of time-sensitive applications exceeds the processing capabilities of IoT devices and poses a significant challenge. In response, researchers have explored the use of edge computing as a potential solution[49]. Researchers proposed an edge-computing architecture that aims to alleviate the congestion and delay experience by traditional multimedia IoT systems which are often limited by bandwidth constraints. To achieve this goal, their framework employs group formation and video group matching techniques to enhance the accurateness of human detection within a specified timeframe.

Likewise, Researchers investigated a mobile edge computing (MEC) framework tailored for unmanned aerial vehicles (UAVs) to facilitate the execution of computing tasks that are time-critical for terminal IoT devices within a restricted period [50]. Additionally, a UAV-assisted MEC system leverages differential evolution (DE) to determine near optimal locations of UAVs for task offloading in IoT devices[51]. Furthermore, edge computing has the potential to revolutionize industrial IoT (IIoT) applications, particularly in intelligent manufacturing[52]. Researchers conducted a thorough analysis and implementation of an edge computing framework for smart factories highlighting the assistances of fast processing, network agility, and autonomy [53]. The deployment of edge computing in IIoT can significantly improve operational efficiency and decision-making processes, thus increasing productivity.

To enhance the efficacy of edge computing, effective task scheduling mechanisms are imperative. however, conventional offloading techniques at the edge may not be adequate to manage a diverse range of task requirements, as highlighted by [54]. In response to this challenge, [55] proposed a joint decision-making approach that takes into account uncertain computing resources and heterogeneous task requirements. Similarly, [56] introduced a semi-distributed offloading technique featured with a multi-user scheduling mechanism to minimize average delay and power consumption for Narrowband Internet of Things (NBIoT) devices. This method utilizes an auction-based scheduling approach where the end devices submit bids to mobile base-stations which determine the scheduling. [57] proposed an energy aware optimization framework that prioritizes each demanded task and formulates a stochastic-aware offloading issue using the Lyapunov optimization method. Although edge centric can effectively tackle the problems of network congestion and prolonged latency that often plague cloud computing, it still has its own set of challenges. These challenges are primarily attributed to the early stages of IoT technology as observed by both [58] and [59]. Unlike cloud computing, edge computing follows a centralized architecture where all nearby IoT device requests are transmitted to an essential edge server. Additionally, edge-centric servers have restricted resources compared to cloud-centric servers and an excessive number of offloading requirements which can increase the overall task completion time for IoT devices.

2.3. EDGE COMPUTING WITH IOT

In IoT environments, sharing heterogeneous networks and sensing resources among multiple applications are common requirements [60]. Load balancing is a critical factor in meeting the ultralow latency demands of IoT applications and ensuring QoS guarantees in cloud computing [61]. Standard data mining methods that analyze two-dimensional vector data are unsuitable for managing vast data sets that feature inherent relational interdependencies, varying weights, directed edges, and heterogeneity across system elements. [62]. These limitations necessitate the use of more advanced data mining techniques that can handle complex data structures and relationships in IoT environments [63]. In cloud computing systems, a lower value results in a more balanced distribution of load which leads to improve system performance. Failure to meet a request deadline can occur if the response time surpasses the deadline or if any of the IoT devices or the user is unable to access any instance of a service. Thus, minimizing these aspects is a crucial way of load balancing in cloud computing systems and their impact on system performance cannot be overstated. [64, 65]. In the context of IoT, fog nodes may offload requests to other fog nodes to balance their workload and reduce response delays [66]. To obtain accurate results, the authors conducted multiple repetitions of experiments to calculate the average and deviation of the outcomes[67]. In their study, the authors presented a novel approach to achieve seamless load balancing within a virtual machine which consequently led to improve the throughput. This technique takes into account the job order processed by the virtual machine[68]. In cloud computing systems, round-robin load-balancing algorithms have been traditionally employed [69]. However, recent research has highlighted the potential impact of inaccuracies in expert-based weight rebalancing estimation when imperfections occur in the framework, especially in load distribution balancing between edge and cloud [70]. In the quest to improve service latency and alleviate offloading exhaustion in the emerging field of edge computing for IoT, researchers have suggested novel approaches [68]. For instance, the authors proposed a QoS- and connection-aware cloud service structure process that accounts for end-to-end QoS requirements in the cloud [69]. Furthermore, in [70], the authors introduced a load-balancing strategy that integrates data-center power consumption with diverse workloads in the cloud. This approach can ensure efficient resource allocation and enhance the overall performance of the system without compromising the accuracy of application results. Moreover, balancing the task frequency across edge nodes, as suggested in [71] can further optimize the performance of edge computing systems without sacrificing the quality of service. In addition, [72] suggests optimal policies for managing resources in edge computing systems. These policies encompass capacity allocation, load balancing, energy optimization, and quality of service assurance. All of which can be effectively implemented to enhance the efficiency and efficacy of edge computing systems

Table 2.1. Previous studies related to edge computing

citation	year	techniques	limitations
Long, C., et al.[49]	2017	Their framework technique employs group formation and video group matching techniques to enhance the accurateness of human detection within a specified timeframe	Alleviate the congestion and delays experience by traditional multimedia IoT systems, which are often limited by bandwidth constraints, however, the study did not consider latency between edge and IoT devices.
Wang, H., et al.[54]	2017	In this work, the HealthEdge task scheduling method is proposed.	The proposed algorithm evaluates whether a job should be executed in a local device or a remote cloud based on the acquired data on the human health state and attempts to decrease the overall processing time of each task as much as feasible. However, data security is not considered in the HealthEdge framework.
He, Q., et al.[48]	2019	EUAGame, a game-theoretic approach that formulates the (EUA) problem as a potential game	This study proposes the EUAGame approach to solve the Edge User Allocation (EUA) issue from the perspective of application sellers in the edge environment but this methodology faces reliability problems.
Zhang, T., et al.[50]	2019	(MEC) framework tailored for unmanned aerial vehicles (UAVs)	Aimed at facilitating the execution of computing tasks that are time-critical for terminal IoT devices within a restricted period. However, Higher tasks execution time
Chen, M. and Y. Hao [53]	2020	Mixed integer non-linear program	Propose for work offloading to an edge cloud or local processing. However, the mixed integer non-linear program continued from the old traditional methods
Yang, L., et al.[51]	2020	UAV-assisted MEC system that controls differential evolution (DE)	To determine close-optimal positions of UAVs for task offloading in IoT devices But the system was more costly.

Chen, Z. and X. Wang [1]	2020	DDPG algorithm	This study reduced the long-run average computational cost in terms of cache delay per user and power consumption by using the MEC system. However, the system did not allocate resources to solve the high latency issue.
Naouri, A., et al.[46]	2021	Framework named DCC , which consists of the device layer, cloud layer and cloudlet layer	In this study, task dependencies and user mobility were taken into account as they assessed the issue of distributing tasks and resources to several tasks of a single application inside a DCC context. However, Higher task execution time.
Liao, L., et al.[71]	2023	The double reinforcement learning computation offloading (DRLCO) algorithm	They took power consumption and task execution delay as optimization goals for mobile users in the (MEC) system, but The study did not take into account the latency delay between edge devices and IoT devices.
Jin, et al.[20]	2023	Q-learning	This study introduces a network framework that incorporates a resource allocation algorithm taking into account factors such, as (user energy consumption and delay). However, when it comes to learning tasks that require several actions, the storage and computational resources at hand may not be sufficient which could ultimately result in ineffective learning results.
Cui, et al.[72]	2023	DQN	This study proposed designing a strategy in environments to reduce latency by saving computing, storage, and resource allocation, However, the strategy was applicable only when discrete, low-dimensional work environments were provided. The edge computing environment is considered a continuous, high-dimensional environment that requires continuous actions.

To provide an overview of the related studies and the proposed work, we present Table 2, which summarizes the key offerings and evaluations of these studies.

2.4. RESOURCE ALLOCATION FOR EDGE COMPUTING

As the economy continues to thrive and technology advances, the number of devices is being seamlessly incorporated into people's lives. These gadgets have attracted attention from sectors, such as academia, industry, and government sparking a great deal of research interest. Al Ali and his team developed a solution, for managing energy in homes[73]. Their goal was to improve customer satisfaction by using data analytics and business intelligence. On the other hand, Wang and his colleagues proposed a method for scheduling tasks in smart homes and healthcare settings specifically focusing on critical situations involving elderly individuals or patients. Their approach relies on edge computing strategies that prioritize real-time considerations. Procopiou and his team developed a detection algorithm that is specifically designed for devices[74]. This algorithm uses chaos prediction and chaos theory where the Chaos Algorithm (CA) detects Flooding and distributed denial of service (DDoS) attacks. Its motivation is to establish an organization climate for home frameworks driven by IoT innovation. His associates proposed an IoT-based home security checking framework to further develop execution by lessening both the bogus positive rate and organization dormancy contrasted with customary security frameworks [75]. Two exploration strategies are presently being investigated to resolve the issue of asset assignment for edge computing without dependence on cloud communities. For example: In their work, they exhibited the significance of sifting through edge hubs to accomplish ideal asset portion and work with relocation. They expected that all clients approach edge servers situated close to figuring areas of interest and afterward, presented a Markov decision process structure to disseminate errands and assets. They likewise proposed an index-based allocation method to work on intricacy and limit correspondence above. At last, this approach permits every client to find the edge server bringing about diminished energy utilization and idleness. From the perspective of the edge node, a review was directed to look at how the size of a cluster (which alludes to the quantity of edge hubs completing errands) influences both help dormancy and energy utilization of these hubs. Their discoveries uncovered that essentially; expanding the quantity of edge hubs doesn't be guaranteed to prompt a reduction in execution idleness. In situations where the time it takes for information transmission is longer than the handling time, the help dormancy can increment at these edge hubs greatly. Subsequently, how well the framework performs depends altogether on the techniques utilized in building clusters of edge hubs and choosing the suitable hubs. To evaluate the effect of cluster techniques on the properties of edge hub groups (like size, idleness, and energy utilization), Queis and collaborators acquainted three methodologies to clustering. These techniques based on upgrading administration dormancy, limiting energy utilization in the clusters, and lessening energy use in hubs inside those groups. Li and his group fostered a heap adjusting framework called self-similarity-based load balancing (SSLB) explicitly intended for fog registering environments [76]. They zeroed in on upgrading applications running on fog foundation. They likewise presented an edge strategy and algorithm to guarantee the proficiency of SSLB. In another review, they proposed a two-stage way to deal with oversee client task planning and asset portion for nodes [77]. The underlying stage involves allocating resources locally where each edge node assigns its resources to users nearby following predefined scheduling guidelines. The ensuing stage involves framing bunches of edge hubs for clients who didn't get figure assets in the underlying designation. This study [78] introduced a two-tier resource scheduling framework, including asset coordination among different fog clusters as well as asset the executives among fog hubs inside a similar fog cluster. Furthermore, they recommended that there is a convergence between the edge layer and the terminal layer that the cell phones in these covering regions demand fog assets as well as give assets. His group [79] introduced a cloud asset algorithm that uses a Markov expectation model. The motivation behind this algorithm is to handle issues connected with task planning and burden adjusting when cloud administration hubs experience disappointments. It includes assessing hub responsibility, choosing missions in addition to hubs for relocation, and settling on the migration way. The primary objective is to work on the reliability of cloud administrations. In their study, they spotted on enhancing energy effectiveness while likewise considering dormancy execution [80]. Their model considered energy utilization and inactivity during the execution in addition to information transmission stages including gadgets, fog hubs, and cloud servers' frameworks. A review, by [81] fostered a model that considered variables like utilization, idleness, and installment costs in different fog registering networks. They utilized queueing hypothesis and activities examination to address the objective enhancement issue connected with hubs and clients. In their exploration, they fostered a real-time traffic management unloading system within an Internet of Vehicles (IoV) framework that utilizes fog nodes [82]. This advancement extends distributed computing by utilizing moving vehicles as fog hubs. In the field of organization asset designation research, they used a cloud resource optimization model for clients [83]. They utilized a two-stage enhancement way to deal with propose an asset distribution methodology for clients in a mobile cloud climate. During this stage the primary goal is to assist mobile cloud clients with making the most advantage of their assets while thinking about cost and energy limitations. In this stage, mobile cloud suppliers need to oversee servers while guaranteeing they manage energy use cutoff points to satisfy the necessities of portable cloud clients. This study [84] proposed a resource allocation model explicitly intended for administrations with an emphasis on improving utility. Ultimately, the aim is to maximize network utility by prioritizing user satisfaction. When it comes to assigning resources for moving enterprise applications to the cloud this study [85], they developed a function to optimize transmission time. Their approach focuses on reducing the duration of cloud migration which ultimately improves user satisfaction and utility. On a note, [86] they outlined an approach to edge computing resource allocation based on pricing. This approach primarily discusses how competitive service processes arise from edge nodes with networking capabilities within a resource allocation framework. Throughout their explanation, they demonstrate that their framework ensures a distribution of resources while also achieving Pareto optimization. This aligns with expectations of fairness, proportionality, and incentives for sharing in the allocation process.

Resource allocations are different for each node and they usually change dynamically depending on the resources booked by the tasks assigned to that node and the resources released at the end of the task. The capability of each node is calculated by considering the resources available to each of them as follows:

$$\tau_{CPU} = \frac{P_{CPU}}{P_{Max}} \times 100\%\ CPU \qquad (2.1)$$

$$\tau_{mi} = \frac{m_i}{m_{iMax}} \times 100\%\ \text{Internal Storage} \qquad (2.2)$$

$$\tau_{me} = \frac{m_e}{m_{eMax}} \times 100\%\quad \text{External Storage} \qquad (2.3)$$

$$\tau_j = (\varphi_1 \times \tau_{CPU}) + (\varphi_2 \times \tau_{mi}) + (\varphi_3 \times \tau_{me}) : \sum \varphi = 1 \qquad (2.4)$$

Where;

τ_j: capability of a node,

PCPU: processing power available within a node,
mi: internal storage available within a node,
me: external storage available within a node,
PMAX: total processing power of a node,
MiMax: total internal storage of a node,
MeMax: total external storage of a node,

ϕ: weight parameter for adjusting the impact degree of resources. Each node's capabilities change depending on a bunch of stuff. Nodes lose their capability when they're assigned new tasks. The amount of decrease (1−μ) is based on the ratio of the resources consumed to the total resources allocated to this node.

$$\tau_j(t+1) = (1-\mu) \times \tau_j(t) \tag{2.5}$$

where μ is a coefficient for determining the ratio of consumed resources to the total amount of resources.

