زانکۆی پۆلیتەکنیکی هەولێر
ERBIL POLYTECHNIC UNIVERSITY

A new Scheduling Scheme in Fog Computing system using Deep Reinforcement Learning Algorithm

A Doctoral Dissertation

Submitted to the council of the college of Erbil Technical Engineering at Erbil Polytechnic University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information Systems Engineering

By

Media Ali Ibrahim

B.Sc. Software Engineering 2007

M.Sc. Advanced Software Engineering 2013

Supervised By

Prof. Dr. Shavan Kamal Askar

Erbil Kurdistan
February 2024

DECLARATION

I declare that the Ph.D. Dissertation entitled:

“**A new Scheduling Scheme in Fog Computing system using Deep Reinforcement Learning Algorithm”** is my original work, and hereby I certify that unless stated, all study contained within this dissertation is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgment is made in the text.

Signature:

Student Name: Media Ali Ibrahim

Date: 18/2/2024

LINGUISTIC REVIEW

I confirm that I have reviewed the dissertation titled "**A new Scheduling Scheme in Fog Computing system using Deep Reinforcement Learning Algorithm**" written by the postgraduate student (**Media Ali Ibrahim**) has been proofread and checked for grammatical, punctuation, and spelling mistakes. Therefore, after making all the required corrections by the student for further improvement, I confirm that this last copy of the dissertation is ready for submission.

Signature: [illegible]

Name of Reviewer: HONAR OSMAN KHIDR

E-mail: honar.hawler@gmail.com

Phone No.: 0750 8989088

Date: 8-01-2024

SUPERVISOR CERTIFICATE

This dissertation has been written under my supervision and has been submitted for the award of the degree of Doctor of Philosophy in Information Systems Engineering with my approval as supervisor.

Signature

Name: Prof. Dr. Shavan Kamal Askar

Date:

I confirm that all requirements have been fulfilled.

Signature:

Name: Mr. Bayad Ahmed

Head of the Department of Information Systems Engineering

Date:

I confirm that all requirements have been fulfilled.

Postgraduate Office

Signature:

Name:

Date:

EXAMINING COMMITTEE CERTIFICATION

We certify that we have read this Dissertation: “A new Scheduling Scheme in Fog Computing system using Deep Reinforcement Learning Algorithm” and as an examining committee, examined the student (Media Ali Ibrahim) in its content and what related to it. We approve that it meets the standards of a dissertation for the degree of Doctor of Philosophy in Information Systems Engineering.

Signature:	Signature:
Name: Assist. Prof. Dr. Bzar Khidir Hussan	Name: Assist. Prof. Dr. Reben MS KURDA
Member	Member
Date:	Date:
Signature:	Signature:
Name: Assist. Prof. Dr.Marwan Aziz	Name: Assist. Prof. Dr.Moayad Yousif Potrus
Member	Member
Date:	Date:
Signature:	Signature:
Name: Professor Dr. Shavan Kamal Askar	Name: Professor Dr. Subhi R. M. Zeebaree
Supervisor	Chairman
Date:	Date:

Signature

Name: Professor Dr. Ayad Zaki Saber

Dean of the College of Erbil Technical Engineering

Date:

ACKNOWLEDGEMENTS

First of all, I would like to thank **ALLAH** almighty, the most merciful and compassionate, for His support, help, and generosity.
I am deeply indebted to my supervisor’s Prof. Dr. Shavan Askar. I would like to express my sincere gratitude to him for his continuous support, his patience, motivation, and immense knowledge during my Ph.D. study and writing this dissertation.

Lastly, I would like to express my heartfelt thanks to my family for their unwavering support during the study and the process of writing my dissertation. Their encouragement and understanding have been invaluable throughout this---- journey.

ABSTRACT

Fog Computing (FC) has recently emerged as a promising new paradigm that provides resource-intensive Internet of Things (IoT) applications with low-latency services at the network edge. However, the limited capacity of computing resources in Fog colonies poses great challenges for scheduling and allocating application tasks. In this dissertation, an Intelligent Scheduling Strategy Algorithm in a Fog Computing system based on Multi-Objective Deep Reinforcement Learning (MODRL) is proposed. MODRL algorithm select nodes (Fog nodes or Cloud nodes) for task processing based on three objectives; current node's Load, node Distance, and task Priority. MODRL is a smart method that integrates the ideas of Multi-Objective Optimization and Deep Reinforcement Learning to tackle intricate decision-making situations involving several conflicting objectives. This technique is especially valuable in situations when there is a requirement to maximize numerous criteria simultaneously, even if they do not exactly line, and where trade-offs need to be taken into account. The proposed model addresses two main problems; task allocation and task scheduling. Employ three Deep Reinforcement Learning (DRL) agents based on a Deep Q Network (DQN), one for each objective. It is a specific form of Artificial Neural Network structure employed in Reinforcement Learning. The DQN algorithm utilizes a Deep Neural Network, commonly a Convolutional Neural Network (CNN), to estimate the Q-function. This enables the model to effectively process intricate input domains. However, this is a more challenging scenario because there is a trade-off among these objectives, and eventually, each algorithm may select different processing nodes according to its own objective, which brings to a Pareto Front problem. To solve this problem, propose using Multi-Objective Optimization, a Non-dominated Sorting Genetic Algorithm (NSGA2), and a Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D), which are Multi-

Objective Optimization algorithms that can choose the optimal node by considering three objectives.

Simulation investigation and experiments using a Python environment with TensorFlow, PyTorch, Pymoo, and PQDM libraries in PyCharm, which is a powerful Python IDE, to simulate and train the Intelligent Scheduling Strategy. As well as, Virtualized data using MatPlotLib in the Jupyter Notebook, indicates that the proposed Intelligent Scheduling Strategy could attain better results for the several employed efficiency, adaptability, and performance metrics: Task Completion Time, Makespan, Transmission Delay, Queueing Delay, Processing Delay, Propagation Delay, Computational Delay, Latency, Network Congestion, Throughput, CPU Load, and Storage Utilization, with an average value of 2.02ms, 10ms, 25ms, 2ms, 1.0ms, 9.5ms,3ms, 3.5ms, 0.10ms, %100, %10, and % 99, respectively.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

4G	Fourth Generation
5G	Fifth Generation
A2C	Advantage Actor-Critic
AI	Artificial Intelligent
ANN	Artificial Neural Network
CC	Cloud Computing
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DDPG	Deterministic Policy Gradients
DDQN	Double Deep Q Network
DISA	Deep Intelligent Scheduling Algorithm
DNN	Deep neural Network
DQBRA	Deep-learning-Q-Network based resource-allocation
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
DRLIS	Deep Reinforcement Learning-based IoT application Scheduling
EA	Evolutionary Algorithm
EATS	Efficient Algorithm Task Scheduling
EC	Edge Computing
EMO	Evolutionary Multi-Objective
EO	Evolutionary Optimization
FC	Fog Computing
FN	Fog Node
GPU	Graphical Processing Unit
IDE	Integrated Development Environment
IoT	Internet of Things
MatplotLib	Matlab Plotting Library
ML	Machine Learning

MODRL	Multi Objective Deep Reinforcement Learning
MOEA/D	Multi Objective Evolutionary Algorithm /Decomposition
MOMDPs	Multi Objective Markov Decision Process
NN	Neural Network
NSGA2	Non-Dominated Sorting Genetic Algorithm
PAES	Pareto Archived Evoalutionary Strength
POF	Pareto Optimal Front
PPO	Proximal Policy Optimization
PQDM	Power Quality Data Manager
PSO	Particle Swarm Optimization
PyCharm	PythonCharm
Pymoo	Python Multi-Objective Optimization
PyTorch	Python Torch
PF	Pareto Front
QMTSF	Q-learning-based Multi-Task Scheduling Framework
QoS	Quality of Service
ReLU	Rectified Linear Unit
RL-G	Reinforcement Learning - Greedy
RR	Round-Robin
SDN	Software Defined Network
SLAs	Service Level agreements
SPEA	Strength Pareto Evolutionary Algorithm
UCB	Upper Confidence Bound
UQRL	UCB-based Q-Reinforcement Learning
VM	Virtual Machine
WIFI	Wireless Fidelity
WSGA	Weighted Sum Genetic Algorithm

CHAPTER ONE
INTRODUCTION

1.1 Overview

In recent years, due to the growing progress in the use of the Internet of Things (IoT), there has been a significant rise in both the number of applications and the amount of data being demanded. In addition, the demand for real-time data processing and analysis is increasing. However, traditional cloud computing faces several threats, such as latency, performance, network breakdown, and security.

Moreover, the traditional Cloud Computing architecture is not compatible with IoT applications because of its inherent limitations, such as limited bandwidth, high latency, and high-power consumption (Alizadeh *et al.,* 2020a)-(Roheed Khaliqyar et al., 2023)-(Sabireen et al., 2021). With the discovery of Fog Computing (FC), these problems have been addressed by computing clouds nearer to IoT. FC provides storage and computation such that all services can be transferred over the network between the IoT and cloud layers. Furthermore, FC can provide full authentication using local computers and share secure information locally or through distributed computing (Laghari et al., 2021)-(Rahman et al., 2021).

Researchers from Cisco Systems proposed FC in 2012. FC acts as a link between the IoT layer, storage devices, and cloud computing (CC). FC in a distributed environment, in which several heterogeneous Fog nodes (FN) can share and connect their computing and storage resources among neighboring nodes for further analytics and processing (Sabireen et al., 2021)-(Alizadeh *et al.*, 2020a)-(Gazori, Rahbari et al., 2020a). Therefore, FC is not a replacement for CC but extends the computation, storage, and communication facilities from the cloud to the edge of the networks(Mukherjee et al.2018). The main objectives of FC are to reduce network traffic, latency, power consumption, and operational costs(Ghobaei-Arani et al., 2020a). FC is a novel approach that aims to bring the Cloud layer closer to the IoT user as well as enhance the Quality of Service (QoS)(Rani et al., 2022). QoS includes optimizing the response time until real-time applications can rapidly make their final decision at the same time as the actions(Abdel-Basset *et al.*, 2021a). Compared with CC, FC delivers services with a quick response time and high quality. Hence, FC might be the best option to allow IoT to deliver highly and efficiently secure services to several IoT clients. It is eventually at places specified by Service Level Agreements (SLAs) or permits the administration of resource provisioning and services outside Cloud Computing, nearer to nodes, at the network edge network layer (Sabireen et al., 2021)

On the other hand, compared to CC, FC does not have sufficient storage and computing resources. Therefore, efficient resource allocation is a significant research problem for FC(Mukherjee et al., 2018). Hence, one of the key challenges in a FC environment for running IoT applications is resource allocation(Naha *et al.*, 2018)(Alizadeh *et al.*, 2020a)(Ghobaei-Arani et al., 2020b)(Islam et al., 2021). Moreover, executing tasks in the FC layer for IoT applications requires efficient resource management and allocation and physical servers in cloud data centers to satisfy QoS requirements. However, achieving this objective faces many serious challenges due to the limitations and complex heterogeneity of Fog resources, the dynamic nature of resource demands, and locality restrictions (Tran-Dang *et al.*, 2022). To improve the operation of FC and further achieve its objectives, a practical and exact scheduling approach is required(Alizadeh *et al.*, 2020a). Additionally, the FC manages services and resources in a decentralized manner. Fog devices provide services to IoT users in a decentralized manner(Niranjan et al., 2018). In general, IoT nodes are connected to a FN in a FC environment. These FNs are responsible for intermediate storage and computation and are located in close proximity to IoT users(Naha *et al.*, 2018). In addition, IoT nodes do not have sufficient resources to analyze or store the generated data, and some of the connected nodes have no intelligence to process the analyzed data in order to make decisions. Thus, they require an external controller to schedule tasks and make decisions(Gazori et al., 2020a). Typically, task scheduling includes assigning which resources will process which tasks. In large-scale systems, including the Cloud layer IoT layer and Fog layer, the possible resources for computation execution contain servers in the Cloud-tier IoT nodes and Fog nodes(Tran-Dang *et al.*, 2022). In addition, scheduling aims to reduce response time, enhance resource utilization, increase user satisfaction, and improve performance(Alizadeh *et al.*, 2020b). Thus, efficient resource management will improve Fog Computing performance, and task scheduling is an essential requirement for performance optimization in Fog Computing environments(Gazori et al., 2020a).

Nowadays, Artificial Intelligence (AI) has become a new technology in the area of information and knowledge technology for the control of heterogeneous and homogeneous nodes connected in FC, as well as the management of data(Hazra *et al.*, 2023). Machine Learning (ML), specifically Deep Reinforcement Learning (DRL), is considered an effective technique that has appealed to the research community (Nassar et al., 2018)to tackle numerous resource management problems. DRL has a strong ability to deal with decision-making problems (Gazori et al., 2020a)so that agents can respond efficiently to the dynamics of the environment. This vision suggests great potential for the application of RL in the concept of FC concerning resource allocation for task execution and offloading to attain improved performance. In addition, RL has been progressively applied and studied to successfully solve resource allocation problems in Fog Computing environments(Tran-Dang *et al.*, 2022). By exploiting a deep neural network (DNN), DRLs can provide accurate regression and estimate precise value functions for RL problems(Nassar et al., 2018). DRL- and deep Q-learning (DQL) based schemes were combined to optimize resource allocation(Liu *et al.*, 2019). In the Reinforcement Learning-based model, Q-Learning with the epsilon-greedy algorithm is applied to derive the best action selection (Tran-Dang *et al.*, 2022)For Multi-Objective which results in a Pareto Front problem that is solved by an optimization algorithm.