The capability of node τ_j increases upon completion of a certain task. The amount of increase is based on the ratio of released resources that are dedicated to that task.

$$\tau_j(t+1) = (1+v) \times \tau_j(t) \tag{2.6}$$

where v is a coefficient for determining the ratio of the released resources to the total amount of resources. The measurement of time (from the instant a request is transmitted to the point at which the first reply to the request is received) is referred to as response time. This period is comprised of three distinct components: propagation time, waiting time, and execution time. The maximum deadline serves as a constraint necessitating that response time remain within this threshold. Hence, the anticipated duration for a task response time on a node should be shorter than the deadline.

The estimated task execution time (ET) on each node is calculated using the following equation:

$$ET = \frac{TL}{Capacity \times cores(T)} \quad (2.7)$$

Where,

TL (task length): length of task T,
Capacity: The rate at which a core can process (MIPS),
Cores(T): Task T's core count

Therefore, the estimated response time in cloud $RT_{i,j}$ is defined as the sum of waiting time and estimated task execution.

$$RT_{i,j} = WT_i + ET_{i,j} \quad (2.8)$$

Where,

WT_i : represents the waiting time for the task to be allocated,
$ET_{i,j}$: represents the estimated execution time.
The processing procedure uses the M/M/1 queuing system. In that system, if the access rate (MIPS) to the fj node is equal to $z_{f,j}$ and the processing capacity of the fj node is equal to $Capacity_{f,j}$, then the expected delay is given as a probability ratio in the following equation:

$$Latency_{i,j} = \frac{1}{Capacity_{f,j} - z_{f,j}} \quad (2.9)$$

2.5. TASK OFFLOADING

By exploiting parallel computing, An offloading-based algorithm named DDLO for MEC networks was proposed for Edge computing networks in a previous study [87]. Additionally, it [88] presented another offloading method based on Deep Q-network (DQN) that aimed to optimize network performance in energy-harvesting mobile edge computing scenarios. In a similar system configuration, [89] explored an online-based offloading strategy utilizing DQN for arbitrary task assignment. Numerous investigations that make use of DQN techniques utilize discretized channel gains as the input state vector. This practice. However, presents difficulties that pertain to the complexity of the data and the sluggishness of the learning process. The intricacies become more apparent when high channel quantization accuracy is a prerequisite, particularly in the context of MEC applications. In addition, the exploratory nature of DQN in selecting actions at each iteration makes it ill-suited for addressing problems that involve high-dimensional state spaces [90].

2.6. REINFORCEMENT LEARNING

2.6.1. Key Concepts Reinforcement Learning (RL)

Reinforcement learning (RL) has emerged as a prominent domain within the larger field of machine learning. Its objective is to explore the mechanisms by which an independent agent can acquire knowledge and enhance its decision-making abilities in a specific environment such as the realm of edge computing. The underlying mechanism behind RL involves a trial-and-error approach, as depicted in Figure 2.1, whereby the agent executes actions and evaluates their corresponding outcomes. The RL framework is predicated on the agent's ability to leverage its current state observations to make informed decisions. In edge computing, an agent's engagement with the environment yields a corresponding reward that can either be affirmative or adverse. Moreover, the agent obtains updated data concerning the current state of the environment. As a result, the agent proceeds to make incremental decisions with the primary aim of maximizing the cumulative rewards. The framework which accounts for the non-deterministic qualities of the environment, as well as the fundamental principle of causality, makes it relevant to a diverse array of artificial intelligence (AI) predicaments [91]. One of the most significant aspects of RL is that the agent can learn without any prior knowledge of optimal behavior. This permits the agent to learn using a trial-and-error approach which facilitates effective and efficient learning.

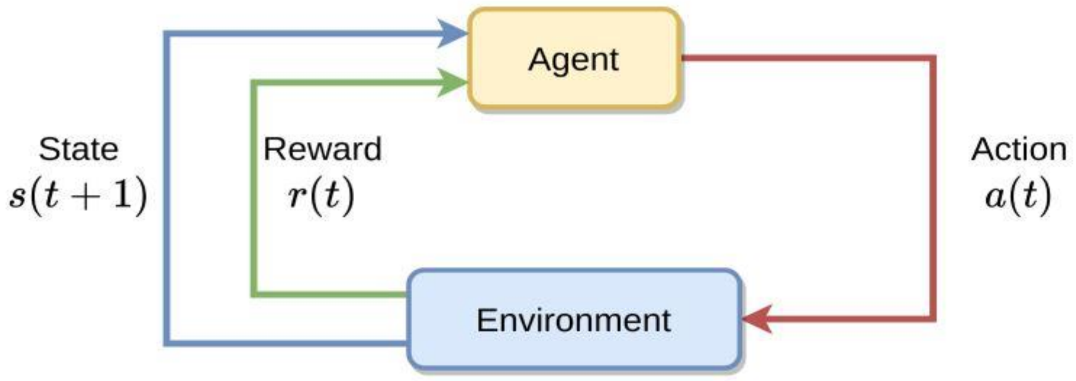

Figure 2.1. framework of reinforcement learning [92].

In contrast to dynamic programming, the optimal behavior in RL necessitates a thorough comprehension of the environment. The agent learns suitable behaviors through iterative interactions with the environment. One of the main obstacles in RL is the exploration/exploitation tradeoff which necessitates a balance between utilizing experience and exploring novel possibilities while maximizing rewards [91].

The stochastic control process can be used to model RL as a discrete-time event. A constantly analyzing agent in edge computing determines whether the environment is perceived as s(t) at every time step t. The agent selects the state of the environment from the variety of possible states after evaluating S. After contemplating the variety of possible actions, A the agent selects a(t) the appropriate action. The system confers a reward based on the agent's decision r(t). This is illustrated in Figure 2.1, where the agent perceives the new state of the environment.

2.6.2. Markov Decision Process

In RL, a mathematical framework referred to as Markov Decision Process (MDP) underpins the process. The MDP incorporates probabilistic control algorithms with discrete temporal context as shown by a 5-tuple. The MDP is composed of five elements: state space, action space, transition probability function, reward function, and discount factor.

- S: This set encompasses all conceivable states that the environment can take.
- A: This set represents the actions that the agent can execute.
- T: $S \times A \times S \rightarrow [0, 1]$. The transition probability matrix maps every state-action pair at a given time "t" to a probability distribution over the subsequent states at time "t+1" in the context of edge computing.
- R: $S \times A \times S \rightarrow R$. The reward function denoted as R, generates a numerical value r(t) that depends on the present state s(t) at time t, the action taken at that time and the subsequent state s(t+1).
- γ: $\gamma \in [0, 1]$. Closer to 1 γ is the future rewards which are of higher value does the agent. A value of γ closer to 1 implies that the agent values future rewards more highly.

It provides a comprehensive model for understanding the complex process of decision-making through the use of the MDP. An MDP calculates the resulting reward for an agent involved in edge computing by taking the current environment and action into account. It is crucial to note that MDPs adhere to the Markov property which simplifies the calculation of transition distributions between states. As a result of the application of this property, the future state of the system is solely governed by the current state of the system whereas any past states are either ignored or considered irrelevant [93].

Policy Definition: In the realm of decision-making, policy represents a set of principles or guidelines that a governing entity employs to ascertain its course of action. At each juncture of this decision-making process, an agent relies upon its established policy to determine the appropriate actions to take. From a mathematical standpoint, a policy is denoted as a function. It is often symbolized as "π" which maps the various states encountered to corresponding actions. There are two principal types of policies:

- A deterministic policy denoted by π: $S \rightarrow A$, maps every state S to a corresponding step or action.
- On the other hand, a stochastic policy is denoted by π: $S \times A \rightarrow [0, 1]$. It describes the probability of selecting a particular action in a given state which is S.

A big part of the edge computing decision-making process is the policy π which acts as a roadmap that connects possible states to the probability distribution of viable actions. By interacting with the environment, the agent adheres to the policy which leads to a series of states, actions, and rewards (τ). As it corresponds to the policy π, the trajectory shows how the agent interacts with the environment.

2.6.3. Reward and Return

The reward function specified in the MDP section holds paramount significance in assessing desirable behavior. The way edge computing determines advantageous results is by generating a numerical output based on the current state, the chosen course of action, and the result. A discount is applied to the cumulative reward from a given trajectory to figure out the value.

$$G(t) = \sum + \infty k = 0 \gamma k r(t + k + 1) \tag{2.10}$$

In previous research [91], RL methods have been extensively used to identify policies (π ∗) that maximize expected returns. Value functions are utilized in RL to assign a value to states, actions, or combinations to determine the agent's interest in such states, actions, or combinations. It is important to note that state-action pairs are valued in edge computing based on what agents' predictions will occur from a particular state. In addition, the agent's future benefits are dependent on the decisions it makes as discussed in [94]. Consequently, value functions are established based on a particular policy. Precisely, the state-value function "V π (s)" signifies the value of states under a given policy π.

This function is approximated by computing the anticipated return that arises from commencing in state S and persistently pursuing policy π. Formally, this can be expressed as:

$$\mathrm{V}\,\pi\,(\mathrm{s}) = \mathrm{E}\pi[\mathrm{G(t)}|\mathrm{s(t)} = \mathrm{s}] \tag{2.11}$$

In the domain of edge computing, evaluating a policy π at a particular point in time t involves using the indispensable metric of expected return denoted as Eπ. In this metric, agents are estimated to earn multiple rewards during an episode according to a given policy π. The Q-function (or action-value function) can also be used in assessing the value of a state-action pair, in addition to the state-value function. Represented as Qπ(s, a), the Q-function approximates the anticipated reward for taking action a from state s and subsequently adhering to policy π:

Formally, this can be expressed as:

$$Q\,\pi\,(s,a) \;=\; E\pi[G(t)|s(t) \;=\; s, a(t) \;=\; a] \qquad (2.12)$$

2.6.4. Optimal Value Functions

In the domain of policies dictating value functions, the paramount function that attains the highest value for all states is acknowledged as the optimal value function and denoted as V ∗ (s). This function epitomizes the utmost value of V π (s) among all policies π:

$$V\;*\;(s) \;=\; max\,\pi\,V\,\pi\,(s) \qquad (2.13)$$

In edge computing, the Q ∗ (s, a) function denotes the action value that yields the maximum expected reward achievable by the agent when it starts from a given state “s” and performs a specific action “a” regardless of the policy employed.

$$Q\;*\;(s,a) \;=\; max\,\pi\,Q\,\pi\,(s,a) \qquad (2.14)$$

Acknowledging that V ∗ (s) represents the maximum anticipated total reward at the beginning of a state. It is of utmost importance in the context of edge computing. Accordingly, the greatest value of Q∗ (s, a) available among all feasible actions corresponds to V ∗ (s). The precise mathematical connection between V ∗ (s) and Q∗ (s, a) can be succinctly expressed as follows:

$$V * (s) = \max a\, Q * (s, a) \tag{2.15}$$

2.6.5. Optimal Policy

In edge computing, the association between Q∗ (s, a) and π ∗ is of critical importance. Specifically, Q∗ (s, a) indicates the expected rewards that an agent would receive when selecting an action while always following the optimal policy [95]. A policy designed to maximize the expected reward from states, on the other hand, selects an action that maximizes the current reward.

Therefore, if the optimal Q-function Q∗ (s, a) is available, the optimal policy π ∗ (s) can be computed directly utilizing the equation provided below:

$$\pi * (s) = argmax\, a\, Q * (s, a) \tag{2.16}$$

2.6.6. Advantage Functions

In the field of edge computing, the Advantage function holds significant importance in reinforcement learning (RL) as it aids in assessing the effectiveness of a chosen action compared to other potential alternatives. The Advantage function is represented as Aπ(s, a) which is closely associated with a policy π and measures the incremental progress achieved by selecting action a in state s as opposed to opting for a random action and adhering to policy π for an extended duration [95]. The mathematical definition of the Advantage function entails the difference between the Q-function and the V-function:

$$A\pi (s, a) = Q\pi (s, a) - V\pi (s) \tag{2.17}$$

In algorithms such as the Proximal Policy Optimization algorithm, the Advantage function is of paramount importance. In our research, we aim to tackle the challenging issue of computation offloading by employing this technique. By employing this approach, we enhance system performance by optimizing computation resource allocation between mobile devices and cloud servers. However, it is pertinent to note that our study will only concentrate on resource management aspects and other factors associated with edge computing will be outside the scope of our research.

2.6.7. Bellman Equations

The Bellman equation constitutes a pivotal component of RL algorithms and holds a prominent position in the RL literature. Its importance stems from the fact that it defines the association among the value function of a specific state and that of the subsequent state. The dissection of the value function into two distinct components specifically, the instant gratification and the reduced value of the upcoming state has proven to be an essential building block for the creation of RL algorithms. This equation not only serves as a fundamental cornerstone for RL algorithm design but also theatres a critical role in the determination of optimal decision-making policies.

Mathematically:

$$V \pi (s) = E[r(t+1) + \gamma V \pi (s(t+1)) | s(t) = s] \tag{2.18}$$

Similarly for the Q-function:

$$Q \pi (s,a) = E[r(t+1) + \gamma E a' Q \pi (s(t+1), a') | s(t) = s, a(t) = a \tag{2.19}$$

As part of edge computing, optimal value functions can be derived using the Bellman equation. In this scenario, the approach involves identifying the action that maximizes the value as opposed to computing the expectation based on a particular policy [95].

$$V * (s) = max\, a\, E[r(t+1) + \gamma V * (s(t+1)) | s(t) = s] \tag{2.20}$$

$$Q * (s,a) = E[r(t+1) + \gamma\, max\, a' \, Q * (s(t+1), a') | s(t) = s, a(t) = a] \tag{2.21}$$

The Bellman equation possesses a unique attribute that simplifies the calculation of the value function. It transforms the value function's computation into a dynamic programming problem, thereby facilitating the recursive determination of optimal solutions from less complicated sub-problems in the field of edge computing. This attribute significantly streamlines the value function computation process, thereby boosting solution efficiency.

2.6.8. Off-Policy vs On-Policy Learning

To address the issue of optimizing offloading decisions in edge computing, RL algorithms have been categorized according to the learning policy employed by the agent. Off-policy methods permit the agent to use a behavior policy that differs from the target policy through which the algorithm aims to identify. This approach enables the agent to behave randomly in the environment, while off-policy methods utilize experience replay from previous samples collected using various policies to determine the optimal policy.

Utilizing off-policy methods such as the Q-learning algorithm offers significant advantages as they promote exploration and enhance sample efficiency without the necessity of collecting new experiences every time as the policy is always updated. [91, 96].

Unlike off-policy techniques, on-policy methods aim to improve the agent's policy for interacting with the environment by aligning the behavior policy with the target policy. The agent generates samples using its existing policy and then enhances it by leveraging the collected experience resulting in an updated policy that is used to gather new data while disregarding previous experience. The same policy is refined iteratively until it attains an optimal policy in a process referred to as convergence [96]. An exemplary illustration of an on-policy-based methodology encompasses the proximal policy optimization algorithm.

2.6.9. Model-Based vs Model-Free RL

To facilitate RL, the function representing the environment model predicts state transitions and associated rewards [95]. MDPs serve as a fundamental framework for edge computing systems where the environmental model is composed of two integral components: the transition probability matrix T and the reward function R. These key elements hold significant importance in the optimal decision-making process within the realm of edge computing.

Using an environment model for planning purposes enables the agent to anticipate possible future actions and make informed decisions. This approach falls under the category of Model-based methods [95]. A notable example of such methods is MuZero [97] which is a derivative of the renowned AlphaZero [98] algorithm.

Although model-based learning has several advantages, it is primarily challenged by the fact that the agent must learn the environment model through trial and error. During interactions with the environment, approximations of the state transition and reward functions are made. However, when a model is learned to approximate the actual model, the learned policy can become biased and exhibit suboptimal performance when applied to the actual model. As a result, discovering the optimal policy can be hindered. Attaining model-based learning still presents an inherent challenge [95]. On the other hand, model-free approaches do not depend on a model for policy inference. Rather, they aim to approximate the policy while neglecting the estimation of environmental dynamics such as state transitions and reward functions. This fundamental dichotomy highlights the trade-offs and considerations that underpin the design of effective machine-learning algorithms.

When considering the potential of leveraging experiences gained from interactions with the environment, agents have the option to estimate V-functions and deduce a policy or they may choose to directly estimate the policy. Notably, the latter approach offers a significant advantage that it can learn an optimal policy without relying on a model of the environment. As a result, this technique is versatile and can be effectively employed across a range of environments including novel or unforeseen scenarios

[99]. This particular trait has led to the widespread adoption of model-free reinforcement learning algorithms which have undergone extensive enhancement and validation in comparison to model-based approaches [95, 99]. Furthermore, consolidating its position as a reliable and adaptable solution for edge computing applications.

2.7. DEEP REINFORCEMENT LEARNING

Artificial Deep Neural Networks (DNNs) are used in deep learning (DL) to replicate the human brain's learning process. With the advantage of DL, the field of machine learning has achieved unprecedented outcomes and outperformed conventional machine learning approaches including computer vision, speech recognition, and language translation [100]. DNNs have demonstrated exceptional performance surpassing human performance in some cases. The exceptional aptitude of deep neural networks to comprehend and recognized intricate patterns of features within voluminous and complex input data especially visual imagery is a remarkable achievement. The complexity of these networks is inherent in their multiple layers of abstraction where intricate operations are executed to extract meaningful information from the data.

DRL is a variant of reinforcement learning that unifies the methodologies of RL and Deep Learning (DL). DRL has facilitated remarkable breakthroughs in artificial intelligence (AI) as demonstrated by the impressive performance of DeepMind's MuZero algorithm. MuZero has achieved superhuman performance in Chess, Go, Shogi, and Atari games. Thus, highlighting the potential of DRL algorithms in enhancing AI capabilities [97].

The utilization of deep learning (DL) techniques has enabled the scalability of (RL) to previously an unattainable decision-making situation especially those involving high-dimensional state and action spaces such as raw sensor data from robots, sounds, or images [101]. RL in edge computing employs deep learning (DL) to model policies or other value functions as deep neural networks (DNNs) for identifying the optimal policy. DNNs excel in handling high-dimensional data that is typical of real-world edge computing problems making them well-suited for estimating learned functions. Additionally, DNNs can learn incrementally as the agent acquires more experience in the environment. Furthermore, it enhances their effectiveness in edge computing applications.

DRL has been effectively implemented to tackle a multitude of problems across diverse domains. In the realm of robotics, DRL has enabled robots to learn optimal policies using camera inputs as evidenced [102].

DRL has garnered substantial attention in diverse domains including but not limited to self-driving vehicles, natural language processing, and healthcare. One notable application of DRL is in training agents for critical driving skills such as object and lane detection, trajectory optimization, and control of steering, acceleration, and braking among other essential tasks [103]. RL techniques have also been applied to natural language processing (NLP) tasks such as text summarization and question answering with notable success [104, 105]. In the healthcare sector, RL agents have been implemented for diverse purposes including but not limited to automated medical diagnoses, dynamic treatment regimens for chronic diseases, drug discovery, and more [106].

In addition, there has been a growing trend of utilizing DRL in various engineering applications including energy optimization [107] and industrial process control [108]. The focus of this study is on multi-access edge computing and aims to investigate the potential of DRL for resource management within this specific domain.

2.7.1. Deep Deterministic Policy Gradient (DDPG)

The issue of efficiently managing large, continuous state and action spaces in edge computing while utilizing a deterministic policy has been addressed through the introduction of the Deep Deterministic Policy Gradient (DDPG) algorithm. This innovative algorithm was first presented in [90]. Instead of using probability distribution for actions, the deterministic nature of DDPG enables the actors to directly compute actions [109]. This approach builds on the actor-critic technique used in DQN which helps avoid iterative optimization at every step and determines optimal Q-values in continuous action space cases. DDPG consists of two neural networks: the actor network $\pi\psi$, also known as the policy network, and the critic network $Q\theta$. Both of which have specific network parameters represented by ψ and θ [110].

The $\pi\psi$ actor-network is responsible for planning a state to an action. The actor is deterministic taking the state as an input and providing the action as an output. An action's Q-value is calculated by mapping the state with the action (a pair of states and actions) of the critic network. To ensure stable learning, DDPG employs a replay buffer and target network similar to DQN. However, it updates the target networks gradually and softly by adjusting the soft target network weights to match the learned network weights [111]. In conclusion, DDPG combines two approaches DQN and deterministic policy gradient to enable the effective learning of continuous actions.

2.8. CHALLENGES

Employing RL to facilitate offloading decisions in edge computing holds immense potential for optimizing the allocation of computing resources and enhancing system performance. The reviewed studies in this literature survey have exhibited the efficacy of RL algorithms in optimizing offloading decisions leading to substantial improvements in system performance.

PART 3

METHODOLOGY

This research introduces an innovative methodology for dynamic task offloading decisions with the primary objective of minimizing the average task completion time. The proposed approach is designed to ascertain the optimal allocation of computational resources within the edge computing environment for specific scenarios. The chapter commences by elucidating the system model, delineating various factors influencing task completion duration. Each factor is systematically expounded upon, and the decision-making process is meticulously formulated. Aligned with the system model, the recommended decision mechanism employs the Deep Deterministic Policy Gradient (DDPG) algorithm. To assess the effectiveness of the proposed approach, a series of experiments are conducted through the simulation of the system model.

3.1. RESEARCH MODEL

This study centers on a system architecture encompassing an Internet of Things (IoT) edge-IoT user framework, characterized by multiple IoT devices and edge servers, as illustrated in Figure 3.1. The system is organized into three tiers, with IoT devices occupying the lower layer and edge servers positioned at the upper layer, functioning within a Wi-Fi network. In the context of a fixed-location IoT device, the set of devices is denoted as I = [1, 2, ... I], where each device is assigned a fixed position. Our research hypothesis posits that inherent limitations in resources such as computational power, storage, and battery capacity impede IoT devices from autonomously executing computational tasks. Moreover, even if executed, these tasks may incur significant latency due to the limited resources, thereby presenting critical issues in time-sensitive communication scenarios [112]. To address this concern, the current research proposes offloading all tasks generated by IoT devices to multiple edge servers with superior performance. This research overlooks the concept of local processing on IoT devices and presupposes that every device has the capacity to produce tasks solicitation simultaneously.

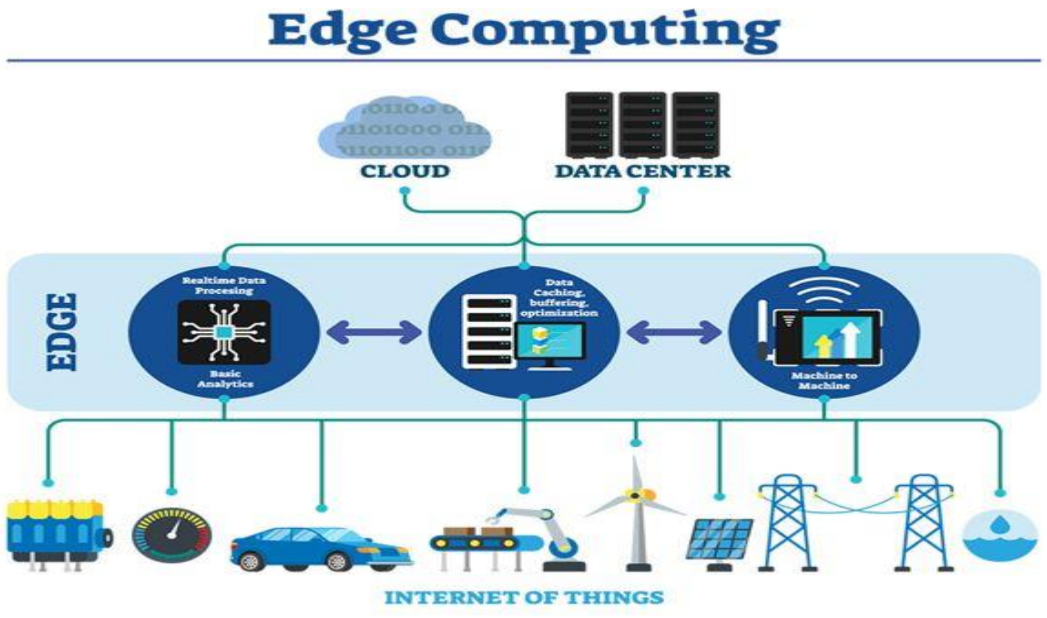

Figure 3.1. System Architecture [113].

Upon initiation, the assignment is structured and programmed for processing at the edge server, employing the offloading resolution algorithm executed at the edge. Following the execution of the mission, IoT devices can retrieve the results from the edge server.

3.1.1. Environment of IoT-Edge Computing

- Navigation data, sourced from the CRAWDAD dataset, has been amassed by users of IoT devices, which subsequently offload tasks to avail computational services from the edge server. Subsequent to the processing of the requested task, users receive the processed task from the edge server and proceed to offload another new task to the edge server.
- When an IoT device submits a task request from the edge, the edge allocates computational resources to process the task, functioning at a rate of (6.3 * 1e7 bytes/sec). This allocation encompasses computing power, including CPUs,

memory, GPUs, and storage within the system. These resources are instrumental in executing computations, performing tasks, and processing data. The system edge server accommodates a maximum number of concurrent users, ensuring an equitable and efficient distribution of computing resources such as processing power and memory.

- Task migration to an alternative edge server becomes viable under specific circumstances, such as when a designated edge server faces impediments in task completion or if relocation proves to be more efficient on another server. The migration speed is, however, capped at 1e9 bytes per second, constrained by available bandwidth. Dynamic adjustments in job distribution among edge servers, constrained by bandwidth limitations, are implemented to optimize resource utilization and enhance overall system performance.
- The task offloading process, initiated when a task is requested from an IoT device or a mobile user, involves sequential steps:

  a. Step 1: Commence the offloading of a task to an edge server.

  b. Step 2: The edge server receives the requested task (2.7 * 1e4 bytes), where the value "2.7 * 1e4 bytes" denotes the size of the data being transmitted.

  c. Step 3: The requested task is processed (1.08 * 1e6 bytes), with "1.08 * 1e6 bytes" representing the size of the data produced as a result of task processing.

  d. Step 4: The mobile user or IoT device imminently receives the requested task (96 bytes), with "96" indicating the size of the data being transmitted back.

  e. Step 5: Disconnect (default).

  f. Step 6: The requested task has been transferred or migrated to another edge server, signifying the relocation of the initially offloaded task for processing. This may occur if the initial edge server encounters difficulties in successful task completion or if load balancing techniques necessitate task redistribution.

3.1.2. Task Offloading Model

This section delves into the assessment of the task offloading model from the perspective of an edge server, considering processing, queueing, and transmission delays. Consistent with the model proposed by Mao et al. (2017), the computation task requested by the i-th IoT device is represented as a tuple, Ai = [Li , Li ′ , Ci], where Li denotes the bits required to specify the size of the input-data, Li′ represents the bits needed to describe the task's output, and Ci indicates the number of CPU cycles necessary to execute the task. The definitions of the terms utilized in this study are elucidated in Table 3.1.

Table 3.1. Definitions of Terms.