Optimization plays an important role in daily life. In computing, optimization refers to application performance with minimum resources or a maximizing system. In optimization, meta-heuristic techniques are more effective in solving realistic problems in several fields, such as computing and engineering, and population-based techniques transform and manipulate a set of solutions through the optimization procedure. These can be categorized into Single-Objective and Multi-Objective techniques. A Multi-Objective method was used to optimize two or more competing goals by simultaneously considering the constraints at the same time(Sharma et al., 2022). The performance of three Multi-Objective Evolutionary Algorithms, namely Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D), Non-Dominated Sorting Genetic Algorithm 2 (NSGA 2), and Weighted Sum Genetic Algorithm (WSGA) to find the best solution for task scheduling in FC to minimize network delay, allocate services, and use resources effectively. The test results show that both MOEA/D and NSGA-II can effectively optimize the objectives compared to WSGA, whereas MOEA/D can minimize the execution time the most (Abdel-Basset *et al.*, 2021b)(Guerrero et al., 2019).

In this study, A scheduling strategy for an FC system utilizing a MODRL algorithm has been proposed. The proposed model tackles two primary issues: task allocation and task scheduling. The DRL algorithm is necessary to formulate an optimal approach for assigning tasks generated by IoT devices to appropriate processing nodes (Fog or Cloud), and subsequently arranging those tasks in the designated Fog nodes according to various criteria. The experimental findings demonstrate that the proposed (DQN+ Multi-Objective Optimization) algorithm is superior and adaptable in comparison to the existing relevant research studies.

1-2 Problem Statements

The investigated problems in this dissertation can be summarized as follows:

1- In Fog Computing environments, efficient resource management for running IoT applications is a critical challenge, often approached through single-objective scheduling, neglecting concurrent optimization of multiple objectives (Load, Distance , and Priority).

2- Existing research exhibits shortcomings such as inadequate consideration of task scheduling and task allocation, insufficient utilization of proper Orchestrator in the Fog layer for management, and a lack of Multi-Objective agents for accurate provisioning of Quality-of-Service requirements.

3- Addressing these gaps necessitates exploring Multi-Objective Optimization schemes, conducting studies that integrate performance, efficiency, and adaptability metrics, and developing approaches that concurrently optimize task scheduling, task allocation, and Quality-of-Service provisioning in Fog Computing environments.

1-3 Study of the Questions:

This dissertation aims at addressing the problems specified in the statement of the problem section. During the study made in this work, some critical research questions have been raised. The following are the research questions of this dissertation:

1- How can task scheduling and task allocation be optimized in Fog Computing environments to enable computing resource utilization and efficient storage for IoT applications?
2- What scheme can satisfy the target of handling tasks scheduling as well as task allocation and at the same time it can be optimized for Multi-Objectives?
3- How to fit the proposed adaptive resource management techniques in the Fog layer to dynamically alter task scheduling and task allocation based on resource availability and changing workloads within the Fog Computing management layer?
4- How can the proposed work achieve a balanced trade-off between response time and task number in the Fog Computing system?

1-4 Aim and objectives

1.4.1 Aim:

Specifically in the context of running IoT applications, the aim of this study is to solve the issue of resource management in Fog computing settings. By simultaneously taking into account numerous goals, the study aims to enhance task scheduling, thereby boosting the Fog Computing system's overall performance, efficiency, and adaptability.

1.4.2 Objectives:

1- Adaptive Resource Management: Create adaptive system that dynamically conduct task scheduling and task allocation in response to resource availability and changing workloads.

2- Multi-Objective Task Scheduling: Implement and design a task scheduling algorithm that simultaneously optimizes a variety of objectives. For example, maximize resource utilization and minimize response time.

3- Optimize task Allocation: Develop task allocation strategies that effectively compute resources and distribute storage in Fog Computing environments while taking into account the unique requirements and constraints of IoT applications.

4- Performance Enhancement: Improve the performance of intelligent scheduling in Fog Computing environments, with an emphasis on better reliability, increased throughput, and reduced latency.

1-5 Scope of the Study

1- Fog Computing Environment: The study pays particular attention to Fog Computing environments, which act as a middle layer between IoT nodes and cloud computing.

2- Resource Management: The main emphasis of the study is on resource management, including computing resources and storage, within the FC environment.

3- Multi-Objective Deep Reinforcement Learning: The study addresses the challenge of Multi-Objective optimization for task allocation and task scheduling. It explores methods to concurrently optimize multiple objectives, including Load, Distance, and Priority.

1-6 Contributions and Proposed System of the Study

In this study, an Intelligent Scheduling Strategy for a FC system based on a DRL algorithm has been proposed. The proposed model addresses two main problems: resource allocation and task scheduling. The DQN algorithm of the DRL is required to develop the best strategy to allocate the tasks generated from IoT devices to suitable processing nodes (Fog or Cloud), as well as to schedule those tasks in the allocated Fog nodes based on multiple criteria. Finally, this research study provides the following contributions:

1- Deployed a Central Smart Fog Orchestrator that manages the Fog system. The Fog nodes host IoT applications to process tasks received from the IoT layer.

2- Proposes the use the MODRL based on a DQN to select the nodes for task processing (Fog nodes or Cloud) based on three objectives: Load, Distance, and Priority.

3- Using a Multi-Objective Optimization Algorithm (MOEA/D and NSGA2). It is a metaheuristic optimization method employed for solving Multi-Objective optimization problems that can choose the optimal node by considering three objectives. To validate the performance, efficiency, and adaptability of the proposed (DQN+Multi-Objective Optimization) algorithm, the Task Completion Time, Makespan, Processing Delay, Propagation Delay, queueingDelay, Transmission Delay, Computational Delay, Latency, CPU Load, and Storage Utilization metrics have been used. The experiment results prove the high performance, effectiveness, and adaptive of the proposed algorithm across all three objectives (Load, Distance, and Priority) compared with the existing related research studies.

1-7 Structure of the Dissertation

This dissertation is divided into *five chapters* and organized as follows:

Chapter 1 includes an overview, problem statements, questions, aims, objectives, scope of the study, contributions, and proposed system of the dissertation.

Chapter 2 presents the literature review and the theoretical background of resource management based on reinforcement learning. Based on the literature, resource management can be categorized as task scheduling and task allocation for a single objective.

Chapter 3 dedicated to task scheduling and task allocation based on Multi-Objective Deep Reinforcement Learning as well as include the proposed system architecture, system model, system environment, and proposed system algorithm.

Chapter 4 illustrates the results of the experimental study as well as compares the proposed algorithm with the current related research studies.

Finally, *chapter 5* presents the conclusion of the study. In addition, future work is given at the end of the conclusion section.

CHAPTER TWO

BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

Scheduling in Fog Computing, especially when utilizing Deep Reinforcement Learning (DRL) methods, is a complicated and dynamic area of research. Dynamic Resource Allocation (DRL) approaches have the capability to enable intelligent decision-making in Fog environments that have limited resources. Low delay and high resource utilization are crucial measures for evaluating the effectiveness of scheduling algorithms in this context. Attaining a minimal delay guarantees prompt and responsive execution of tasks, while optimizing resource usage improves the capacity and overall efficiency of FN. The investigation of advanced scheduling algorithms supported by DRL is crucial for improving the adaptability and performance of FC system. Moreover, scheduling in Fog Computing is crucial for maximizing resource utilization, distributing workloads evenly, minimizing delays, promoting energy efficiency, controlling QoS, responding to changing environments, improving fault tolerance, and prioritizing jobs. Efficient scheduling enhances the overall efficiency, dependability, and promptness of FC systems.

This chapter addresses the background of scheduling in FC based MODRL. It also mentions previous and related research works. A summary table of the mentioned research, the architecture and methods used, the objective of each one, the tool simulated, and the limitations will be shown at the end of this chapter. This table will assist in proposing an intelligent scheduling strategy that will fulfill the shortages of these modified or proposed networks.

2.1.1 Fog /Cloud Computing with IoT

With the Internet of Things, billions of physical things can be connected to each other and share data for a wide range of uses. On the other hand, some IoT applications may find it problematic when unsupported capabilities like geographic distribution, location awareness, and low latency are included in IoT. On the other hand, a number of risks, including latency, performance, network outages, and security, affect traditional cloud computing. Furthermore, due to its intrinsic restrictions, like constrained bandwidth, high latency, and high-power consumption, the traditional Cloud Computing architecture is incompatible with Internet of Things applications. FC is coupled with IoT to expand networking capabilities, storage, and compute to the Edge layer in order to provide these functions (Ni *et al.*, 2018).

FC brings the cloud closer to IoT nodes, as shown in figure 2.1 (Dizdarevic *et al.*, 2018). FC, a concept presented by Cisco, is the extension of Cloud Computing capabilities to the network's edge, closer to IoT devices and sensors. It offers several benefits, including low latency, bandwidth efficiency, privacy and security, scalability, reliability, edge intelligence, and adaptability. By processing data closer to the source, Fog Computing reduces latency, optimizes bandwidth usage, and minimizes the need for sensitive data to be sent to the Cloud. It also enhances reliability by ensuring that local processing at the edge continues even in the event of a centralized cloud service downtime. Fog Computing also facilitates edge intelligence, enabling devices to make complex decisions locally without relying heavily on cloud services. Rather than transferring IoT data to the Cloud layer, the FN enable local IoT data processing and storage at IoT nodes. The FN offers higher-quality services with faster reaction times than the cloud. Therefore, FC may be regarded as the greatest option to enable the IoT to offer efficient and safe services for a variety of IoT nodes. On the other hand, FN are regularly network nodes equipped with additional storage and computing power. Though it is problematic for such nodes to match the resource capacity of Cloud Computing servers, Consequently, sensible management of Fog resources is essential for the effective operation of the FC system (Atlam et al., 2018). Typically, task scheduling includes assigning which resources will process which tasks. In large-scale systems, including Cloud-layer IoT layers and Fog layers, the possible resources for computation execution contain servers in the cloud-tier IoT nodes and Fog nodes (Tan et al., 2017).

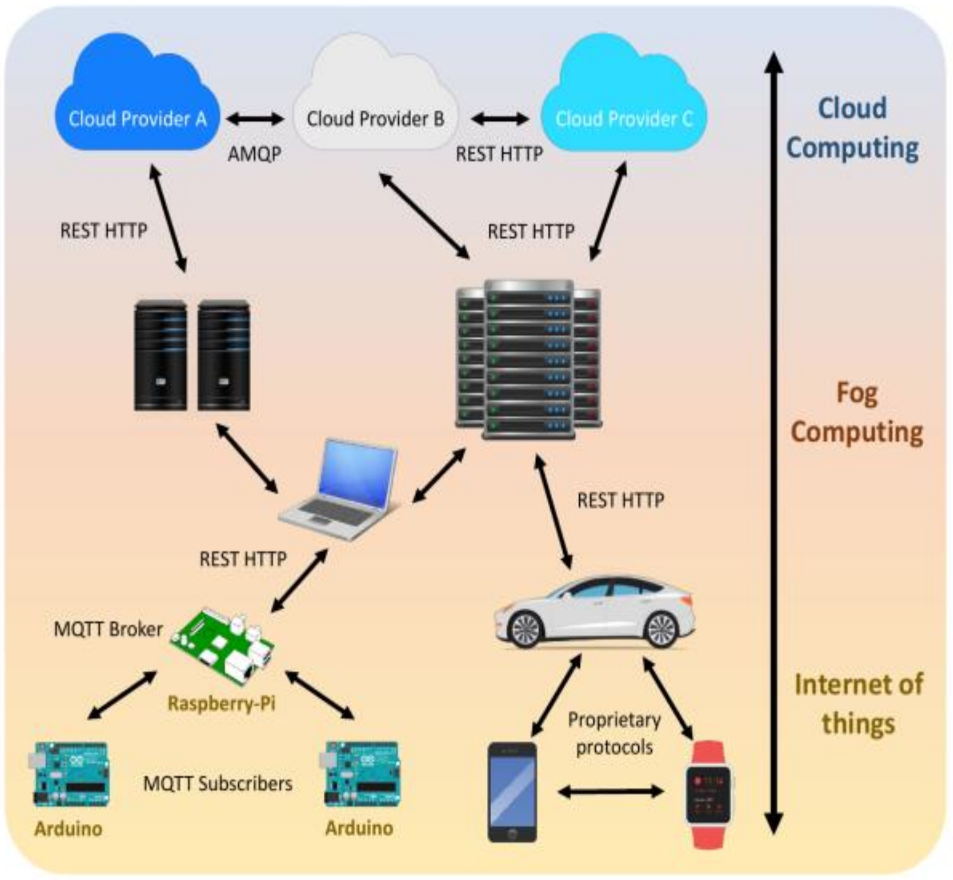


Figure 2. 1: *Fog Computing is an extension of cloud but closer to IoT*

2.1.2 Resource Management in Fog Computing

As FC is still a developing part of research, there is insufficient research focusing on resource management. FC is essential to operate separately to ensure uninterrupted services even once there are variable connections with the cloud layer. It is also mandatory to fully integrate with the Cloud layer when complete resource connectivity is restored. So, there is a requirement for the development of effective orchestration mechanisms and resource management to confirm acceptable performance of services and applications while taking advantage of Cloud skills (Dlamini et al., 2019).