Terms	Definition
$\bar{F}t\ es$	The capacity required by the edge to process all tasks queued at the edge in time t
$Ft\ es$	At time t, the current edge capacity is available
$Fi\ es$	Allocation of computing capacity at the edge to IoT devices
Li	Input-data size for IoT device i task definition
Ci	IoT device i fi compute task required number of CPU cycle
I	Defining the task of an IoT device based on its input data size in bits
$Nt\ 'es$	Currently, there are no unfinished tasks on the edge t
$Li\ '$	IoT device i output-data size is defined by the number of bits in the task
$Nt\ es$	At the time, how many tasks have been completed in the edge t
$Ti\ es$	IoT device i processing task at edge - computation delay
$Wi\ es$	There is a queueing delay until a task is retrieved at the edge for IoT device i
$Q\ es$	Processing queue at edge

3.1.2.1. Edge Server

Assuming that we are addressing an Internet of Things (IoT) device denoted as "i," it can be observed that the edge server assigns a computation task to it with a specific capacity expressed in terms of "f_i^{es}" cycles per second. The delay incurred during the execution of this task at the edge server can be calculated by aggregating the total computation delay, which can be mathematically expressed as follows:

$$T_i^{es} = \frac{Ci}{f_i^{es}}. \tag{3.1}$$

Upon arrival at the edge server, the designated task is placed in the edge processing queue, denoted as Q^{es}. Operating on a first-in, first-out processing model, this queue employs a discernment mechanism to identify the suitable processing destination. It is presumed, in accordance with this premise, that the queue consistently monitors the number of outstanding requests. In accordance with [37] proposed model, the retrieval time for task i in the edge queue Q^{es} can be determined by calculating W_i^{es}, where there are i tasks in the queue. This can be achieved using the following equation:

$$W_i^{es} = \sum_{n=1}^{i-1} T_n^{es} \tag{3.2}$$

The total delay for an IoT device i at the edge server is a combination of the processing delay and the queuing delay, represented by $\{T_i^{es}, W_i^{es}\}$.

3.1.3 Problem Formulation

The focal point of this investigation revolves around the intricate challenge of minimizing task completion time when employing IoT devices with edge computing capabilities. A notable constraint emerges when the singular edge server is congested, impeding the attainment of minimal completion times for IoT devices. This challenge emanates from the inherent limitation of the solitary edge server, which possesses finite resources, and the surge in offloading requests adversely affects task completion time, consequently depriving IoT devices of achieving minimal completion durations.

To surmount this extant obstacle, a pragmatic strategy involves the distribution of a subset of designated tasks, allocating them for processing across both the singular and multiple edge servers. This strategic allocation aims to streamline the offloading process, thereby enhancing overall system performance. The principal objective of this study is to devise a judicious decision-making mechanism for offloading, underpinned by the Deep Deterministic Policy Gradient (DDPG) algorithm. This mechanism optimizes the task offloading ratio from IoT devices to the multitude of edge servers.

The proposed approach constitutes an innovative departure from conventional computation offloading techniques. The paramount goal is the reduction of the average total duration of task completion.

3.2. OFFLOADING DECISION MECHANISM

In this segment, we present our approach to addressing the issue delineated in the preceding section, utilizing the DDPG algorithm as our offloading decision mechanism. In this study, we introduce our method that incorporates different factors with the aim of achieving peak efficiency, including computation delay, transmission delay, and queueing delay. Our approach employs a decision-making mechanism, which is developed using the conventional framework of RL as expounded by [114] and relies on a MDP as the underlying model.

3.2.1. Deep Deterministic Policy Gradient (DDPG)

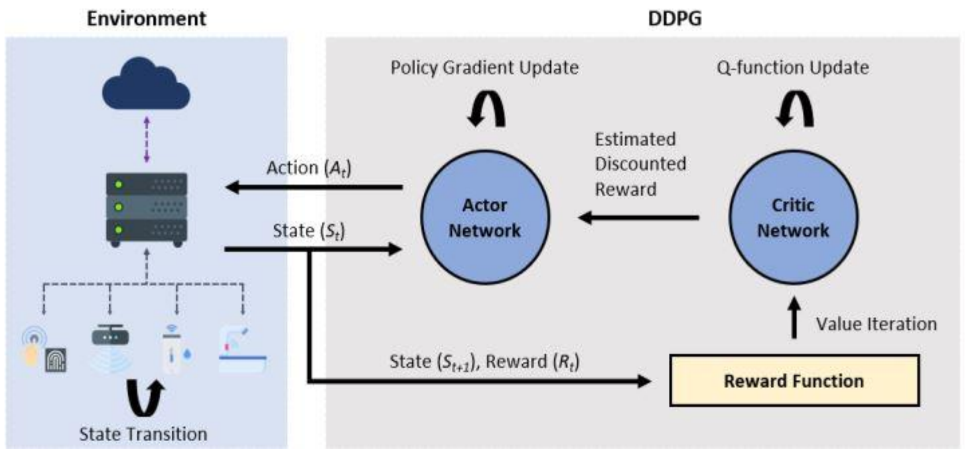

Figure 3. 2. Dynamic task offloading mechanism based on DDPG [1].

The DDPG algorithm, introduced in this study [90], represents an evolved iteration of the Deep Q-Network (DQN), incorporating an actor-critic methodology. Addressing the intricacies associated with expansive state and action spaces in RL, the (DDPG) algorithm emerges as a strategic solution by employing a deterministic policy. This obviates the need for iterative optimization to acquire optimal Q-values at each time step, as the algorithm progressively refines the values of state-action pairs, consequently enhancing the learning agent's behavior. The discrete nature of ascertaining the feasibility of task migration from IoT devices to the edge is distinct from the continuous resource allocation problem. For the optimization of (IoT) tasks, our approach leverages the DDPG algorithm, facilitating the offloading decision-making process from IoT gadgets to multiple edge servers. Notably, the algorithm's capacity to update model weights at each step facilitates rapid adaptation to a dynamic environment, rendering the DDPG algorithm instrumental in efficiently allocating computational resources.

Each time slot necessitates a decision on whether to offload tasks in chronological order, given the chronological ordering of task arrivals. According to the DDPG algorithm, the action at, state St, and reward Rt are all determined based on time t, accomplished through the deployment of (RL) agent on the edge server. The DDPG algorithm, as depicted in Figure 3.2, guides the decision-making process for task offloading. As discussed in Chapter 2.3.2, the DDPG algorithm employs both actor and critic neural networks. The actor network receives state information, and the actor network outputs action information. The critic network, utilizing the state and action as inputs, estimates their values. Subsequently, the environment acquires the new state St+1 and reward Rt from the actor network when the actor network initiates an action At (specific task offloading decision). The state St+1 is updated through this process, serving as the environment's environmental state. In the critic network, the reward function imparts the optimal state value to the agent, influencing its behavior. The actor network, informed by feedback from the critic network, enhances its performance in the realm of policy improvement. This feedback is contingent upon the Q-value or reward associated with a specific action. Our study contributes to this enhancement by introducing a novel policy gradient and a fresh Temporal Difference (TD) metric, quantifying the discrepancy between projected and realized rewards. The intricacies of the state space, action space, and reward function are comprehensively elucidated in this report, along with a detailed exposition of the mechanisms underlying our proposed methodology.

3.2.1.1. State Space

The state space St at a given time t is intricately determined by the present state of both IoT gadgets and the edge server. The edge server state encompasses various factors, including the available computing resources for each edge server, the migration bandwidth between edge servers, the offloading goal of each mobile user or IoT device, and the location of each mobile user. Formalizing this, the state space St at a given time t is defined as:

$$St = [F_t^{es},\ \bar{F}_t^{es}, N_t^{es}, N'^{es}_t] . \tag{3.3}$$

where the server's available capacity at time F_t^{es}, the required capacity to process all tasks in the edge queue at time $\bar{F}_t^{es}$, the number of tasks completed on the edge server at time N_t^{es}, and the number of tasks that remain unfinished at time N'^{es}_t.

3.2.1.2. Action Space

Within the decision-making framework for agents, a critical decision arises at time t within the action space At. This encompasses the execution of all computations at the edge, with the following components:

$$At = [\sum_{n=1}^{i} T_n^{es}, \alpha] \tag{3.4}$$

The indicator α within the action space represents the computational power required for tasks offloaded from IoT gadgets to edge servers.

3.2.1.3. Reward Function

The reward function, formulated to quantify the immediate reward for undertaking an action, is expressed as:

Rt = R(St, At, S{t+1}) (3.5)

The 'Reward *(Rt*)' is a function denoted as 'Reward function (R*(St, At, S{t+1}))*' measuring the immediate benefit or cost associated with taking action At while in state *St* and transitioning to state *S{t+1}*. This metric serves as an evaluative tool to gauge the desirability of an action within a specific state.

3.3. EXPLORATION AND EXPLOITATION

Exploration and exploitation are pivotal concepts in the realm of (RL). The DDPG algorithm adeptly combines both exploration and exploitation to acquire optimal policies within continuous action spaces. The algorithm integrates exploration by introducing noise to the actor's policy during action selection, promoting the exploration of diverse actions and preventing fixation on suboptimal solutions. Exploitation, on the other hand, involves utilizing learned knowledge to maximize expected cumulative rewards. In DDPG, exploitation transpires when the actor selects actions based on the learned deterministic policy, refined through training to maximize expected rewards. The actor-critic architecture of DDPG amalgamates both these facets, with the actor network learning action-selection policies and the critic network estimating value functions for evaluating actions. This feedback loop informs the actor about expected rewards, facilitating iterative improvements.

3.4. PROPOSED MODEL ALGORITHM

This study scrutinizes the offloading of tasks and the allocation of computing resources in the context of (IoT) and edge computing. The selection of the optimal node, prioritized based on proximity to the user network, and the offloading of tasks, as well as the allocation of resources, are conducted through the DDPG deep reinforcement learning algorithm. The proposed model for resource allocation and task offloading, utilizing the DDPG algorithm, is depicted in Figure 3.3. The procedural steps include the initialization of networks, prioritizing nodes for IoT users, and iterative updates of network parameters. The algorithm ensures convergence through continual adaptation to the dynamic environment and the fulfillment of specific conditions. The comprehensive elucidation of the proposed model encompasses considerations of state spaces, action spaces, and reward functions, providing a robust foundation for understanding the intricacies of our innovative methodology.

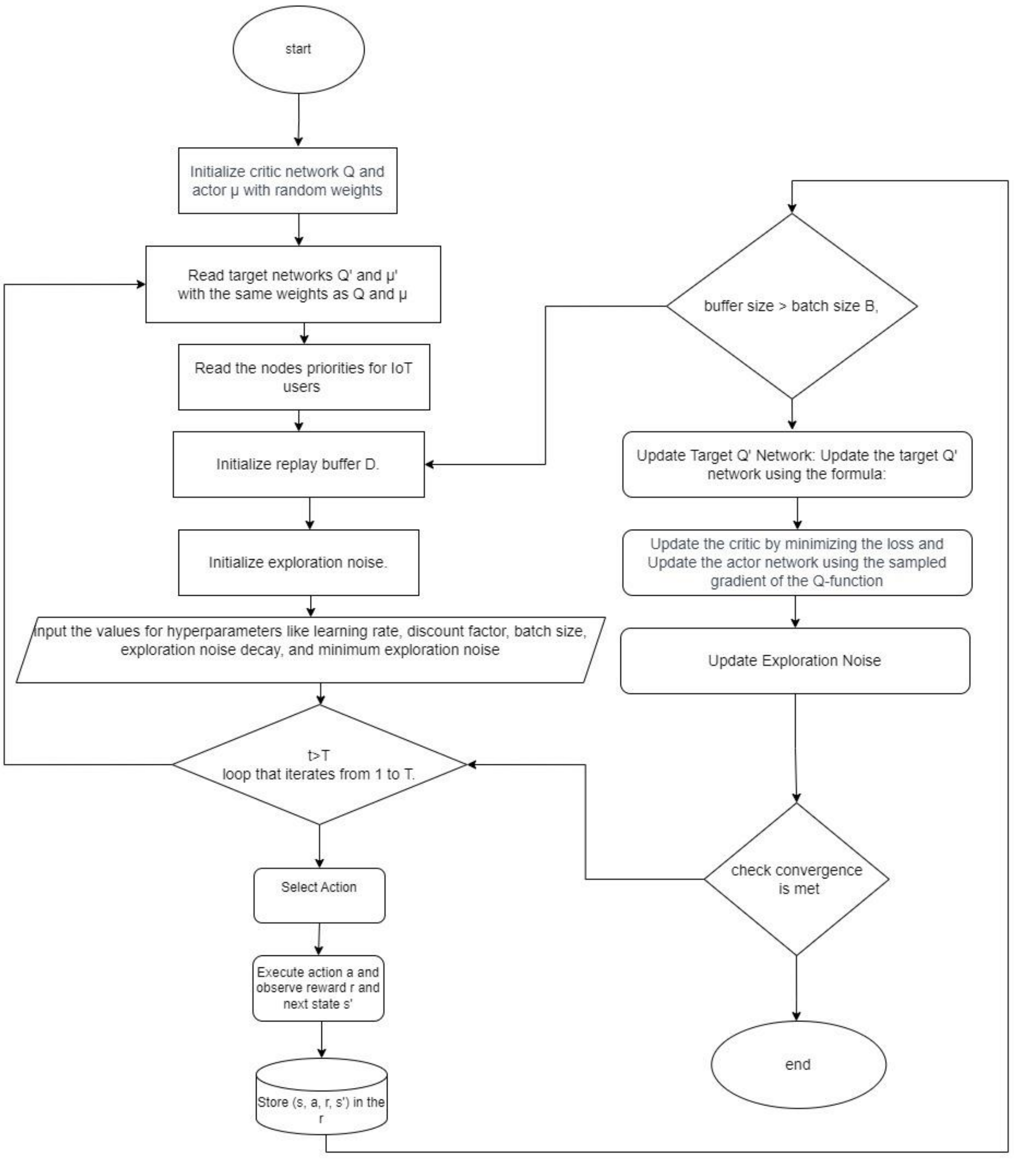
start
Initialize critic network Q and actor μ with random weights
Read target networks Q' and μ' with the same weights as Q and μ
Read the nodes priorities for IoT users
Initialize replay buffer D.
Initialize exploration noise.
input the values for hyperparameters like learning rate, discount factor, batch size, exploration noise decay, and minimum exploration noise
t>T loop that iterates from 1 to T.
Select Action
Execute action a and observe reward r and next state s'
Store (s, a, r, s') in the r
buffer size > batch size B,
Update Target Q' Network: Update the target Q' network using the formula:
Update the critic by minimizing the loss and Update the actor network using the sampled gradient of the Q-function
Update Exploration Noise
check convergence is met
end

Figure 0.3. Dynamic decision-making mechanism flowchart for resource allocation and dynamic task offloading.