FC includes the utilization, efficient allocation, and monitoring of computing resources in a distributed environment. It involves resource discovery, allocation, Load balancing, dynamic scaling, QoS management, energy efficiency, fault tolerance, security and privacy, monitoring and analytics, task scheduling, adaptability to dynamic environments, and elasticity. Resource discovery involves determining available devices or Fog nodes' capabilities, allocation based on task requirements, and load balancing to prevent bottlenecks. Dynamic scaling allows the Fog Computing infrastructure to scale up or down in response to changing workloads. QoS management prioritizes meeting QoS requirements for user satisfaction. Security measures protect computing resources and data, and monitoring and analytics help identify trends and make informed decisions. Task scheduling intelligently optimizes resource use, and adaptability to dynamic environments allows for dynamic resource allocation. Thus, the aim of resource management in FC is to select FNs that take the form of algorithms and best process IoT data and are implemented within specific Fog layer controllers or Fog nodes. As well as, resource management relies on additional software and hardware structures, such as the Application Programming Interface (API), or controllers, to be implemented within the Fog system for suitable FN selection. Furthermore, resource management is the key aspect that determines the performance of FC (Fahimullah et al., 2022).

Currently, a central Reinforcement Learning agent is used in Fog layer resource management techniques. Distributed multi-agent Reinforcement Learning has been applied to huge systems, where each agent manages a portion of the total FNs, hence improving overall system resource management (Martinez et al., 2020).

2.1.2.1 Scheduling in Fog Computing:

Task allocation and task scheduling are two closely connected ideas in FC, but they refer to separate aspects of handling computational tasks within a FC system. Scheduling plays a significant role in Fog resource management; task scheduling is the facility to plan tasks for the suitable resources in FC. As FC contains distributed and heterogeneous resources, task scheduling becomes complex, which is the main challenge in Fog computing. Scheduling in Fog computing offers numerous advantages, such as optimizing resources, balancing workloads, reducing latency, improving energy efficiency, managing QoS, adapting to changing environments, ensuring fault tolerance, prioritizing tasks, enabling scalability, optimizing costs, and enhancing user experience. The combined benefits of these characteristics help to the efficient functioning of Fog Computing systems in diverse applications and sectors. The scheduling problems have also been categorized into five groups: task allocation, task scheduling, workflow scheduling, job scheduling, and resource scheduling (Matrouk et al., 2021)

In a distributed computing environment, task scheduling refers to the procedure of determining when and where to execute an exact task. As well as, the main challenge of task scheduling in FC is to satisfy IoT users' dynamic requirements in real-time with FNs incomplete resource capacities. So, it is not conceivable to schedule the complete task on one node; it is distributed among multiple Fog colonies and separated into sub-tasks. These FNs are distributed in a colony and are typically achieved by a Fog scheduler. It plays an important role in processing the Priority tasks locally within a colony to moderate service delays (Kaur et al., 2021).

Scheduling is a crucial aspect of FC, aiming to efficiently allocate and manage computing resources for tasks or applications in a distributed and dynamic environment as shown in figure 2.2. It involves making decisions about when and where to execute tasks, considering factors such as resource availability, task priorities, latency requirements, and energy efficiency. Scheduling algorithms optimize resource utilization, distribute tasks across available resources, reduce latency, optimize energy consumption, and manage QoS requirements. They also adapt to dynamic environments, ensuring efficiency in a shifting environment. Overall, scheduling is essential for optimizing resource usage, balancing loads, reducing latency, improving energy efficiency, managing QoS, adapting to dynamic environments, enhancing fault tolerance, prioritizing tasks, and supporting elasticity. Therefore, scheduling plays a vital role in FC to professionally manage these resources and assign them to each task simultaneously. On the other hand, task allocation includes deciding which exact resources should be allocated to execute a specific task. Thus, task allocation is one of the key challenges in running IoT applications in Fog Computing. FC needs to manage job distribution in highly efficient ways to enhance the QoS of these latency-sensitive applications (Alsmirat et al., 2020).

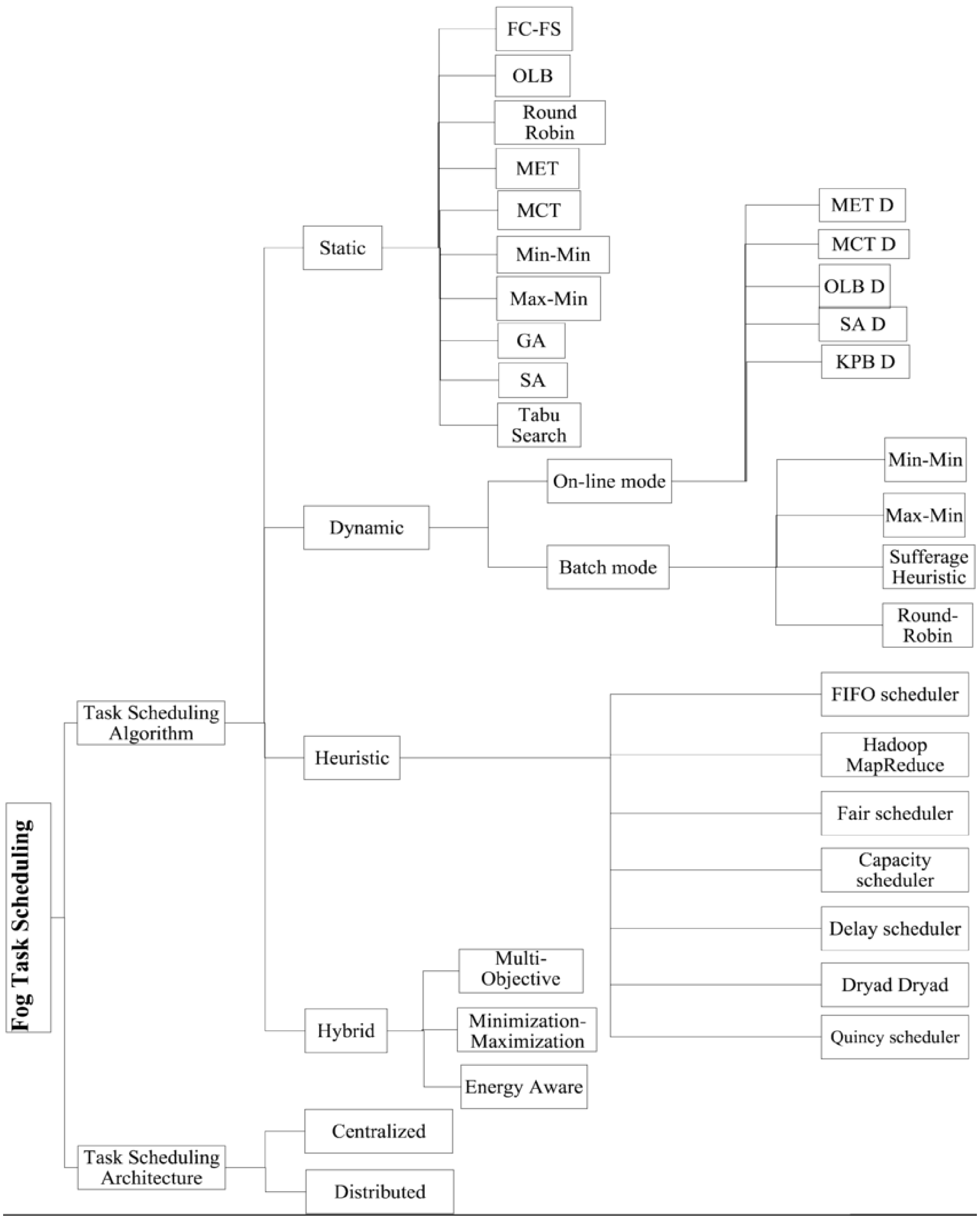


Figure 2. 2: *The taxonomy of task scheduling in a Fog computing System*

2.1.3 Deep Reinforcement learning

In recent years, remarkable advancements have been made in solving challenging problems across numerous fields using DRL. DL and RL, two important sub fields of ML, have made important strides in both the progress of the study of practical applications and theoretical frameworks, allowing high-dimensional, interactive learning. Reinforcement Learning is the process of teaching an agent to make a series of decisions in an environment by interacting with it and receiving feedback in the form of rewards or punishments. DL, particularly deep neural networks, is utilized to process input spaces with a large number of dimensions and intricate mappings. In addition, Deep Reinforcement Learning has demonstrated exceptional efficacy in resolving intricate issues and attaining performance beyond human capabilities in several fields. The capability to process input spaces with a large number of dimensions and acquire hierarchical representations renders it an influential method for numerous practical applications. Nevertheless, it necessitates meticulous examination of the obstacles and factors associated with instruction and implementation. For example, resource management (scheduling), automatic driving, and game strategy design (Henderson et al., 2018)(Wang *et al.*, 2023)

Deep RL = RL Algorithm + Artificial Neural Network (ANN)

A main reason for interest in DRL is that it functions well on current computers and appears to have various applications. The aim of DRL is to Find the optimal course of action that maximizes reward across all possible states of the environment (the Fog layer) is the goal of DRL. The system tries out actions, learns from the feedback, and interacts with high-dimensional and complicated settings to do this. Deep Neural Networks (DNN) are used by DL to estimate complex and high-dimensional environments. issues whose complexity prevents tabular approaches from offering precise solutions. The DL has advanced significantly; machines can now recognize pedestrians in a series of photos and can understand sentences. Moreover, the field of RL involves by trial and error. And it learns from feedback; RL doesn't rely on pre-exist datasets for training; it independently selects actions and learns by receiving feedback from the environment (Plaat, 2022)(Tan, Yan and Guan, 2017b)

(Mao *et al.*, 2016)believes RL methods are particularly well-suited to resource management systems. First, the decisions made by these systems frequently follow repetitive patterns, resulting in a wealth of training data that proves advantageous for RL algorithms. This is exemplified in scenarios like cluster scheduling decisions and the consequent performance outcomes. Second, RL can model complex systems and decision-making policies as DNN equivalent to the models used for game-playing agents. In addition, RM problems are ubiquitous in networks and computer systems. Examples include scheduling in compute colonies, virtual machine placement in CC, and relay selection in Internet telephony.

Various profound Reinforcement Learning (DRL) methods have been created to tackle distinct aspects of instructing agents to make sequential judgments in settings. DRL algorithms fall into two primary categories: Both model-based and model-free algorithms, as shown in figure 2.3. These methods are merely a fraction of the wide array of techniques in the field of DRL. The selection of an algorithm is frequently influenced by the specific attributes of the problem being addressed, such as the characteristics of the environment, the type of behaviors involved, and the desired trade-off between exploration and exploitation. Task scheduling is defined as deciding which tasks are processed by the IoT layer, the Fog layer, or the Cloud layer in order to achieve the goal design purposes of minimizing the computation cost and long-term service delay for the FC environments under the scheduler in the

IoT-Fog or IoT-Cloud systems. So, a DQL-based scheduling algorithm is introduced (Tran-Dang *et al*. 2022).

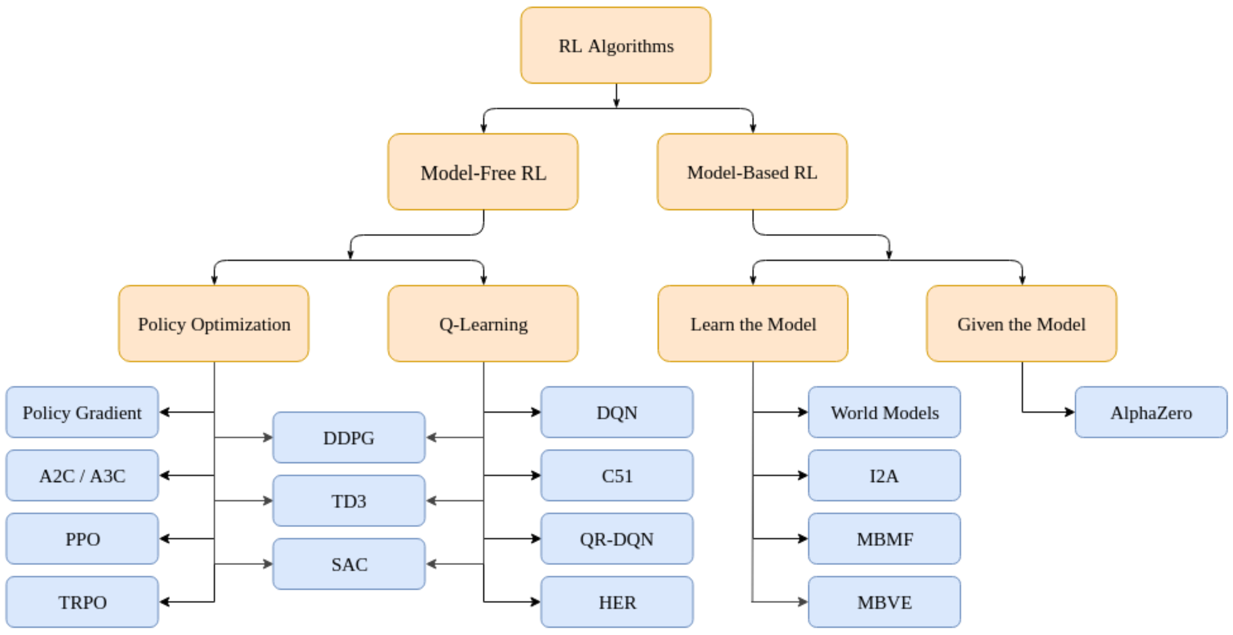


Figure 2. 3: *A Taxonomy of RL algorithms*

2.1.3.1 Model-free Algorithms

Model-free algorithms nature make up the first main category of DRL algorithms, as well as establish the epitome of a straight learning procedure over experience. More precisely, an agent in an environment tries to learn the best policy for solving a task by straight altering the experience collected as a consequence of achieved actions, into a consequential policy. Model-free RL algorithms are a type of approaches that work without having explicit knowledge of the dynamics of the environment they are operating in. These techniques acquire optimal tactics by directly interacting with the environment, depending on trial and error instead of a pre-established model (Tran-Dang *et al*. 2022).

Value-based algorithms, such as Q-Learning methods, use the value function to inform decision-making, whereas policy-based techniques, like REINFORCE and Proximal Policy Optimization (PPO), explicitly define the agent's strategy through parameterize. Actor-Critic approaches, such as Deep Deterministic Policy Gradients (DDPG) and Advantage Actor-Critic (A2C), integrate elements from both value and policy-based methods. They involve an actor for decision-making and a critic for evaluating actions. Model-free Reinforcement Learning is especially advantageous in situations when the environment is intricate or its dynamics are uncertain. It provides flexibility in a wide range of applications where learning is achieved through direct interactions, without the need for explicit models. Some examples of these algorithms are Q-Learning, REINFORCE, A2C, and DDPG. Deep QL is one of the main categories of model-free family as described in below section (Lazaridis, 2020).

2.1.3.1.1 Deep Q-Network

Q-Learning is arguably one of the off-policy strategies and one of the most applied representative RL approaches. Currently, due to the overall advancements in Reinforcement Learning (RL), there has been a widespread exploration and implementation of several adaptations of Q-Learning. One such modification is Deep Q-Learning, which integrates traditional Q-Learning with Deep Neural Networks (DNN). In addition, one of the most popular algorithms is Deep Q-Learning, developed in 2016 at Google. Furthermore, Deep Q-Learning adds two techniques to the value estimation via an ANN. The target Q technique is one, and an experience replay is the other. The Q technique's value approximation via Neural Network (NN) is highly imbalanced; this is stabilized by the experience replay. Every action, every state, and every reward in the experience replay technique are valued based on past states. Thus, there are relationships among states, incentives, and actions. These relationships mean that learning the estimate function in a steady-state manner is not possible. Furthermore, the experience replay removes connections by buffering the experience and extracting the learning data at random(Jang *et al.*, 2019).

Moreover, A Deep Q-Network (DQN) is an approach for Reinforcement Learning that integrates the ideas of Q-Learning with DNN. This permits for the acquisition of optimum strategies in environments characterized by state spaces with a large number of dimensions. DeepMind introduced DQN, which use Deep Neural Networks to estimate Q-values, enabling it to effectively deal with intricate, practical scenarios. The approach utilizes experience replay to improve the stability of training by storing and randomly selecting previous experiences, and target networks to produce more consistent estimations of Q-values during updates. DQN rose to attention because of its remarkable performance in playing Atari 2600 video games, demonstrating its capacity to generalize across diverse settings. The incorporation of ε-greedy exploration in DQN has established it as a fundamental algorithm in the realm of Deep Reinforcement Learning, exerting significant influence on later advancements and practical implementations. DQN is an adaptation of the traditional Q-Learning method that offers three key improvements: (1) estimating the Q-values of the upcoming state using previous network parameters (2) utilizing mini-batches of arbitrary training data as an alternative to updating the most recent experience in steps; and (3) a deep CNN architecture for Q-function approximation, as shown in the figure 2.4 (Roderick et al., 2017).

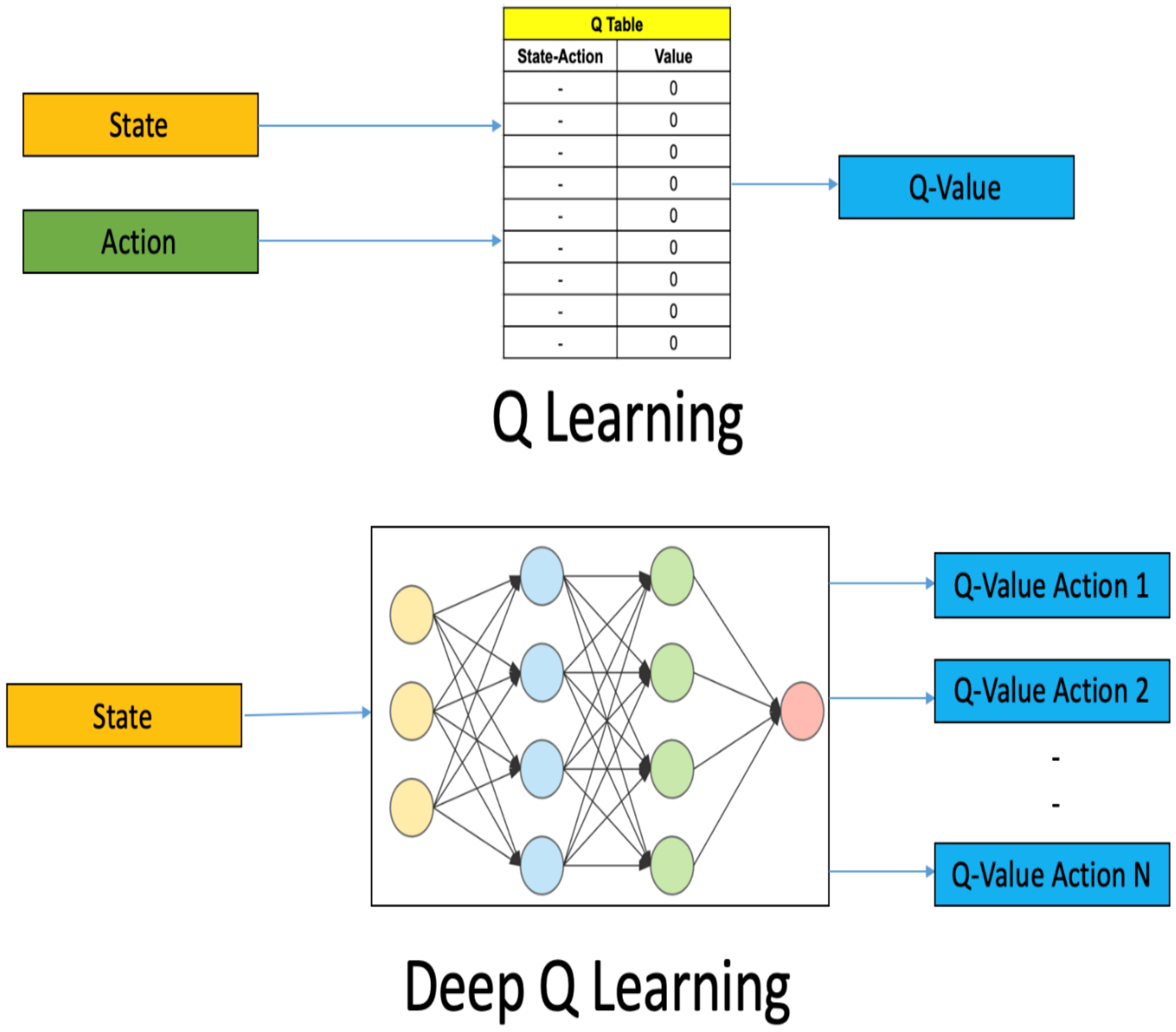


Figure 2. 4*: DQN Process.*

2.1.4 Multi-Objective Deep Reinforcement Learning

MODRL has been observed as a significant research topic due to the Multi-Objective characteristics of adaptive optimal control problems and several practical sequential decision-making scenarios in the real-world. In conventional Deep Reinforcement Learning (DRL), the objective usually revolves around maximizing a solitary cumulative reward signal. In addition, MODRL entails the establishment of several objective functions, each of which represents a separate goal or criterion that the agent strives to optimize. The presence of varying sizes and units among these objectives poses a challenge to the optimization process. On the other hand, MODRL tackles issues that involve many objectives, which may potentially be in conflict, and require an agent to find a balance. This is especially pertinent in intricate real-life situations when decision-making entails balancing various objectives. (Liu, Xu and Hu, 2015).

As well as, numerous real-world problems have multiple, probably incompatible objectives. For instance, an agent may want to maximize the performance of a web application server while minimizing its power consumption. So, such problems can be demonstrated as Multi-Objective Markov Decision Processes (MOMDPs) as well as solved with MODRL. As it is typically not clear how to assess accessible trade-offs between different objectives a priori, there is no single best policy. MODRL is an extension of traditional DRL that handles situations where there are multiple incompatible objectives to optimize concurrently. In addition, MODRL aims to find policies that provide a trade-off between these objectives. In MODRL, the main goal is to learn policies that can handle a Pareto-optimal trade-off between different purposes as opposed to a single best objective. Furthermore, each objective is considered a separate task in MODRL, which is a type of multi-task learning. The agent gains the ability to weigh the objectives when making decisions as shown in Figure 2.5 (Mossalam *et al.*, 2016) (Thi Nguyen et al., 2020).

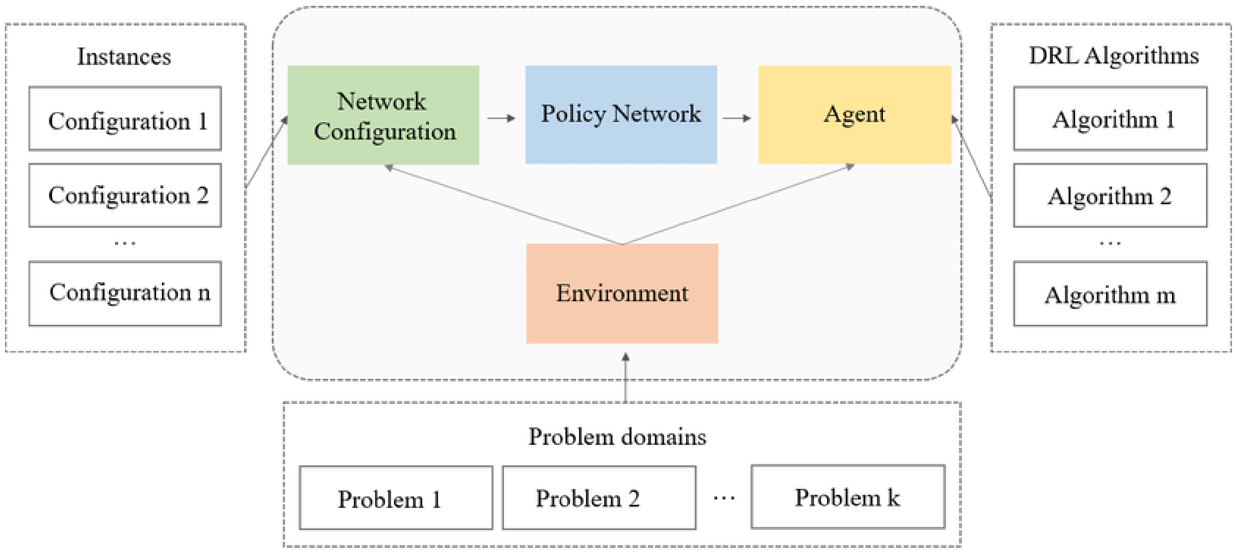


Figure 2. 5 :*System architecture of MODRL Framework*

2.1.5 Multi-Objective Optimization Using Evolutionary Algorithms

Recently, Evolutionary Multi-Objective Optimization (EMO) has become a useful and popular field of application and research. As well as a specific area of Artificial Intelligence and optimization that focuses on solving problems with multiple objectives. Evolutionary Optimization (EO) algorithms use a population-based approach in which a new population of solutions evolves with each iteration, and more than one solution contributes in an iteration. Multi-Objective optimization is the process of optimizing many objectives that are in conflict with each other. The objective is to identify a collection of solutions that are situated on the Pareto Front, where improving one objective requires sacrificing others. Evolutionary Algorithms (EAs), which draw inspiration from natural selection, are frequently employed to address intricate challenges, giving rise to Multi-Objective Evolutionary Algorithms (MOEAs). MOEAs, such as Genetic Algorithms and Differential Evolution, utilize populations of potential solutions, apply genetic operators, and iteratively improve solutions over generations to explore the Pareto Front. Evaluations of solutions are conducted using Pareto dominance, while diversity preservation approaches guarantee the presence of a well-balanced collection of Non-Dominated solutions. MOEAs are utilized in various fields such as engineering design and finance to tackle complex decision-making situations that involve numerous conflicting objectives. They offer decision-makers a variety of trade-off solutions along the Pareto Front to facilitate informed decision-making. Furthermore, Multi-Objective Optimization problems by their very nature led to a set of Pareto-optimal solutions that need to be further processed in order to arrive at a single optimal solution. Since the usage of population in an iteration allows an EO to concurrently locate numerous non-dominated keys, achieving the first task node becomes a very natural proposition for using an EO. This reflects a trade-off between objectives in a single simulation run (Deb and Deb, 2014).