3.5. EXPERIMENTS

This section evaluates the effectiveness of the proposed task offloading mechanism in addressing the previously outlined challenges. Throughout our analysis, we delve into the intricacies of the system model, the decision-making process for offloading, and pertinent simulations. The implementation utilizes the Python programming language on an HP ZBook laptop, tailored for (RL) training.

3.5.1. Simulation Settings

The subject matter revolves around an (IoT) system operating at the edge-user interface. The network encompasses 10 IoT devices initially, progressively increasing to 20 and 30, with tasks offloaded to 10 edge servers. Computational resources for each edge server are capped at (6.3 * 1e7 bytes/sec), indicating a processing capacity of 63 Mbps. The edge server responsible for providing computational resources to mobile consumers operates at (6.3 * 1e7 bytes/sec). Tasks can be transferred between edge servers with a bandwidth constraint of (1e9 byte/sec). The code utilizes the CRAWDAD dataset hosted by IEEE Dat-aPort to provide relevant information about the environment, including user navigation patterns.

Table 0.2. The simulation parameters.

Parameter	**Value**
Number of IoT devices	10, 20, 30
Number of edges	10
Capacity of the edge server	63 Mbps
Number of episodes	100
Number of steps per episode	3000
Actor network rate	0.001
Critic network rate	0.002
reward discount	0.99
Replay buffer size (memory capacity)	10000
Batch size	64

3.5.2. Simulation Process

The study employs a training routine with 100 episodes, each mandating 3000 steps for every task sequence. The environmental conditions reset at the outset of each episode. At each step, the agent selects an action in accordance with a pre-determined policy π, updating the environmental status using an array of IoT devices, an edge server, and multiple edge servers. Post-action selection, the edge server either handles the requests or delegates them to other edge servers if it cannot process the request. The edge server updates its computation capability and records the number of completed tasks. The ensuing step involves generating a numerical reward, a new action is selected, and the cycle resumes.

PART 4

RESULTS AND DISCUSSION

In this chapter, we presented a dynamic task offloading decision mechanism based on the DDPG for optimizing edge tasks and resource allocation in an IoT edge system.

4.1. DISCUSSION

Our simulation results underscore the efficacy of the proposed DDPG-based task offloading mechanism in optimizing edge computing tasks and resource allocation within an IoT edge system. The mechanism surpasses baseline strategies, showcasing more stable convergence, reduced average task completion times, and a heightened count of successfully completed tasks across various scenarios.

The adaptability and dynamic decision-making process inherent in the DDPG algorithm render it particularly well-suited for addressing the challenges posed by IoT edge computing. In environments characterized by resource constraints and dynamic workloads, our mechanism utilizes the continuous action space of DDPG to dynamically adjust the task offloading ratio. This adaptive approach optimizes task completion time and resource utilization.

Furthermore, our mechanism demonstrates a capacity to adapt to evolving conditions, such as an increasing number of IoT users. It achieves this by intelligently distributing tasks among edge servers, a crucial aspect for ensuring efficient resource utilization and minimizing latency in IoT edge computing environments. Our research significantly contributes to advancing edge computing in IoT, presenting a dynamic and efficient task offloading mechanism that enhances the responsiveness and capability of edge systems in handling diverse workloads effectively.

4.2. REWARD VARIATION

This segment of the research investigates reward value changes, reflecting an increasing number of completed tasks during specific intervals in the training process, each interval representing one episode. The dependent variable, measured on the x-axis by the number of episodes, and on the y-axis by the cumulative reward obtained by the agent across all episodes, demonstrates the average reward obtained through multiple runs of the training process. During initial training stages, the agent explores various actions to comprehend associated rewards.

.

That is, at each iteration, the average execution time for each task is calculated, so:

- If the average execution time during the current iteration is higher than the average execution time in the previous iteration, then:

Reward= Reward – N

Where N is the number of tasks in current iteration.

- If the average execution time during the current iteration is less than the average execution time in the previous iteration, then:

Reward= Reward + N

This analysis focuses on specific scenarios, including 10 user 10 edge, 20 user 10 edge, 30 user 10 edge, and 60 user 10 edge. With increasing training processes, a noticeable rise in the reward's value is observed in Figures 4.1, 4.2, 4.3, 4.4 and 4.5.

4.3. EVALUATION

We conducted a comprehensive simulation study, comparing our DDPG-based mechanism to two baseline strategies: QL and DQN algorithms. The results demonstrated the superiority of our mechanism in terms of stability, average task completion time, and the average count of successfully completed tasks across various scenarios with varying numbers of IoT users. Illustration and evaluation of algorithm performance in order to demonstrate the advantages of the proposed algorithm, we compare its performance with some reinforcement learning algorithms. Each algorithm encountered the same network change in the simulation. Between every two changes, the network model was trained or replicated a certain number of times. In the following figures, orange represents the DDPG algorithm considering user cooperation, respectively. Blue and green indicate DQN and QL respectively. These algorithms from deep reinforcement learning are taken for comparison because they are typical and efficient. After continuous research and promotion, these algorithms have shown high efficiency and good performance in solving dynamic optimization problems [20] [72].

the following figures shows the reward curve of the DDPG algorithm. The abscissa represents the episode turn, and the ordinate is the average reward of the current turn. It can be seen that DDPG can realize a rapid increase in rewards and eventually converge on a stable result; thus, this algorithm is effective.

In addition, the results showed the effectiveness of DDPG as the number of users increased.

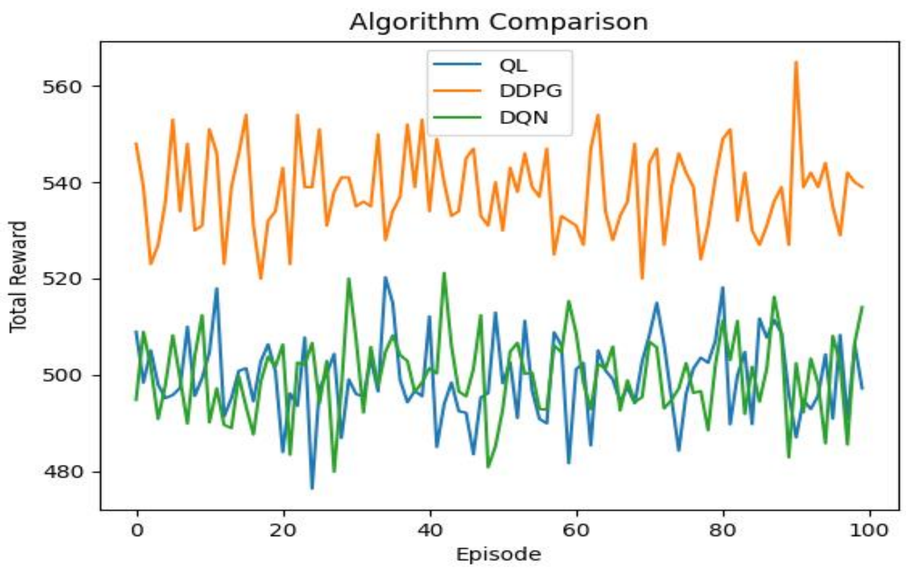

Figure 4.1. 10 edge 10 users.

In this changing number of users, DDPG has excelled and demonstrated efficiency by achieving an average of 560 units while QL and DQN lag behind at 500 units each. The reason DDPG excels in this scenario is because it returns to handling effective control and policies. With fewer users, DDPG was able to adjust and release it as a higher system requirement, reducing it.

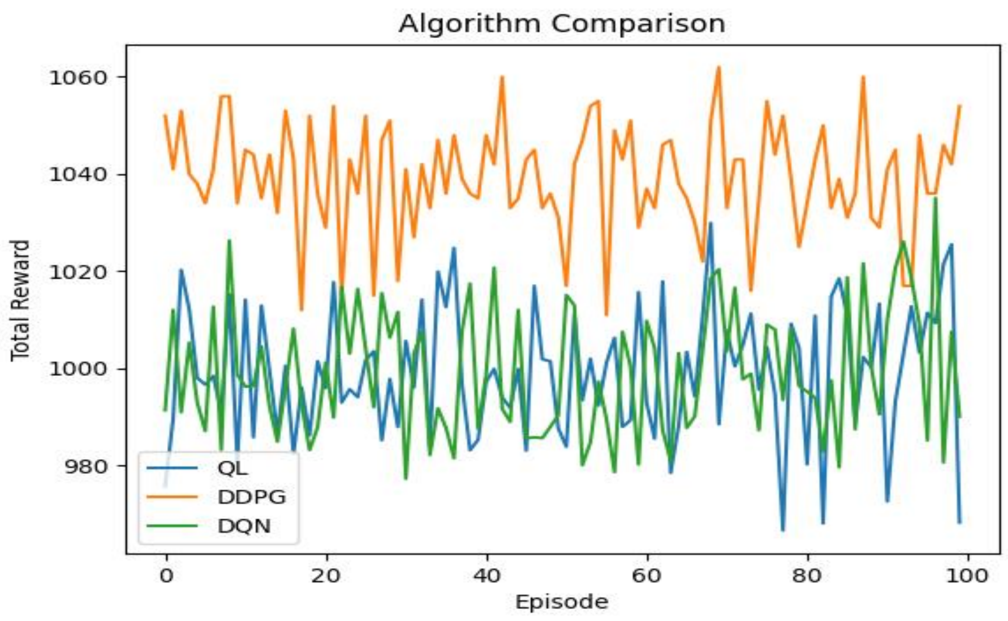

Figure 0.2. 10 edge 20 users.

As the number of users increased to 20, the complexity of continuous control increased. But DDPG continued to shine, recording around 1,050 units, while QL and DQN struggled slightly and recorded around 1,000 units each.

DDPG's ability to adapt and fine-tune policies remains its main advantage. He was able to navigate increasing complexity and fine-tune its policy, achieving superior performance compared to other algorithms.

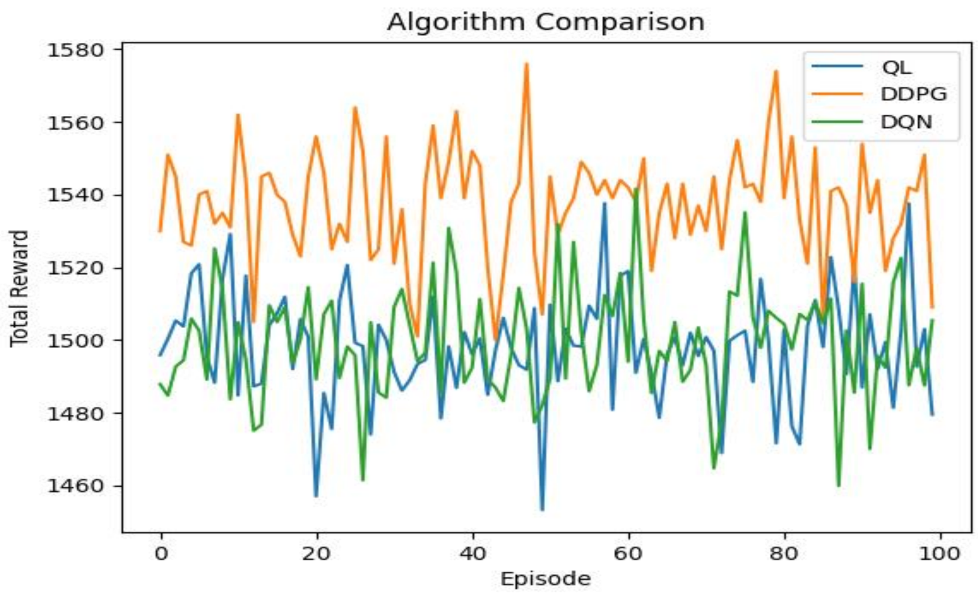

Figure 0.3. 10 edge 30 users.

As the number of users increased to 30, the challenges in continuous control escalated. But DDPG once again managed to shine, recording an average delay of around 1,540 units. Conversely, QL and DQN had difficulty keeping up with the increase and recorded around 1,500 units each.

DDPG's ability to adapt to increasing complexity and fine-tune policies ensures that it can withstand challenging edge computing environments. Its continued outperformance has demonstrated its ability to achieve outstanding performance in scenarios that require precise and constant control.

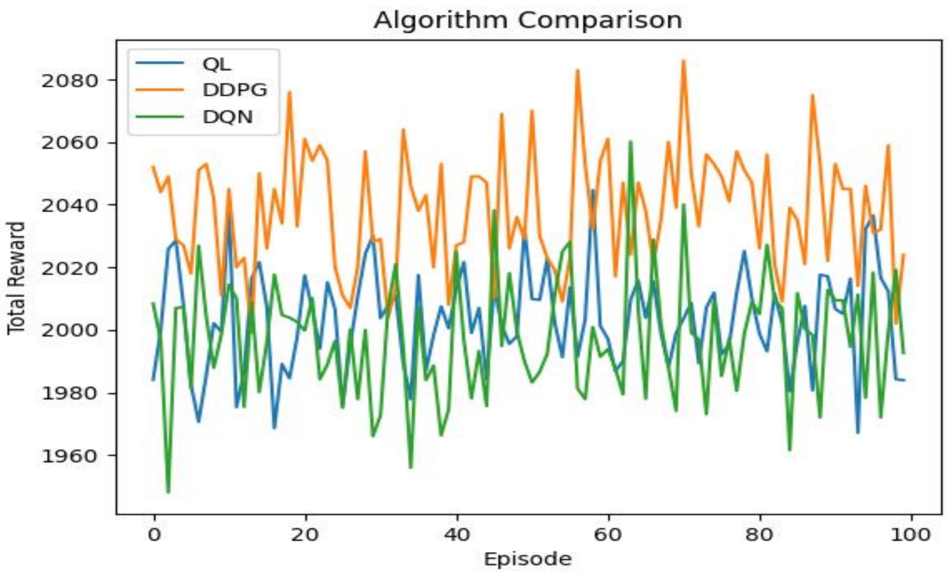

Figure 4.4. 10 edge 40 users.