Furthermore, MOEA are suitable for handling a wide range of difficult Multi-Objective issues that involve two or three objectives. MOEAs have been developed to effectively handle complex Multi-Objective Optimization problems (MOPs) that involve two or three objectives. (Von Lücken et al., 2014). In addition, Multi-Objective Optimization, specifically MOEAs, is essential for addressing intricate issues that entail conflicting aims. This method enables a sophisticated comprehension of trade-offs by showcasing a varied array of solutions on the Pareto Front, providing decision-makers with a broad spectrum of possibilities. The practicality of this approach is relevant in practical situations, such as engineering design and resource allocation, when decision-makers need to take into account numerous objectives at the same time. Multi-Objective Evolutionary Algorithms are highly effective in enhancing the variety of solutions, managing uncertainties, and adjusting to changing settings. They offer a strong foundation for optimizing many interrelated objectives simultaneously. MOEAs, by preventing premature convergence, provide flexibility and scalability, making them important tools for solving NP-Hard problems and various complex optimization issues in different areas. Several Multi-Objective Optimization Algorithms are to discover a large number of Pareto ideal vectors that are consistently distributed along the PF and therefore good representatives of the whole PF (Zhang et al., 2007). MOEA/D and NSGA are two significant MOEAs that are widely recognized. (Özdemir et al., 2013).

2.1.5.1 Multi-Objective Evolutionary Algorithm based on Decomposition

MOEA/D is a widely used optimization algorithm employed for solving problems involving Multiple-Objectives. The process involves breaking down a Multi-Objective Problem (MOP) into several individual optimization sub-problems and then optimizing them collaboratively. MOEA/D is an algorithmic approach that breaks down Multi-Objective Optimization problems into Single-Objective sub-problems, allowing parallelized optimization. It balances convergence and diversity, providing a comprehensive set of trade-off solutions. It's adaptable, suitable for large-scale optimization challenges, and can be enhanced through hybridization with other optimization techniques. As well as, it is designed to find a set of explanations that represent a trade-off between conflicting objectives. In MOEA/D, decomposition mechanisms are used to push the population to the Pareto Optimal Front (POF), whereas a set of consistently distributed weight vectors is implemented to preserve the variety of the population. In addition, MOEA/D works by decomposing the Multi-Objective problem into multiple Single-Objective sub-problems and then optimizing them simultaneously. Additionally, the concept of sub-problem area—which was initially introduced in MOEA/D—can help advance the harmony between the algorithm's exploration and exploitation as it is being developed. Furthermore, the value of the combination function determines which solutions are chosen in the MOEA/D, greatly increasing the pressure to pick for the genuine POF and providing additional benefits while resolving MOPs. In order to implicitly achieve good population diversity, a set of regularly distributed weight vectors is utilized simultaneously (Qiao *et al.*, 2019)(Chen *et al.*, 2021).

Typically, MOEA/D works as (Li, 2021):

(1) Initialization: Choose a heuristic method or initialize a population of potential solutions at random.

(2) Decomposition: Divide the Multi-Objective problem into several smaller scalar problems. While focusing on a single target to be optimized, each sub-problem takes the other objectives into account as constraints or reference points.

(3) Evolution: Apply an evolutionary algorithm to each sub-problem to generate a population of solutions over numerous generations (evolutionary algorithms are often a version of genetic algorithms). Operations like selection, crossover, mutation, and replacement are part of the evolution process.

Solutions are assessed during evolution based on how well they perform in relation to the relevant sub-problem. Scalarization is frequently used to measure performance, such as the Tchebycheff approach, weighted sum, or Pareto dominance.

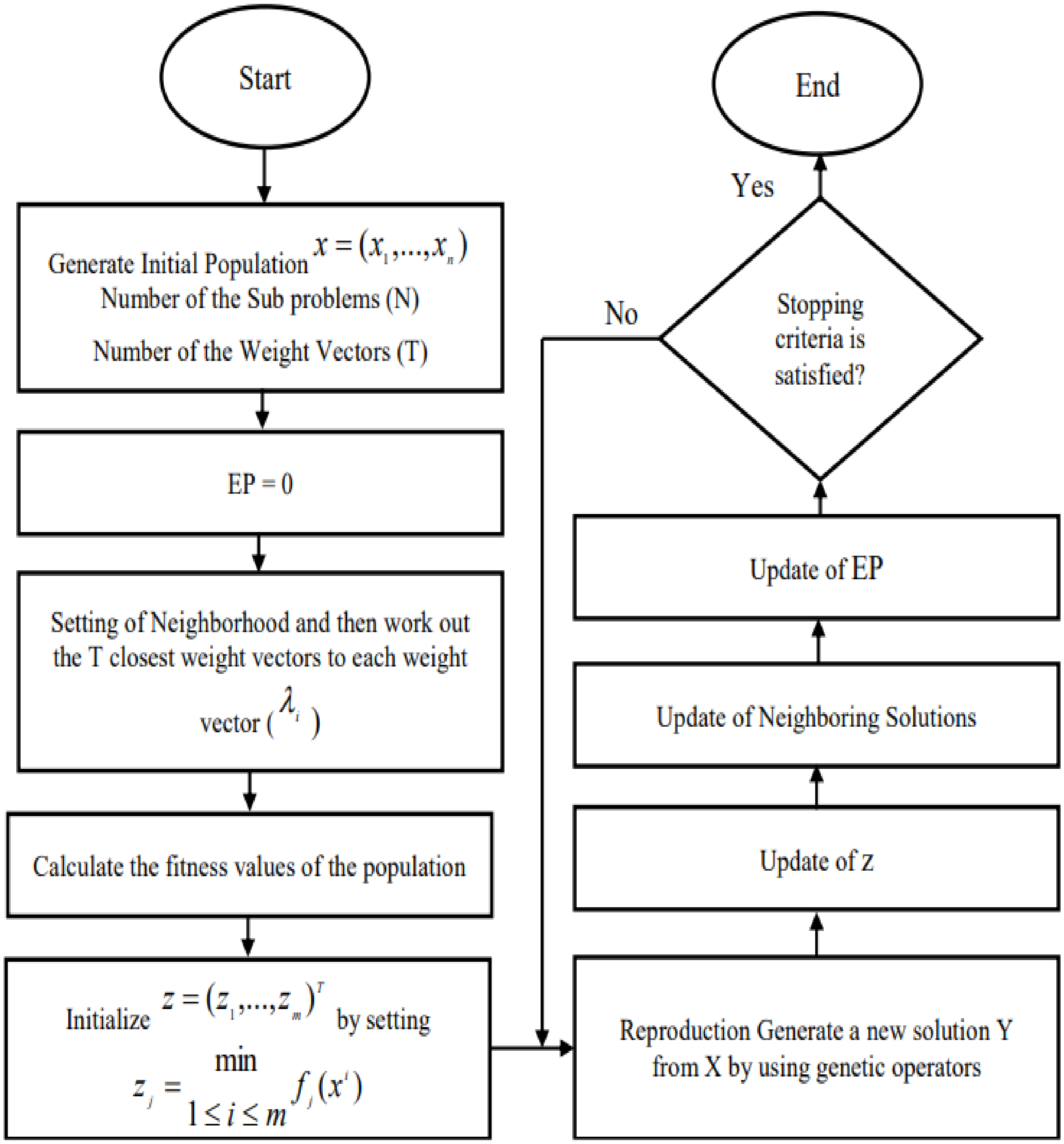
(4) Update External Archive: Keep the finest solutions thus far in a repository or archive that is external to the organization. This repository records all Non-Dominated (Pareto-optimal) solutions found during the optimization process.

(5) Neighborhood Selection: MOEA/D often services a neighborhood assembly that describes how sub-problems are organized. The optimization process can be made better by exchanging information between solutions to nearby sub-problems.

(6) Convergence Criterion: MOEA/D characteristically has a termination condition based on the computational budget or the number of generations. When this requirement is satisfied, the algorithm ends.

(7) Result Extraction: The Pareto-optimal answers kept in the external archive once the algorithm finishes up represent the trade-off solutions for the Multi-Objective problem. These solutions offer a variety of possibilities for distinct trade-offs between objectives to and decision-makers. Figure 2.6 shows the flow chart of MOEA/D.

Thus, MOEA/D is essential for addressing intricate issues that have contradictory aims. This method enables a sophisticated comprehension of trade-offs by showcasing a varied array of solutions on the Pareto Front, providing decision-makers with a broad spectrum of possibilities. The practicality of this concept can be applied to real-life situations, such as engineering design and resource allocation, where decision-makers need to take into account numerous objectives at the same time.



Start
Generate Initial Population $x = (x_1, \ldots, x_n)$
Number of the Sub problems (N)
Number of the Weight Vectors (T)
EP = 0
Setting of Neighborhood and then work out the T closest weight vectors to each weight vector (λ_i)
Calculate the fitness values of the population
Initialize $z = (z_1, \ldots, z_m)^T$ by setting $z_j = \min_{1 \le i \le m} f_j(x^i)$
Reproduction Generate a new solution Y from X by using genetic operators
Update of z
Update of Neighboring Solutions
Update of EP
Stopping criteria is satisfied?
No
Yes
End

Figure 2. 6: *Flow chart of MOEA/D*

2.1.5.2 Non-Dominated Sorting Genetic Algorithm

NSGA II is an Evolutionary Algorithm advanced as a response to the limitations of early evolutionary algorithms. It is a popular evolutionary optimization algorithm used in Multi-Objective Optimization problems. NSGA II starts with the initialization of random populations. In addition, it is considered to be the discovery of a set of solutions that are considered Pareto-optimal or Non-Dominated(Mohamad Shirajuddin et al., 2023). As well as, NSGA-II outperforms two contemporary MOEAs: strength-Pareto EA (SPEA) and Pareto-archived evolution strategy (PAES) in terms of converging near the true Pareto-optimal set and finding a diverse set of solutions. Pareto optimality means that no solution in the set can be enhanced in one objective without worsening another objective. Constrained Multi-Objective Optimization is significant from the point of view of applied problem solving. the application of NSGA-II to real-world and more complex Multi-Objective Optimization problems (Deb *et al.*, 2002a). The NSGA-II-Algorithm has three features (Kaur et al, 2018):

1- Elitist principle: The most effective responses (Non-Dominated solutions) from the current population are kept and passed on directly to the following generation without any modification under the elitist method used by NSGA-II. This makes sure that the most well-known solutions do not disappear during the course of evolution. Over generations, elitism helps to retain a set of Pareto-optimal solutions and prevents the algorithm from accelerating its convergence to less-than-ideal solutions.

2- Non-Dominated solutions: A set of Non-Dominated (Pareto-optimal) solutions will be discovered using NSGA-II. It tries to find solutions that, without affecting at least one other target, cannot be used to improve any one objective. The algorithm clearly divides the population into various Pareto Fronts based on non-dominance, guaranteeing that the best options for balancing the competing goals are included in the outcome.

3- An explicit diversity-preserving mechanism. It is included in NSGA-II, mostly through crowding Distance sorting. Solutions are further arranged according to their crowding distances after being divided into Pareto Fronts. Higher crowding distance solutions are selected during the selection process. In order to ensure that the final set of solutions takes place over a broad range of objective space and captures many trade-offs, this technique promotes variation within each Pareto Front.

The original population is generated as typical. After preparing, the population is divided into fronts and sorted by Non-Domination. Only non-dominant individuals of the present population reside by the first-front. The entities of the second front are dominated by first front entities only. Each member of every front is given a fitness value, and these values are determined by the level of the front. Individuals at the first level receive one fitness value, those at the second level receive two, and so on, as shown in figure 2.7 (Verma, et al, 2021)(Deb et al., 2002b).

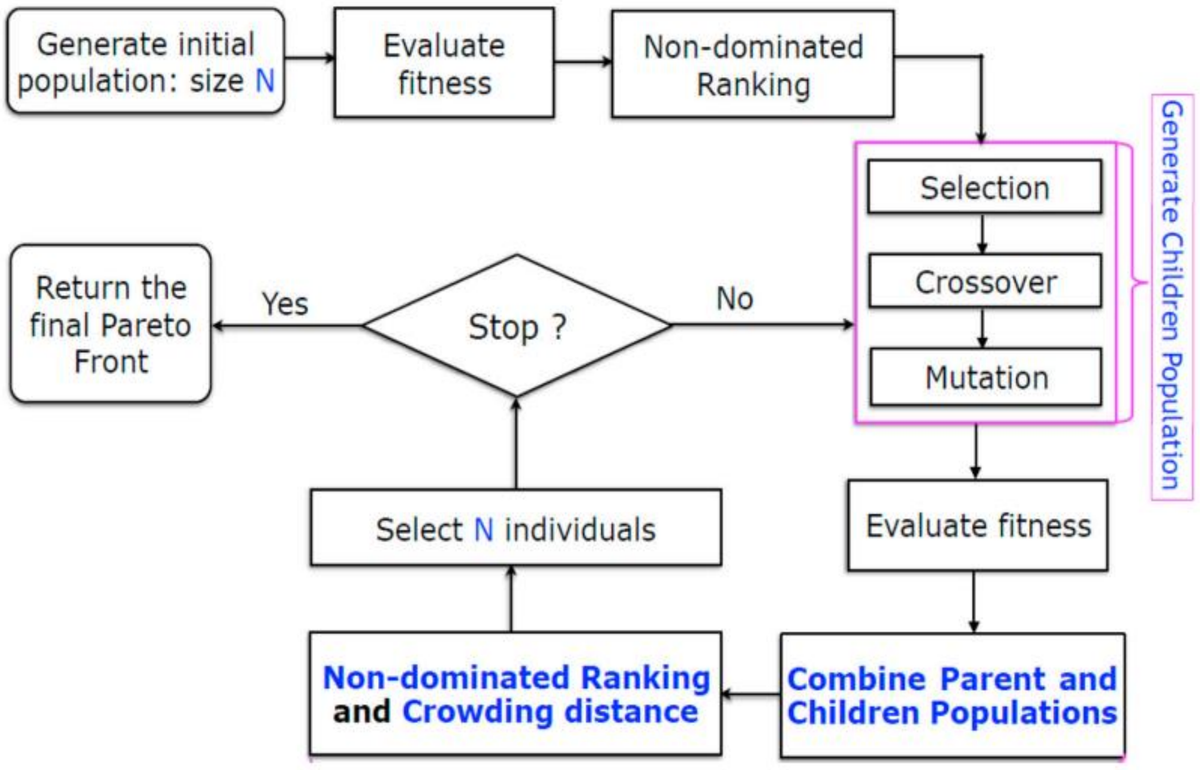

Generate initial population: size N
Evaluate fitness
Non-dominated Ranking
Selection
Crossover
Mutation
Generate Children Population
Return the final Pareto Front
Yes
Stop ?
No
Select N individuals
Evaluate fitness
Non-dominated Ranking and Crowding distance
Combine Parent and Children Populations

Figure 2. 7: *Flow chart NSGAII.*

2.2 RELATED WORK

FC is a novel approach. Resource management is one of the most significant research directions in FC models. Efficient resource management will improve the FC system's performance. Task scheduling and Task allocation are necessary requirements for performance optimization in the FC, and this study aims to investigate the problems in scheduling tasks and allocating tasks. Numerous representative works in the literature are discussed and listed below.

(Wu *et al.*, 2021) employed Deep Reinforcement Learning (DRL) to address the scheduling issue in Edge Computing (EC) with the aim of enhancing the QoS delivered to users of industrial Internet of Things (IoT) applications. The authors introduced a novel scheduling algorithm called the Deep Intelligent Scheduling Algorithm (DISA), which utilizes a Double Deep Q Learning. That is an EC-based network architecture for communication scheduling, and makes sure policies that are created dynamically are stable. Simulation results were implemented, indicating that the DISA can attain better network performance than the traditional scheduling algorithm.

(Gazori et al., 2020c) outlined the task scheduling of Fog Computing system based IoT applications with the purpose of computing cost under the resource, minimizing long-term service delay, and meeting deadline constraints. The authors introduced a novel scheduling technique that utilized the target network, employing a Double Deep Q-Learning (DQL) approach. The goal of the algorithm was to maximize its objective function so that results would be more stable. The results demonstrate that the suggested algorithm outperforms several baseline methods in terms of computation cost, energy consumption, task accomplishment, and service delay, as well as handling load balancing and single point of failure challenges.

(Wei *et al.*, 2018) investigated whether Cloud-based application users Could learn to make wise job dispatching choices on their own. This problem is solved using a smart QoS-aware job-scheduling framework for application providers. A DRL-based job scheduler is a main element of this system. With respect to the characteristics of RL, the proposed algorithm could dynamically familiarize itself with the fluctuations and uncertainties of workloads. According to the simulation results, the proposed job-scheduling strategy can effectively decrease the average work response time compared with other baseline algorithms in the IoT edge system.

(Sheng *et al.*, 2021) studied the computationally intensive task scheduling problem. Both the task assignment and the task execution order had to be optimized while taking into account the different tasks and resources that were available. The authors formulated optimization problems as a Markov Decision Process model. A policy-based reinforcement learning (RL) technique was presented to address the task scheduling problem, utilizing a fully connected Neural Network (NN) to extract the relevant characteristics. Evaluation results validated that the proposed algorithm achieves a better success ratio and cumulative task satisfaction degree than the baseline task scheduling algorithms.

(Sellami *et al.*, 2020) introduced Deep ReinforcemenLearning energy efficient task scheduling in a Software-Defined Network, that is based Fog IoT network, which reduces traffic overhead and network latency by centralizing network control. In addition, the proposed algorithm addresses the resource-planning problem and performs efficient energy-task allocation in a distributed and dynamic IoT environment. The performance evaluation of the simulation findings shows the effectiveness of the proposed solution in increasing energy efficiency, ensuring lower-latency communication, and performing both global and local optimizations.

(Qi, Zhuo et al., 2020b) defined a task scheduling problem for a Cloud-Edge computing architecture. To address the problem of long delays in achieving DL tasks in the EC layer. proposed a scheduling algorithm dependent on asynchronous advantage Actor-Critic based on DRL and modeled it as a Markov Decision Process. The results showed that the proposed algorithm can reduce task processing time compared to the existing RL-G and DQN algorithms.

(Lakhan *et al.*, 2022) examines the issue of resource allocation in Fog networks enabled by Software-Defined Networking (SDN). The study presents a novel architecture that utilizes containers and incorporates several Fog nodes. In addition, develops Deep-Learning-Network-Based Resource Allocation (DQBRA) by considering the architecture. This approach has multiple components to address the problem. The components consist of a mobility controller, a resource search mechanism, and a resource allocation and task migration module. Performance evaluation shows suggested runtime and schemes outperform existing schemes and frameworks.

(Mseddi *et al.*, 2019) presents a novel online resource allocation method designed for dynamic FC environments. The objective is to maximize the number of user requests that are fulfilled within a predetermined latency threshold. The FC environment is represented as a Markov discrete method, taking into account the dynamic behavior and movement of FN as well as the availability of resources. Next, introduce an intelligent Deep-Reinforcement Learning system for resource allocation. The suggested technique demonstrates near-optimal performance compared to heuristic, state-of-the-art alternatives.

(Zheng *et al.*, 2022) presented a novel approach for workload scheduling with Deep Reinforcement Learning to achieve workload balance, minimize service time, and decrease the rate of failed tasks. Meanwhile, employ Deep-Q-Network (DQN) techniques to address the intricacy and large dimensionality of the workload scheduling problem. The simulation findings demonstrate that the proposed technique outperforms current approaches in terms of service time, Virtual Machine (VM) use, and unsuccessful job rate. The utilization of the DRL method offers an effective resolution to the task allocation issue in Edge Computing.

(Jin *et al.*, 2023) address the job scheduling problem specifically with IoT systems within a Cloud Computing environment. The objective is to minimize the duration of the task. The task scheduling problem is well recognized as a formidable challenge. Present a unique and efficient Reinforcement Learning algorithm for addressing the task scheduling problem in IoT systems. This algorithm integrates combinatorial optimization techniques to ensure that it achieves stable lower limits. Perform tasks in a group, select tasks using reinforcement learning, and solve them using combinatorial optimization techniques. The experimental results demonstrate that the suggested algorithm exhibits exceptional performance across many contexts.

(Wang et al., 2023) introduce a framework called QMTSF, which utilizes Q-learning for Multi-task Scheduling Framework. The framework comprises two stages: Initially, jobs are given to appropriate servers in the cloud environment based on server type. Furthermore, a more advanced Q-Learning algorithm known as UCB-based Q-Reinforcement Learning (UQRL) is used on each server to allocate tasks to a Virtual Machine. The agent employs a sophisticated decision-making process by using its prior experiences and interactions with the environment. Furthermore, the agent acquires knowledge through the use of rewards and punishments in order to develop the most effective approach for allocating tasks and scheduling them on various virtual machines. The goal is to minimize the total length of task execution, and the average time used for processing jobs while still guaranteeing that task deadlines are met. Performed simulation studies to assess the efficacy of the suggested mechanism in comparison to conventional scheduling techniques such as Particle Swarm Optimization (PSO), random, and Round-Robin (RR). The experimental results indicate that the proposed QMTSF scheduling framework surpasses previous scheduling techniques in terms of both the makespan and average task processing time.

(Goudarzi et al., 2023)introduce a novel technique, namely Deep Reinforcement Learning Intelligent Strategy (DRLIS), which utilizes Deep Reinforcement Learning to effectively enhance the response time of diverse IoT systems and evenly distribute the workload among Fog and Edge servers. utilized DRLIS as an operational scheduler within the Fog function-as-a-service system architecture to establish an integrated Fog-Edge-Cloud server-less Computing environment. The results show that, in comparison to metaheuristic algorithms and other Reinforcement Learning techniques, DRLIS efficiently reduces the operating costs of Internet of Things applications in load balancing, reaction time, and weighted cost, respectively.

After reviewing most of the articles published recently to tackle task scheduling. They distributed this problem as a single objective. However, this problem must be solved by concurrently optimizing more than one objective. In addition, Resource Utilization and Latency are crucial factors in Fog Computing schedulers. However, it is significant to note that numerous previous works tend to prioritize different metrics, resulting in outcomes that may not accurately reflect the critical aspects of Resource Utilization and Delay management. Thorough analysis of the performance aspects

***Table 2. 1**: Comparative among the Reviewed works*

Ref.	**Architecture**	**Algorithm**	**Objectives**	**MODRL**	**Simulation Tools**	**Shortcomings**
Wei *et al.*	Centralized	DRL based on a (DQN)	Response time, and resourceutilization rate	×	Python, TensorFlow	High response time
Mseddi *et al.*	Cluster	DRL	Success ratio ,Cumulative reward	×	Not mentioned	Lack of resource utilization, and Latency
Sellami *et al.*	Centralized SDN Scheduler	DRL based SDN	Available Energy and Latency	×	Not mentioned	High latency and not taken resource utilization
Qi, Zhuo et al.	Centralized	DRL according to the (MDP)	Task Failure and delay	×	Tensorflow	the training model needs expensive GPU resources.
Gazori et al.	Hierarchical	RL based on a (DDQN)	Latency, Response time, makespan time, and waiting time	×	Python,Keras SimPy	High Latency, Response, makespan , and waiting time
Wu *et al.*	Centralized	DRL based on a (DDQN)	Latency, bandwidth, delivery time	×	Python, Tensorflow	Not involved resource utilization.
Sheng *et al.*	Only Edge Layer	DRL according to the (MDP)	Cumulative task satisfaction degree vs Task arriving rate	×	Python, Pytorch	Not involved resource utilization, and task completion time

Lakhan *et al.*	Centralized SDN Scheduler	DRL based on a (DQN)	Total cost	×	Python-Ruby-Perl	delay-sensitive and delay-tolerant of IoT workloads
Zheng *et al.*	Centralized	DRL based on a (DQN)	Failed Tasks	×	CloudSim	Not involved resource utilization, and task completion time
Jin *et al.*	Not mentioned	RL based QL.	task runtime	×	Not mentioned	exclusive analysis of theperformance aspects
Wang et al.	Not mentioned	QRL	Makespan and complete time	×	CloudSim	High Makespan and complete time
Goudarzi et al.	Hierarchical	DRL according to the (MDP)	load balancing, response time, and weighted cost	×	FogBus2	the limited resources

2.3 Summary of the Chapter

In this chapter, two main topics are explained. Firstly, a background on these tools that have been used in this work, such as scheduling in Fog Computing, their implementation, and some methods (DQN, NSGA2, and MOEA2). Secondly, a literature review about scheduling in Fog Computing based on Reinforcement Learning. In addition, a comprehensive table is provided, outlining important aspects of significant contributions to scheduling in Fog Computing, along with their corresponding limitations.