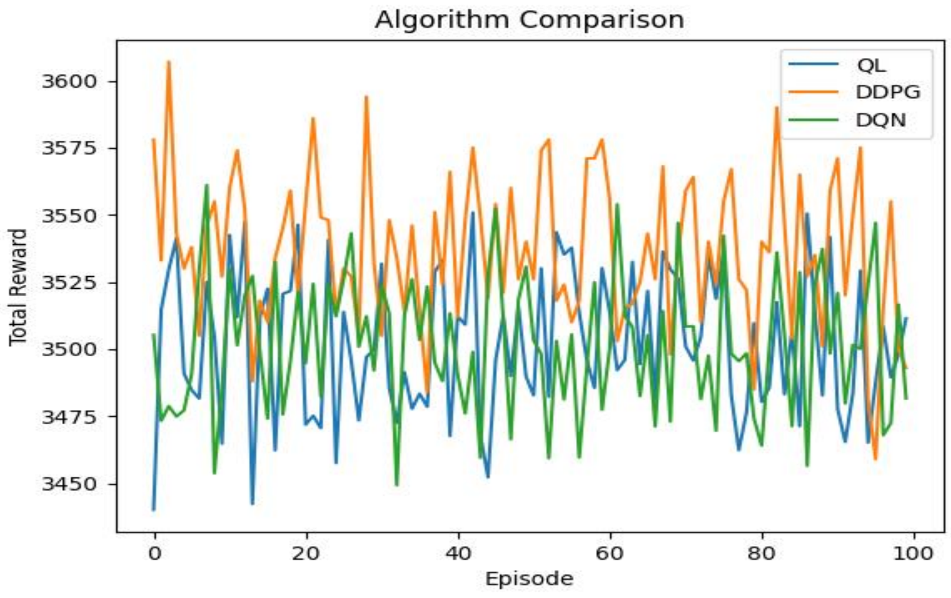

Figure 4.5. 10 edge 70 users.

In the subsequent figures 4.4 and 4.5, we increase the number of IoT users in our network to 40 and 70 respectively. Notably, our proposed algorithm consistently outperforms QL and DQN algorithms under these conditions.

Table 4.1 shows the use of a number of Internet of Things users in order to observe the extent of the superiority of our proposed algorithm in terms of obtaining cumulative reward units in the shortest time compared to the QL and DQN algorithms.

Table 4.1. Effectiveness Between Three Algorithms.

Number of users	**DDPG**	**QL**	**DQN**
10	56 units	500 units	500 units
20	1050 units	1000 units	1000 units
30	1450 units	1500 units	1500 units
40	2060 units	2000 units	2000 units
70	3575 units	3500 units	3500 units
TIME	23%	10%	20%

4.4. TASKS OFFLOADING RESULTS

To gain insight into our DDPG-based mechanism's decision-making process, we analysed the task offloading ratio employed in different scenarios. Our analysis reveals that our DDPG-based mechanism dynamically adjusts the task offloading ratio based on the number of IoT users and system conditions. In scenarios with a higher number of IoT users, the mechanism tends to allocate a greater proportion of tasks to the server, optimizing task completion time. Conversely, in scenarios with fewer IoT users, the mechanism relies more on edge processing, minimizing latency. This adaptive approach ensures optimal task offloading under varying circumstances, enhancing the overall efficiency of the IoT edge system.

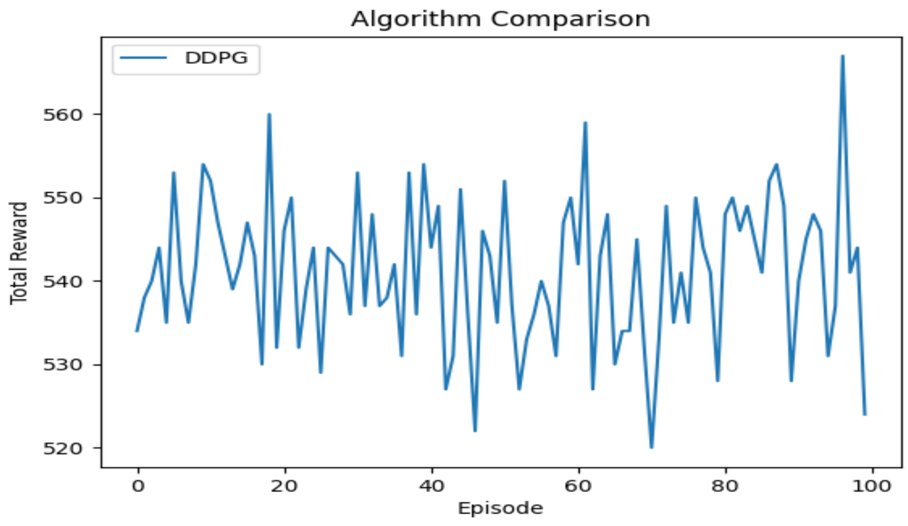

Figure 4.6. 10 user 10 edge.

We note from Figure 4. the increase in the value of the reward with the progression of the number of iterations, and this means that the average execution time for the tasks arrived during this period decreases, and we also note that this improvement in the value of the reward is during a period of only100 iterations, and this indicates that the proposed methodology learns very quickly and during the number of iterations Little compared to other reinforcement learning methods that need a large number of iterations during the training phase until reaching the optimal solution for the desired goal. Reducing the number of iterations in the training phase is very beneficial for CPUs as it avoids CPU fatigue and exhaustion.

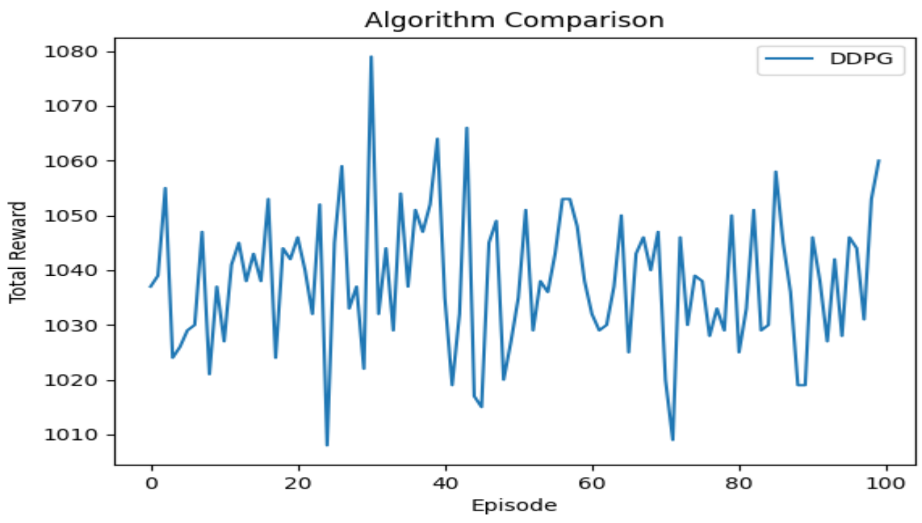

Figure 4.7. 20 user 10 edge.

We note from Figure 4.7 that with increasing the number of users to 20, the proposed methodology maintained its ability in terms of learning and maximizing the value of the reward during a few iterations, as in Figure 4.6, but here the value of the reward decreases compared to the previous Figure 4.6, and the reason for this is due to the fact that increasing the number of users leads to an increase in the load and therefore an increase in the number of tasks to be performed.

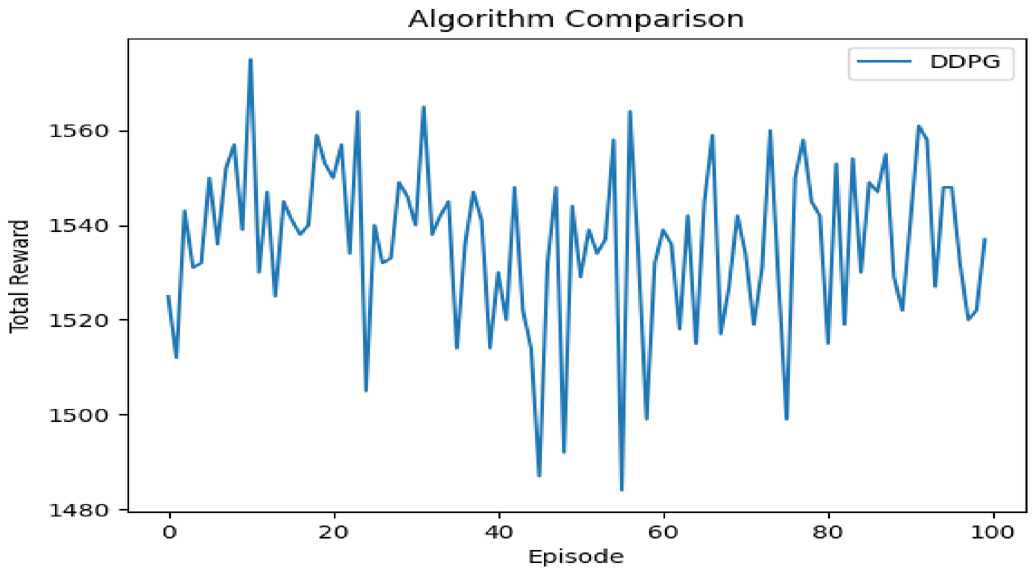

Figure 4.8. 30 user 10 edge.

The same applies to Figure 4.8, where the number of users was increased to 30 while maintaining the number of iterations during the training phase. We also notice the increase in the value of the reward, but with a lower value than Figure 4.7, due to the increase in the number of tasks to be performed.

4.5. VARIANCE IN EXPLORATION

Within the realm of reinforcement learning (RL), specifically concerning (DDPG) algorithm, the concept of "variance in exploration" involves the deliberate introduction of randomness or uncertainty into the exploration process. This strategic incorporation aims to motivate the RL agent to explore diverse actions and states within its environment. Exploration holds significant importance in RL, serving as a means for the agent to uncover near-optimal strategies by experimenting with various actions and assimilating knowledge from their outcomes. The relevance of variance in exploration becomes pronounced when confronted with action spaces housing a multitude of potential actions.

In RL algorithms, including DDPG, a policy is employed, indicating that for a given state, the objective is to produce a specific action without introducing variability. However, effective exploration of the action space demands the introduction of a certain degree of randomness. In the case of DDPG, this exploration is often achieved by introducing noise to the actor network's action output. This injected noise introduces variability in the selected actions, enabling the agent to traverse diverse regions within the action space. Various methods exist to introduce exploration variance in DDPG, with one common approach involving the utilization of an Ornstein-Uhlenbeck Process. This process generates correlated noise, aiding the agent's exploration while maintaining a certain level of consistency over time.

The introduction of noise to the action output serves as a method to introduce randomness, and the degree of noise can be regulated by adjusting its deviation. The level of variance during exploration plays a pivotal role in determining the balance between exploring possibilities and exploiting known ones. Striking an optimal balance in the amount of variance utilized for exploration is crucial for the agent to effectively learn a policy. Excessive variance may impede exploration, making it challenging for the agent to identify optimal solutions. Conversely, insufficient variance may lead to exploration causing the agent to become entrenched in suboptimal policies. Achieving this balance becomes a critical aspect in fine-tuning DDPG and similar reinforcement learning algorithms, as it significantly influences the agent's ability to learn and discover optimal policies, especially within continuous action spaces.

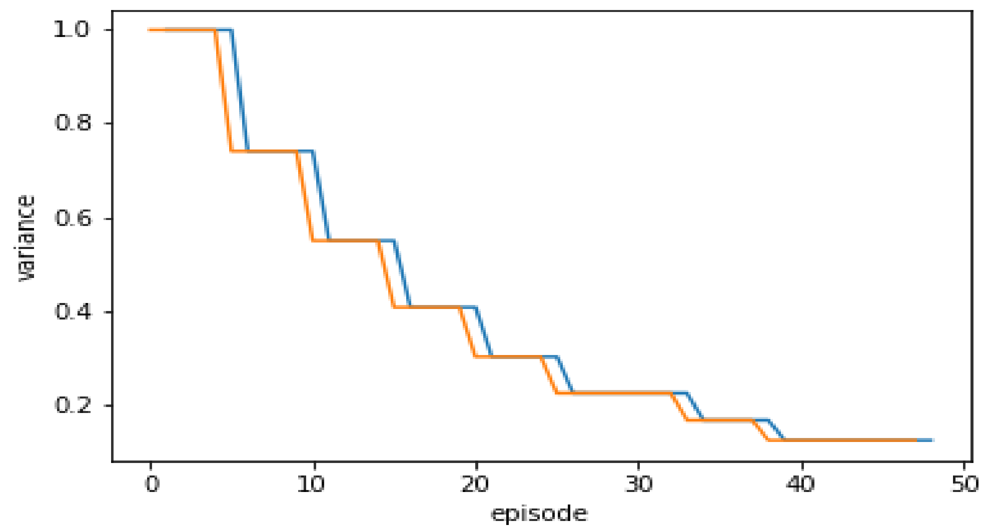

Figure 4.9. 10 user 10 edge.

We note from the figure 4.9 that the value of the variance decreases with the increase in the number of iterations. Therefore, the system learns and adjusts its parameters until a stable load is reached at iteration 40, and then maintains the stability of the load during the remaining period. Therefore, the proposed methodology was able to achieve fairness in the distribution of resources to tasks over a period of time. Ideally, very few repetitions.