CHAPTER THREE

Research Methodology

3.1 INTRODUCTION

Multi-Objective Deep Reinforcement Learning, which supports scheduling in Fog Computing, addresses the challenging problems of task scheduling and task allocation in Edge environments. By integrating MODRL, the scheduler gains the ability to optimize many objectives simultaneously, including reducing latency, maximizing resource use, and improving the QoS. This sophisticated method utilizes Deep Reinforcement Learning to adjust and develop scheduling policies according to the dynamic and diverse characteristics of Fog Computing systems. The objective of MODRL-based scheduling is to achieve a balance between opposing goals by considering many objectives. This leads to FC systems that are more adaptive, capable of efficiently handling varying tasks and meeting the strict requirements of various applications.

3.2 Proposed System Architecture

The system model consists of a three-layer architecture:

- The IoT layer generates high-rate tasks . The IoT layer consists of multiple IoT devices and gateways that communicate with the upper layers. IoT devices have the following characteristics:
    - Provided with computation capability (CPU, RAM, storage, etc.).
    - Have very strict constraints in terms of CPU, and memory.
    - Provided sensors that generate the tasks.
    - Supported wireless communication (4G, 5G, and WIFI)

- The FC layer consists of multiple Fog colonies, and each Fog colony has multiple Fog nodes endowed with computing capabilities. Deployed a Central Smart Fog Orchestrator that manages the Fog system. The FN hosts IoT applications to process tasks received from the IoT layer. The Fog nodes can be:
    - Provided with computing capability (CPU, RAM, storage, etc.).
    - Passive Fog node (battery-powered).
    - Fog devices are provided with wireless and wired capabilities.
    - The Smart Fog Orchestrator is endowed with the intelligence to allocate tasks and instruct each Fog node on how to schedule the tasks.
- The Cloud Computing layer comprises one or multiple powerful Cloud servers that run IoT applications to process tasks. The Cloud Computing layer is:
    - Deployed after the Fog layer (far from the IoT layer).
    - Provided with high computation capability (CPU, RAM, storage, etc.).
    - Support wireless or wired communication.
    - Have limited communication capability: Communication between Cloud servers and mobile devices is slow, usually through the Internet.

The proposed system architecture is summarized in Figure 3.1:

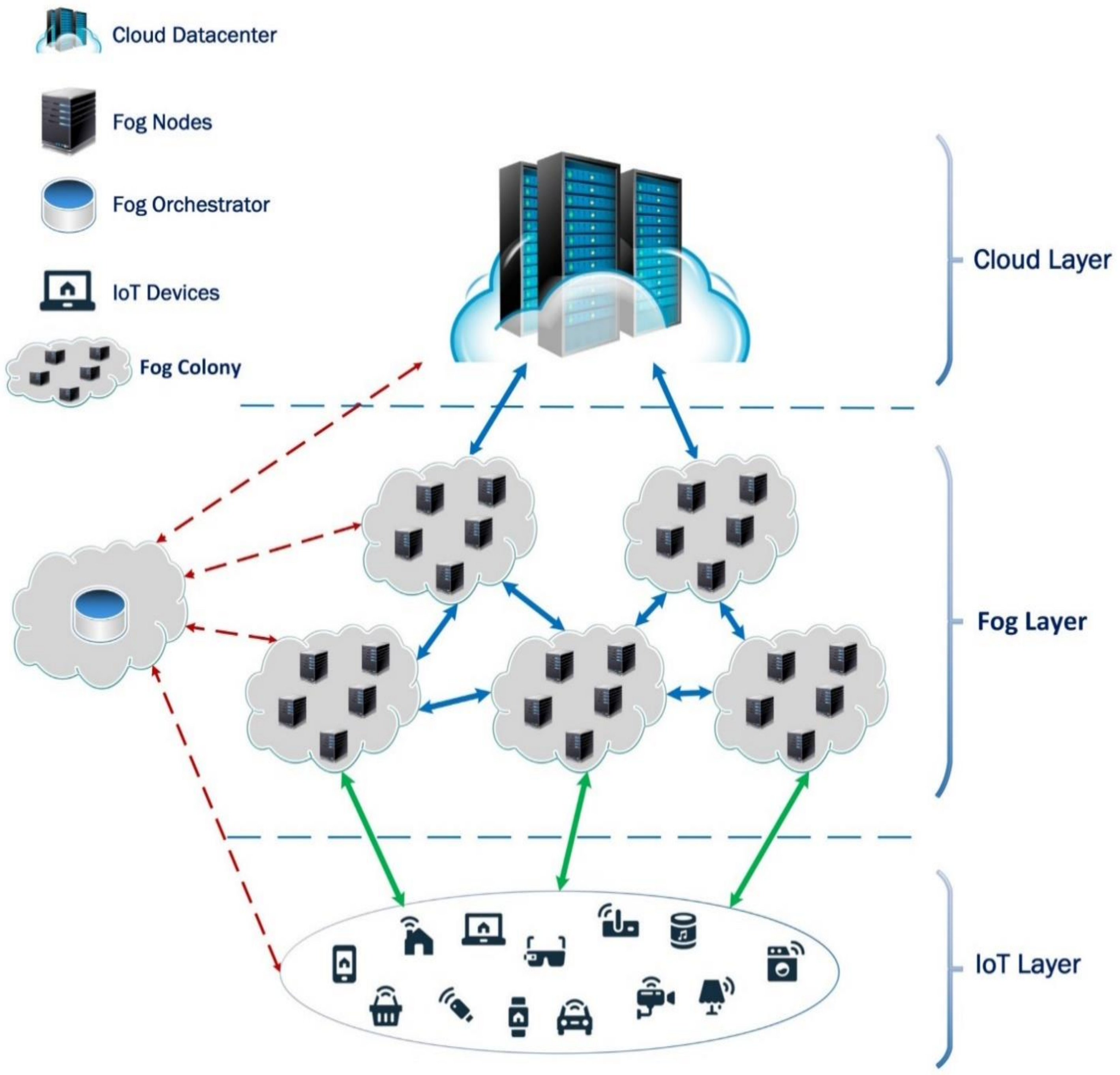


Figure 3. 2*: Proposed System Architecture*

3.3 Proposed System Model

In this section, the communication, processing, and task models have been defined.

3.3.1 Communication Model

Because communication between the IoT and the upper layers is entirely wireless, impact factors in wireless communication are considered, such as

- Distance/Signal area.
- Bandwidth.
- Transmission rate, data rate, and speed.
- Transmission power and signal strength.
- Channel gain.
- Path Loss.
- Fading.
- Interference.
- Noise power.
- Uplink and downlink.

Consequently, these parameters have an impact on the transmission time (communication delay). However, the connection between the Cloud and Fog layer uses wired technologies; thus, it is not subjected to the same factors.

Formally, the communication model is designed based on two transmission supports and the Shannon formula (Verdú, 1998).

- The wireless transmission support that IoT devices use transmits tasks to the upper layer (Fog, Cloud).
- The wired transmission support is used to transmit the intra- and inter-Fog colonies to the Cloud.
- The communication delay is the total transmission time Tr_i of task i from the source IoT device to the destination Fog node.

$$Tr_i = Dp_i + Dc_i \tag{3.1}$$

Where Dp_i is the wireless communication delay required to transmit the task from the IoT device to the wireless gateway and Dc_i is the wired communication delay.

The wireless communication delay Dp_i required to transmit the task from the IoT device to the wireless gateway is submitted to wireless communication constraints. Moreover, a scheduling delay is considered (the scheduling delay is discussed below). Wireless communication delay is expressed as follows:

$$Dp_i = Ds_i + \frac{\theta_i}{R_i} \tag{3.2}$$

Where R is the data rate. R is expressed as follows:

$$R_i = Wlog_2\left(1 + \frac{P_i H_i}{N_0}\right) \tag{3.3}$$

where W represents the channel bandwidth, P_i is the transmission power, H_i is the channel gain, and N_0 is Noise power.

The wired communication delay Dc_i is generated whereas transmitting task i from one node to another until the task reaches its destination. This type of delay depends on the number of hops crossed by the task and the bandwidth B_k of the transmission support link k used to transmit the task. Similar to communication delay, scheduling delay was also considered.

$$Dc_i = Ds_i + \sum_0^k \frac{\theta_i}{B_k} \tag{3.4}$$

The scheduling delay Ds_i represents the waiting time φ_n of task i in the queue of the network device before transmission. The scheduling delay depends not only on the task position n in the queue but also on the task Priority P_i, the tasks with higher priorities are immediately scheduled first to be delivered, whereas tasks with fewer priorities are more delay-tolerant; hence, they can have more waiting time in the queue. The scheduling delay is expressed as follows:

$$Ds_i = \sum_0^n \varphi_n \qquad (3.5)$$

where n is the waiting time equal to the throughput of the network device.

The communication delay model is summarized in the next Figure 3.2:

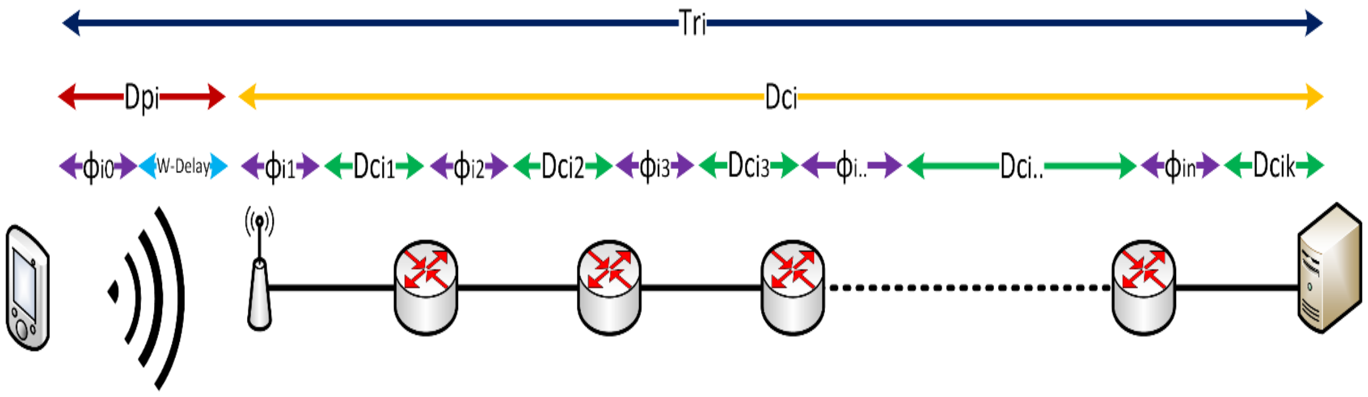


Figure 3. 3: *Communication Delay*

3.3.2 Processing Model

Supposing that all processing entities (Fog node and Cloud) are provided with the same parameters but with different power characteristics.

- CPU Frequency/computing rate
- CPU Load

The computing time or processing delay is affected by CPU Frequency, task CPU requirement, and waiting time.

Formally, similar to the communication model, the tasks are processed based on their Priority; high-Priority is processed first, and low-Priority tasks are delayed.

The total computing delay Tc_i is described in the next equation:

$$Tc_i = Cs_i + Cp_i \qquad (3.6)$$

Where Cs_i is the scheduling delay for the task to be processed and Cp_i is the time required for the task to be processed by the computing node.

Scheduling delay Cs_i represents the waiting time of task i in the queue of the computing node. As in the communication model, the scheduling delay for computing depends on both task position m in the CPU queue and task Priority ρi. In this case, the scheduling delay is related to Cp_i of the leading task m, which by itself relies on the CPU speed and CPU requirements of the tasks. The scheduling time is expressed as follows:

$$Cs_i = \sum_0^m Cp_m \qquad (3.7)$$

The computing time Cp_i of task i is the time required by the computing node to process tasks. The computing time is calculated based on the CPU speed ω_z allocated by the computing node (VM, container, etc.) to process task i, and the task CPU requirement δ_i. The computing time is described by the following equation:

$$Cp_i = \frac{\delta_i}{\omega_z} \qquad (3.8)$$

3.3.3 Task Model

The generated tasks from the IoT can be modeled with the following parameters:

- Task size
- Task CPU requirement
- Task generation rate
- Task Priority based on application type

Formally, IoT devices generate multiple tasks i, the task generation rate follows the Poisson process, and the Poisson distribution is expressed as follows (Sherbrooke, 1968):

$$Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{3.9}$$

where λ is the rate parameter, which is a measure of frequency (the average rate of events, in our case, traffic rate) per unit time (for example 10kbps).

The generated task is defined with the following tuple $< \theta i, \delta i, \rho i >$:

- θi is the size of the task in bytes. It impacts communication delays.
- δi is the CPU requirement, which represents the number of CPU instructions required to process a task. It can influence the load in the computing layers (Fog and Cloud). Heavy processing tasks can result in node overload.
- ρi is the task Priority, which is classified into three categories: high, medium, and low. Priority depends on the type of IoT application; for instance, critical-latency applications are classified as high-Priority.