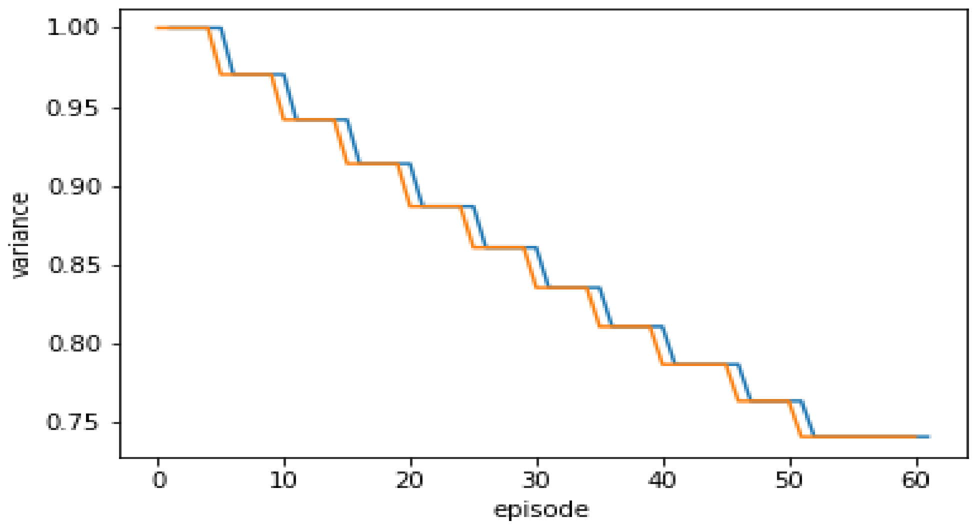

Figure 0.10. 20 user 10 edge.

We note from the figure 4.10 that the value of the variance decreases with the increase in the number of iterations. Therefore, the system learns and adjusts its parameters until a stable load. As we mentioned earlier, the increase in the number of users leads to an increase in the number of tasks. Therefore, we notice an increase in the variance value compared to the previous case when the number of 10 users.

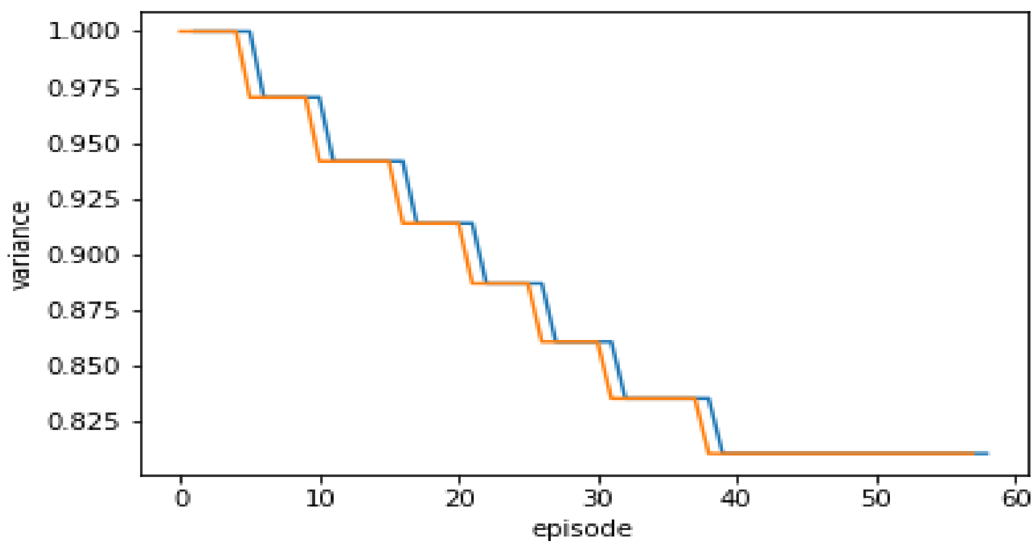

Figure 4.11. 30 user 10 edge.

We notice from the figure 4.11 that the value of the variance decreases with the increase in the number of iterations. Therefore, the system learns and adjusts its parameters until a stable load is reached at iteration 40. As we mentioned earlier, the increase in the number of users leads to an increase in the number of tasks, and therefore we notice an increase in the value of the contrast compared to the previous two cases. But we note here that despite the increase in the number of tasks, the proposed methodology has reached the stage of stability of the load for iteration 40. The reason for this is that the process of generating tasks is done randomly and balancing the load is related to the size of the incoming tasks and the requirements of each task. Therefore, the requirements of the tasks in this scenario are less than In the previous two scenarios, therefore, the stability stage was reached at iteration 40, and this confirms the flexibility of the proposed methodology and its ability to deal with the incoming tasks according to the requirements of each task.

We conclude from previous figures 4.9 and 4.10 and 4.11 the reward remains relatively stable, with the variance remaining around 1.0 for the first few episodes. As training progresses, the reward variance decreases, indicating that the algorithm is becoming more stable. The baseline variance also decreases over time, which is a good sign that the predicted reward is becoming more accurate. Overall, these results suggest that the RL algorithm is working well and the agent is successfully learning to perform the task.

4.6. LATENCY

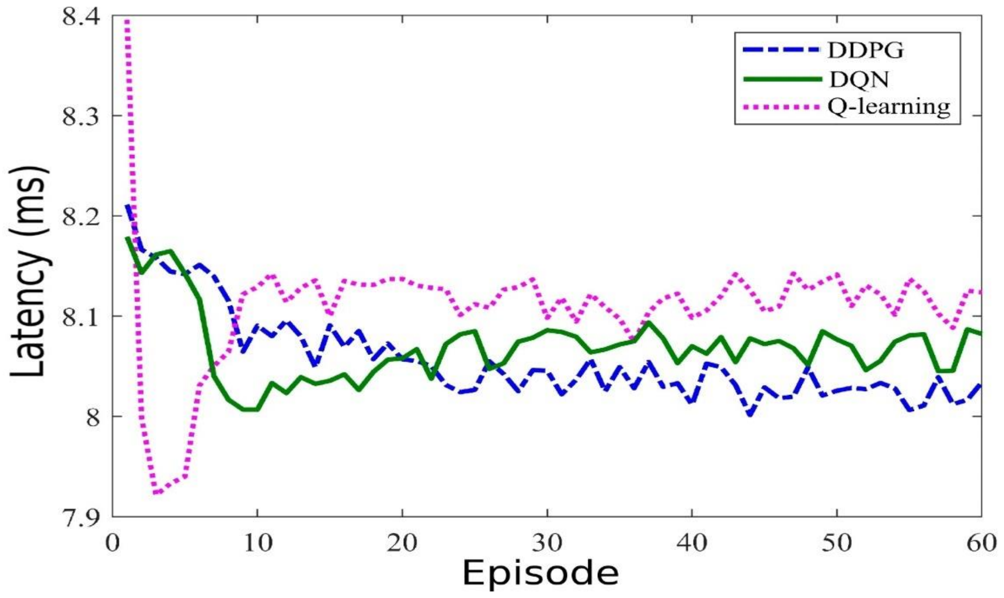

Figure 4.12. Comparison of Three Algorithms for Reducing Response Time in Edge Computing and User Interaction.

This section provides a comparative examination of the algorithm introduced in this paper alongside those in references [72] and [20] all under identical conditions, with a focus on the completion time of the resource allocation strategy task. The initial step involves the calculation of task completion times for various algorithmic resource allocation strategies. Figure 4.12 illustrates the average task completion times for different algorithms across varying numbers of tasks. Upon close inspection of Figure 4.12, it becomes evident that, irrespective of the number of tasks, the algorithm proposed in this paper exhibits the shortest average task completion time in comparison to the algorithms presented in references [20] and [72]. Notably, as the number of tasks increases, the growth rate of the average task completion time remains relatively modest. This observed trend can be attributed to the efficacy of the computing resource allocation strategy outlined in this paper, which adeptly prioritizes task migration to IoT devices with the swiftest response times. Contrastingly, reference [72], utilizing (DQN), introduced edge computing and a comprehensive architecture in Mobile Edge Computing (MEC). However, it did not yield improvements in the objective function solution process, resulting in extended solution times. Reference [20], employing (QL), a deep reinforcement learning algorithm, delved into the general optimization of computation efficiency. Nevertheless, detailed analyses and optimizations of time consumption calculation models for local tasks and user tasks were lacking.

This comparative analysis sheds light on the superior performance of the proposed algorithm in minimizing average task completion times across diverse task quantities, emphasizing the effectiveness of the computing resource allocation strategy introduced in this research.

PART 5

CONCLUSION AND FUTURE WORK

In the context of edge-IoT environments, effective task offloading management is paramount for ensuring timely and optimal task completion. While existing research has primarily explored individual task offloading strategies, our study introduces a dynamic mechanism that facilitates collaborative task offloading between IoT users and edge servers. Our proposed mechanism employs the Deterministic Deep Policy Gradient (DDPG) algorithm to optimize offloading ratios, minimizing the average task completion time.

Our experiments, encompassing training iterations and performance metrics, including reward variation, average task completion time, and the average amount of completed work, demonstrate the efficacy of our decision mechanism. Utilizing object detection tasks and distinct neural networks, our comprehensive decision mechanism scheme exhibits remarkable convergence during training, resulting in significant improvements in task completion time and the ability to process a greater number of tasks per second.

As part of future research endeavors, our focus is on further enhancing the efficiency of our task offloading decision mechanism. This enhancement will involve considerations such as task prioritization, energy consumption, and the mobility patterns of IoT devices within our system model. Additionally, expansion of our system model to accommodate multiple edge servers is planned, as our current model assumes a configuration of 10 servers.

REFERENCES

[1] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," *EURASIP Journal on Wireless Communications and Networking,* vol. 2020, pp. 1-21, 2020.

[2] K. Ashton, "That ‘internet of things’ thing," *RFID journal,* vol. 22, pp. 97-114, 2009.

[3] K. Xiao, Z. Gao, W. Shi, X. Qiu, Y. Yang, and L. Rui, "EdgeABC: An architecture for task offloading and resource allocation in the Internet of Things," *Future Generation Computer Systems,* vol. 107, pp. 498-508, 2020.

[4] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems,* vol. 87, pp. 278-289, 2018.

[5] K. Matsui and H. Nishi, "Error correction method considering fog and edge computing environment," in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS),* 2019, pp. 517-521.

[6] T. Zheng, J. Wan, J. Zhang, and C. Jiang, "Deep reinforcement learning-based workload scheduling for edge computing," *Journal of Cloud Computing,* vol. 11, p. 3, 2022.

[7] K. Sadatdiynov, L. Cui, L. Zhang, J. Z. Huang, S. Salloum, and M. S. Mahmud, "A review of optimization methods for computation offloading in edge computing networks," *Digital Communications and Networks,* 2022.

[8] B. Rababah, T. Alam, and R. Eskicioglu, "The next generation internet of things architecture towards distributed intelligence: Reviews, applications, and research challenges," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC),* vol. 12, 2020.

[9] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE,* vol. 107, pp. 1738-1762, 2019.

[10] S. Safavat, N. N. Sapavath, and D. B. Rawat, "Recent advances in mobile edge computing and content caching," *Digital Communications and Networks,* vol. 6, pp. 189-194, 2020.

[11] R. U. Rasool, H. F. Ahmad, W. Rafique, A. Qayyum, and J. Qadir, "Security and privacy of internet of medical things: A contemporary review in the age of

surveillance, botnets, and adversarial ML," *Journal of Network and Computer Applications,* p. 103332, 2022.

[12] I. T. Christou, N. Kefalakis, J. K. Soldatos, and A.-M. Despotopoulou, "End-to-end industrial IoT platform for Quality 4.0 applications," *Computers in Industry,* vol. 137, p. 103591, 2022.

[13] O. Ali, M. K. Ishak, M. K. L. Bhatti, I. Khan, and K.-I. Kim, "A Comprehensive Review of Internet of Things: Technology Stack, Middlewares, and Fog/Edge Computing Interface," *Sensors,* vol. 22, p. 995, 2022.

[14] P. Chakraborty, R. N. Dizon-Paradis, and S. Bhunia, "ARTS: A Framework for AI-Rooted IoT System Design Automation," *IEEE Embedded Systems Letters,* vol. 14, pp. 151-154, 2022.

[15] N. Sarrafzade, R. Entezari-Maleki, and L. Sousa, "A genetic-based approach for service placement in fog computing," *The Journal of Supercomputing,* vol. 78, pp. 10854-10875, 2022.

[16] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, *et al.*, "Videoedge: Processing camera streams using hierarchical clusters," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 115-131.

[17] Q. Liang, W. A. Hanafy, A. Ali-Eldin, and P. Shenoy, "Model-driven cluster resource management for AI workloads in edge clouds," *ACM Transactions on Autonomous and Adaptive Systems,* vol. 18, pp. 1-26, 2023.

[18] P. P. Gaikwad, J. P. Gabhane, and S. S. Golait, "A survey based on Smart Homes system using Internet-of-Things," in *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, 2015, pp. 0330-0335.

[19] K. Moser, J. Harder, and S. G. Koo, "Internet of things in home automation and energy efficient smart home technologies," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2014, pp. 1260-1265.

[20] Y. Jin and Z. Chen, "A Fast Resource Allocation Algorithm Based on Reinforcement Learning in Edge Computing Networks Considering User Cooperation," *Electronics,* vol. 12, p. 1459, 2023.

[21] D. C. Nguyen, S. Hosseinalipour, D. J. Love, P. N. Pathirana, and C. G. Brinton, "Latency optimization for blockchain-empowered federated learning in multi-server edge computing," *IEEE Journal on Selected Areas in Communications,* vol. 40, pp. 3373-3390, 2022.

[22] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network," *IEEE Internet of Things Journal,* vol. 8, pp. 2238-2251, 2020.

[23] W.-C. Chiang, Y. Li, J. Shang, and T. L. Urban, "Impact of drone delivery on sustainability and cost: Realizing the UAV potential through vehicle routing optimization," *Applied energy,* vol. 242, pp. 1164-1175, 2019.

[24] F. Song, Z. Ai, H. Zhang, I. You, and S. Li, "Smart collaborative balancing for dependable network components in cyber-physical systems," *IEEE Transactions on Industrial Informatics,* vol. 17, pp. 6916-6924, 2020.

[25] J. Oueis, E. C. Strinati, and S. Barbarossa, "Small cell clustering for efficient distributed cloud computing," in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, 2014, pp. 1474-1479.

[26] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1-7.

[27] M. Babar and M. Sohail Khan, "ScalEdge: A framework for scalable edge computing in Internet of things–based smart systems," *International Journal of Distributed Sensor Networks,* vol. 17, p. 15501477211035332, 2021.

[28] W. Ahmad, A. Rasool, A. R. Javed, T. Baker, and Z. Jalil, "Cyber security in IoT-based cloud computing: A comprehensive survey," *Electronics,* vol. 11, p. 16, 2021.

[29] N. Almusallam, A. Alabdulatif, and F. Alarfaj, "Analysis of privacy-preserving edge computing and Internet of Things models in healthcare domain," *Computational and Mathematical Methods in Medicine,* vol. 2021, 2021.

[30] X. Li, L. Zhao, K. Yu, M. Aloqaily, and Y. Jararweh, "A cooperative resource allocation model for IoT applications in mobile edge computing," *Computer Communications,* vol. 173, pp. 183-191, 2021.

[31] T. A. S. Srinivas, A. D. Donald, I. D. Srihith, D. Anjali, and A. Chandana, "The Rise of Secure IoT: How Blockchain is Enhancing IoT Security."

[32] Y. He, Y. Wang, C. Qiu, Q. Lin, J. Li, and Z. Ming, "Blockchain-based edge computing resource allocation in IoT: A deep reinforcement learning approach," *IEEE Internet of Things Journal,* vol. 8, pp. 2226-2237, 2020.

[33] A. Sarah, G. Nencioni, and M. M. I. Khan, "Resource Allocation in Multi-access Edge Computing for 5G-and-beyond networks," *Computer Networks,* vol. 227, p. 109720, 2023.

[34] J. Chen, T. Wu, M. Shi, and W. Jiang, "Porf-ddpg: Learning personalized autonomous driving behavior with progressively optimized reward function," *Sensors,* vol. 20, p. 5626, 2020.

[35] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things," From active data management to event-based systems and more:

Papers in honor of Alejandro Buchmann on the occasion of his 60th birthday, pp. 242-259, 2010.

[36] S. Nižetić, P. Šolić, D. L.-d.-I. González-De, and L. Patrono, "Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future," *Journal of Cleaner Production,* vol. 274, p. 122877, 2020.

[37] A. Jaddoa, G. Sakellari, E. Panaousis, G. Loukas, and P. G. Sarigiannidis, "Dynamic decision support for resource offloading in heterogeneous Internet of Things environments," *Simulation Modelling Practice and Theory,* vol. 101, p. 102019, 2020.

[38] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE communications surveys & tutorials,* vol. 19, pp. 1628-1656, 2017.

[39] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet of Things Journal,* vol. 5, pp. 1275-1284, 2018.

[40] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology,* vol. 68, pp. 856-868, 2018.

[41] B. Pourghebleh, V. Hayyolalam, and A. Aghaei Anvigh, "Service discovery in the Internet of Things: review of current trends and research challenges," *Wireless Networks,* vol. 26, pp. 5371-5391, 2020.

[42] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, *et al.*, "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions," *Computer Networks,* vol. 195, p. 108177, 2021.

[43] S. Hu and G. Li, "Dynamic request scheduling optimization in mobile edge computing for IoT applications," *IEEE Internet of Things Journal,* vol. 7, pp. 1426-1437, 2019.

[44] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Computer Networks,* vol. 130, pp. 94-120, 2018.

[45] Q. You and B. Tang, "Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things," *Journal of Cloud Computing,* vol. 10, pp. 1-11, 2021.

[46] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim, and H. Ning, "A novel framework for mobile-edge computing by optimizing task offloading," *IEEE Internet of Things Journal,* vol. 8, pp. 13065-13076, 2021.

[47] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for internet of things," *IEEE Transactions on Cloud Computing,* vol. 9, pp. 1050-1060, 2019.

[48] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems,* vol. 31, pp. 515-529, 2019.

[49] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia IoT systems," *IEEE Transactions on Multimedia,* vol. 20, pp. 1126-1139, 2017.

[50] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Transactions on Industrial Informatics,* vol. 16, pp. 5505-5516, 2019.

[51] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet of Things Journal,* vol. 7, pp. 6898-6908, 2020.

[52] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge computing in IoT-based manufacturing," *IEEE Communications Magazine,* vol. 56, pp. 103-109, 2018.

[53] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications,* vol. 36, pp. 587-597, 2018.

[54] H. Wang, J. Gong, Y. Zhuang, H. Shen, and J. Lach, "Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes," in *2017 IEEE International Conference on Big Data (Big Data),* 2017, pp. 1213-1222.

[55] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications,* vol. 37, pp. 668-682, 2019.

[56] L. Lei, H. Xu, X. Xiong, K. Zheng, and W. Xiang, "Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-IoT edge computing system," *IEEE Internet of Things Journal,* vol. 6, pp. 5345-5362, 2019.

[57] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Joint computation offloading and scheduling optimization of IoT applications in fog networks," *IEEE Transactions on Network Science and Engineering,* vol. 7, pp. 3266-3278, 2020.

[58] J. Jiang, Z. Li, Y. Tian, and N. Al-Nabhan, "A review of techniques and methods for IoT applications in collaborative cloud-fog environment," *Security and Communication Networks,* vol. 2020, pp. 1-15, 2020.

[59] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology,* vol. 68, pp. 5031-5044, 2019.

[60] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience,* vol. 47, pp. 1275-1296, 2017.

[61] M. Sharma, R. Kumar, and A. Jain, "A QoS-Enabled Load Balancing Approach for Cloud Computing Environment Join Minimum Loaded Queue (JMLQ)," *International Journal of Grid and High Performance Computing (IJGHPC),* vol. 14, pp. 1-19, 2022.

[62] D. Bacciu, A. Micheli, and M. Podda, "Edge-based sequential graph generation with recurrent neural networks," *Neurocomputing,* vol. 416, pp. 177-189, 2020.

[63] W. Yu, G. Hou, and J. Li, "Supply chain joint inventory management and cost optimization based on ant colony algorithm and fuzzy model," *Tehnički vjesnik,* vol. 26, pp. 1729-1737, 2019.

[64] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics,* vol. 16, pp. 6172-6181, 2019.

[65] I. Lera, C. Guerrero, and C. Juiz, "Availability-aware service placement policy in fog computing based on graph partitions," *IEEE Internet of Things Journal,* vol. 6, pp. 3641-3651, 2018.

[66] S. Mohmmad, R. Dadi, S. N. Pasha, D. Kothandaraman, Shabana, and R. Kanakam, "Cost function for energy (CFE) in Fog based IoT networks," in *AIP Conference Proceedings,* 2022, p. 020066.

[67] O. Skarlat, V. Karagiannis, T. Rausch, K. Bachmann, and S. Schulte, "A framework for optimization, service placement, and runtime operation in the fog," in *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC),* 2018, pp. 164-173.

[68] D. B. LD and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied soft computing,* vol. 13, pp. 2292-2303, 2013.

[69] K. Mishra and S. K. Majhi, "A binary Bird Swarm Optimization based load balancing algorithm for cloud computing environment," *Open Computer Science,* vol. 11, pp. 146-160, 2021.

[70] A. Jagannath, J. Jagannath, and T. Melodia, "Redefining wireless communication for 6G: Signal processing meets deep learning with deep unfolding," *IEEE Transactions on Artificial Intelligence,* vol. 2, pp. 528-536, 2021.

[71] L. Liao, Y. Lai, F. Yang, and W. Zeng, "Online computation offloading with double reinforcement learning algorithm in mobile edge computing," *Journal of Parallel and Distributed Computing,* vol. 171, pp. 28-39, 2023.

[72] T. Cui, R. Yang, C. Fang, and S. Yu, "Deep Reinforcement Learning-Based Resource Allocation for Content Distribution in IoT-Edge-Cloud Computing Environments," *Symmetry,* vol. 15, p. 217, 2023.

[73] A.-R. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, and M. AliKarar, "A smart home energy management system using IoT and big data analytics approach," *IEEE Transactions on Consumer Electronics,* vol. 63, pp. 426-434, 2017.

[74] A. Procopiou, N. Komninos, and C. Douligeris, "ForChaos: Real time application DDoS detection using forecasting and chaos theory in smart home IoT network," *Wireless Communications and Mobile Computing,* vol. 2019, 2019.

[75] X. Wang and J. Li, "Design of Intelligent Home Security Monitoring System Based on Android," in *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2018, pp. 2621-2624.

[76] C. Li, H. Zhuang, Q. Wang, and X. Zhou, "SSLB: self-similarity-based load balancing for large-scale fog computing," *Arabian Journal for Science and Engineering,* vol. 43, pp. 7487-7498, 2018.

[77] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *2015 IEEE 81st vehicular technology conference (VTC spring)*, 2015, pp. 1-6.

[78] Y. Sun, F. Lin, and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II," *Wireless Personal Communications,* vol. 102, pp. 1369-1385, 2018.

[79] P. ZHOU, B. YIN, X.-s. QIU, S.-y. GUO, and L.-m. MENG, "Service reliability oriented cloud resource scheduling method," *ACTA ELECTONICA SINICA,* vol. 47, p. 1036, 2019.

[80] Z. Chang, Z. Zhou, T. Ristaniemi, and Z. Niu, "Energy efficient optimization for computation offloading in fog computing system," in *GLOBECOM 2017-2017 IEEE Global Communications Conference,* 2017, pp. 1-6.

[81] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal,* vol. 5, pp. 283-294, 2017.

[82] X. Wang, Z. Ning, and L. Wang, "Offloading in Internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics,* vol. 14, pp. 4568-4578, 2018.

[83] L. Chunlin and L. LaYuan, "Cost and energy aware service provisioning for mobile client in cloud computing environment," *The Journal of Supercomputing,* vol. 71, pp. 1196-1223, 2015.

[84] S. Li and W. Sun, "A mechanism for resource pricing and fairness in peer-to-peer networks," *Electronic Commerce Research,* vol. 16, pp. 425-451, 2016.

[85] S. Li, Y. Zhang, and W. Sun, "Optimal resource allocation model and algorithm for elastic enterprise applications migration to the cloud," *Mathematics,* vol. 7, p. 909, 2019.

[86] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Transactions on Cloud Computing,* vol. 9, pp. 302-317, 2018.

[87] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile networks and applications,* pp. 1-8, 2018.

[88] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Transactions on Vehicular Technology,* vol. 68, pp. 1930-1941, 2019.

[89] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall),* 2018, pp. 1-6.

[90] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa*, et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971,* 2015.

[91] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning,* vol. 11, pp. 219-354, 2018.

[92] https://theaisummer.com/Reinforcement_learning/, "The secrets behind Reinforcement Learning," accessed 13-6-2023.

[93] A. O. Castaneda, "Deep reinforcement learning variants of multi-agent learning algorithms," *Edinburgh: School of Informatics, University of Edinburgh,* 2016.

[94] N. Goyal, G. Tsivgoulis, J. J. Chang, K. Malhotra, A. Pandhi, M. F. Ishfaq*, et al.*, "Admission neutrophil-to-lymphocyte ratio as a prognostic biomarker of outcomes in large vessel occlusion strokes," *Stroke,* vol. 49, pp. 1985-1987, 2018.

[95] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Advances in neural information processing systems,* vol. 33, pp. 19884-19895, 2020.

[96] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*: MIT press, 2018.

[97] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature,* vol. 588, pp. 604-609, 2020.

[98] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez*, et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815,* 2017.

[99] P. Swazinna, S. Udluft, D. Hein, and T. Runkler, "Comparing model-free and model-based algorithms for offline reinforcement learning," *IFAC-PapersOnLine,* vol. 55, pp. 19-26, 2022.

[100] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, pp. 436-444, 2015.

[101] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866,* 2017.

[102] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research,* vol. 37, pp. 421-436, 2018.

[103] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani*, et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems,* vol. 23, pp. 4909-4926, 2021.

[104] E. Choi, D. Hewlett, J. Uszkoreit, I. Polosukhin, A. Lacoste, and J. Berant, "Coarse-to-fine question answering for long documents," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 209-220.

[105] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," *arXiv preprint arXiv:1705.04304,* 2017.

[106] C. Yu, J. Liu, S. Nemati, and G. Yin, "Reinforcement learning in healthcare: A survey," *ACM Computing Surveys (CSUR),* vol. 55, pp. 1-36, 2021.

[107] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, *et al.*, "On-line building energy optimization using deep reinforcement learning," *IEEE transactions on smart grid,* vol. 10, pp. 3698-3708, 2018.

[108] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering,* vol. 139, p. 106886, 2020.

[109] S. Guha, "“Deep Deterministic Policy Gradient(DDPG): Theory and Implementation," *Towar. Data Sci*, pp. 1-10, 2020.

[110] G. Matheron, N. Perrin, and O. Sigaud, "The problem with DDPG: understanding failures in deterministic environments with sparse rewards," *arXiv preprint arXiv:1911.11679,* 2019.

[111] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274,* 2017.

[112] S. Park, D. Kwon, J. Kim, Y. K. Lee, and S. Cho, "Adaptive real-time offloading decision-making for mobile edges: deep reinforcement learning framework and simulation results," *Applied Sciences,* vol. 10, p. 1663, 2020.

[113] M. Kadhum, S. Manaseer, and A. L. A. Dalhoum, "Cloud-edge network data processing based on user requirements using modify mapreduce algorithm and machine learning techniques," *International Journal of Advanced Computer Science and Applications,* vol. 10, 2019.

[114] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning* vol. 135: MIT press Cambridge, 1998.

RESUME

Fatimah Najeh Abulateef Al-ZUABIDI, her primary and elementary education in Iraq. She completed her undergraduate studies at Dijlah University College in 2010-2011 Baghdad – Iraq. Then she started her master’s degree in Department of Computer Engineering at Karabuk University.