3.4 System Framework

The core mechanism of the system framework revolves around seamless Orchestration and efficient allocation of computing resources to tackle tasks in dynamic environments. The mechanism of the system framework works as follow:

- To keep track of the environment, the system state is periodically sent as “state information messages” to the Smart Fog Orchestrator.
- Based on the state information of the environment, the Fog Orchestrator runs the Intelligent DRL algorithm to select the most suitable Fog colony or Cloud to process tasks.
- After that, the tasks will be sent for processing

- Although the results of the processing sent to the users are neglected, the Fog Orchestrator receives SLA feedback for each allocated and scheduled task.

3.4.1 The Reinforcement Learning Environment

In this section, a custom-Reinforcement-Learning environment is introduced, designed specifically for task assignment in such systems. The environment provides an interface for the agent to interact with the system, observe its state, take action, and receive rewards based on different performance criteria.

A. Environment Design

The Reinforcement Learning environment is implemented as a subclass of the '*gym.Env*' class, OpenAI Gym Environment is a fundamental component of the OpenAI Gym toolkit, which is a Python library widely used for developing and comparing Reinforcement Learning algorithms. It provides a standardized interface for defining and interacting with Reinforcement Learning environments. In addition, ensures compatibility with various Reinforcement Learning algorithms and frameworks. The environment takes several parameters during initialization, including the number of nodes in the system, reward mode, and maximum number of time steps.

1. State Space

The state space of the environment represents the state of the system and the task to be assigned. It is defined as a continuous box space with the following components:

- Load of each node: The Load of each node in the system is represented as a floating-point value between zero and the maximum load value.
- Task size: The size of the task waiting to be assigned, represented as a floating-point value between zero and the maximum task size.
- Task CPU requirement: The CPU requirement of the task waiting to be assigned is represented as a floating-point value between 0 and the maximum CPU requirement.
- Task Priority: The Priority of the task waiting to be assigned is encoded as a one-hot vector with three values (low, medium, and high).

2. Action Space

The action space of the environment represents the actions that the agent can take to assign a task to a specific node. It is defined as a discrete space in which the number of nodes is the number of possible actions.

3. System Representation

The environment maintains an internal representation of the system consisting of multiple nodes. The system class encapsulates the behavior of nodes and provides methods for task assignment, load processing, and task generation.

B. Environment Dynamics

The environmental dynamics simulate the interaction between the agent and the system. For each time step, the agent takes action by assigning a waiting task to a node. The system processes assigned tasks, updates node loads, and generates new tasks. The reward is calculated based on different criteria, including the load distribution, communication cost, and task priorities.

1. Task allocation and Task Scheduling

When an agent assigns a task to a node, the environment updates the system accordingly. The node processes the assigned task, and its load is updated based on the CPU requirements of the task. The task assignment action is represented as an index of a node in the action space. The agent systematically sends the task to the cloud instance (a node with high resources that imposes a large communication cost, as it is supposed to be far in the network) when no resources are left unused in the Fog colonies. At the node level, tasks are first scheduled based on their Priority and then based on their order of arrival. If the CPU capacity is insufficient to handle the next task in the queue, it will systematically try to allocate the task until no task that can fit within the node’s resources is left. The Orchestrator receives the system state about task Fog nodes and Fog colonies and commands the IoT device to send a task to write to the Fog node and command the Fog node to schedule those tasks based on the Orchestrator's decision.

2. Reward Calculation

The reward function's design is essential in influencing the learning process of the MODRL algorithm. The reward function determines the numerical feedback that the agent receives as a result of its actions and decisions in the environment as shown in figure 3.3. When considering Fog Computing scheduling with objectives like Load, Distance, and Priority, the reward function should accurately represent the desired balance and priorities. The reward function in deep reinforcement learning for scheduling directs the agent's decision-making process, aiming to optimize resource use and minimize task execution delays. It is based on three factors: Load, Distance, and Priority. Pareto's goals guide the agent's behavior, enabling adaptability to changing Fog environment conditions. The scheduling agent's efficiency and intelligence are enhanced through reward calculation, improving the efficacy and productivity of Fog Computing systems.

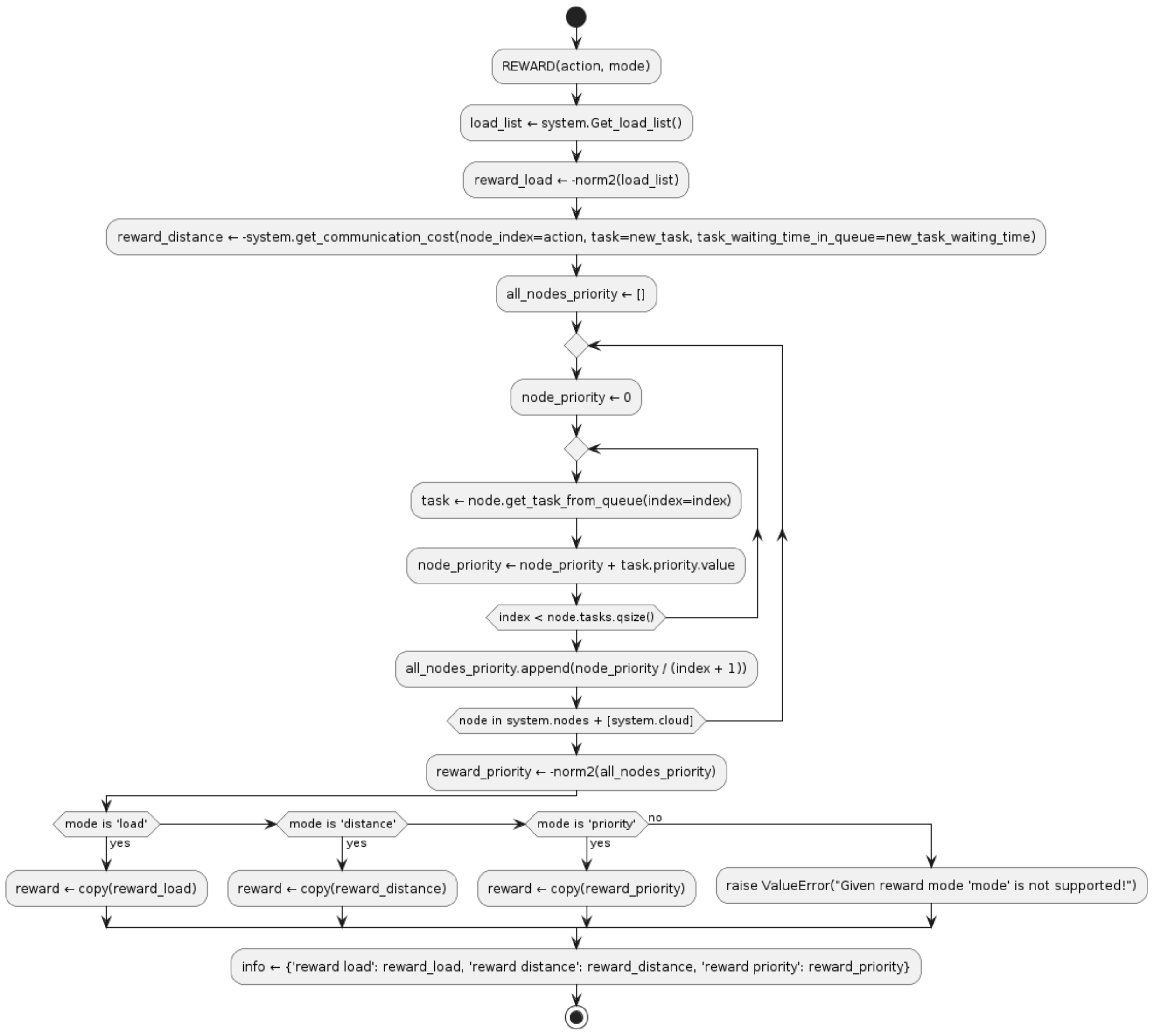

REWARD(action, mode)
load_list ← system.Get_load_list()
reward_load ← -norm2(load_list)
reward_distance ← -system.get_communication_cost(node_index=action, task=new_task, task_waiting_time_in_queue=new_task_waiting_time)
all_nodes_priority ← []
node_priority ← 0
task ← node.get_task_from_queue(index=index)
node_priority ← node_priority + task.priority.value
index < node.tasks.qsize()
all_nodes_priority.append(node_priority / (index + 1))
node in system.nodes + [system.cloud]
reward_priority ← -norm2(all_nodes_priority)
mode is 'load'
yes
mode is 'distance'
yes
mode is 'priority'
yes
no
reward ← copy(reward_load)
reward ← copy(reward_distance)
reward ← copy(reward_priority)
raise ValueError("Given reward mode 'mode' is not supported!")
info ← {'reward load': reward_load, 'reward distance': reward_distance, 'reward priority': reward_priority}

Figure 3. 4: *Reward Function Process.*

3.5 PROPOSED ALGORITHM: DQN + MULTI-OBJECTIVE OPTIMISATION

Proposing the employment of MODRL to select nodes for task processing (Fog nodes or Cloud) based on three objectives:

- Node current Load: The processing time is improved by selecting the least-loaded node.
- Node Distance: Improves the communication time, which depends on the Distance between the selected node and the amount of traffic on the link.
- Task Priority: Improve the scheduling time by selecting the node that has fewer Priority tasks in their queue (i.e., selecting nodes with less load is not enough, because even unloaded nodes may have Priority tasks).

Deep Reinforcement Learning contains training DNN to make serial results in an environment to maximize a reward function. DRL is probably employed to create dynamic and adaptive scheduling decisions in task processing. Hence, employ three DRL agents, one for each objective. However, this is a more challenging scenario because there is a trade-off among these objectives, and eventually, each agent may select a different processing node according to its own objective, which brings us to a Pareto Front problem.

DQN algorithm with the epsilon-greedy algorithm was employed to solve the provided environment. The DQN is a popular Reinforcement-learning algorithm that combines Deep Neural Networks with the Q-Learning algorithm to handle high-dimensional state and action spaces. In this setting, trained three separate DQN agents, each with a different reward mode: Load, Distance, and Priority.

The epsilon-greedy strategy was used to balance exploration and exploitation during training. At each step, the agent chooses either to exploit the current knowledge by selecting the action with the highest estimated Q-value or to explore the environment by selecting a random action. Initially, the exploration rate is high, allowing the agent to explore various actions and states; however, over time, it gradually decays to favor the exploitation of learned knowledge, as shown in Figure 3.4.

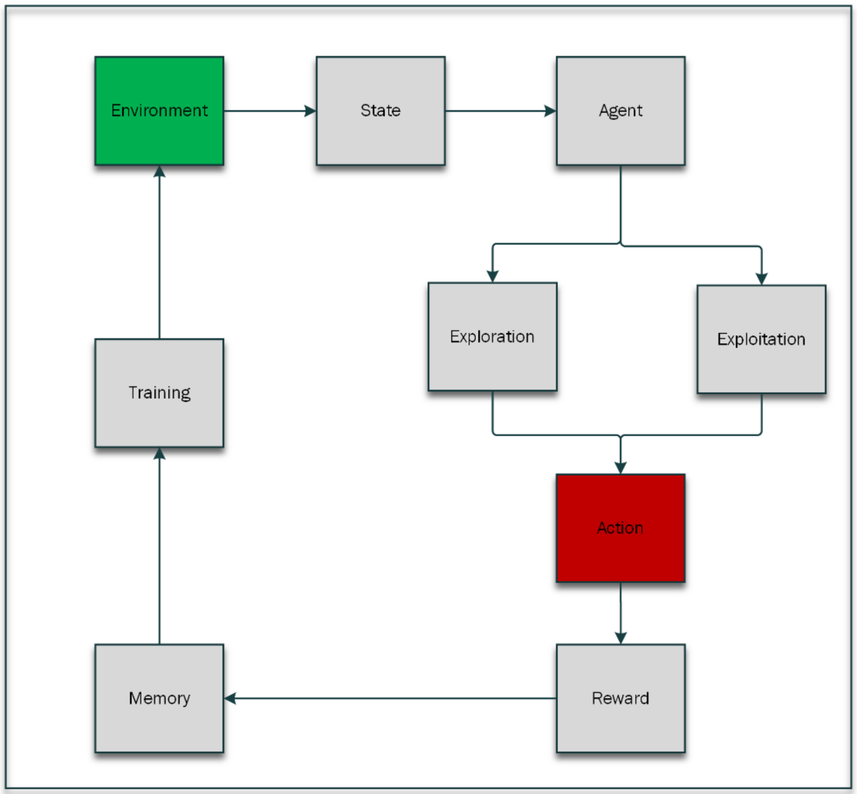


Figure 3. 4: *Illustration of Deep Reinforcement Learning Environment. It depicts the fundamental constituents of the environment, comprising the state, agent, and action. The exploration and exploitation phases are illustrated as a means of achieving a balance between discovering new methods and utilizing existing ones, with the agent and action serving as key components. After the action is taken, the subsequent reward is stored in memory. The iterative process of storing experiences in memory and subsequent training enhances the learning and decision-making abilities of the reinforcement learning algorithm*.

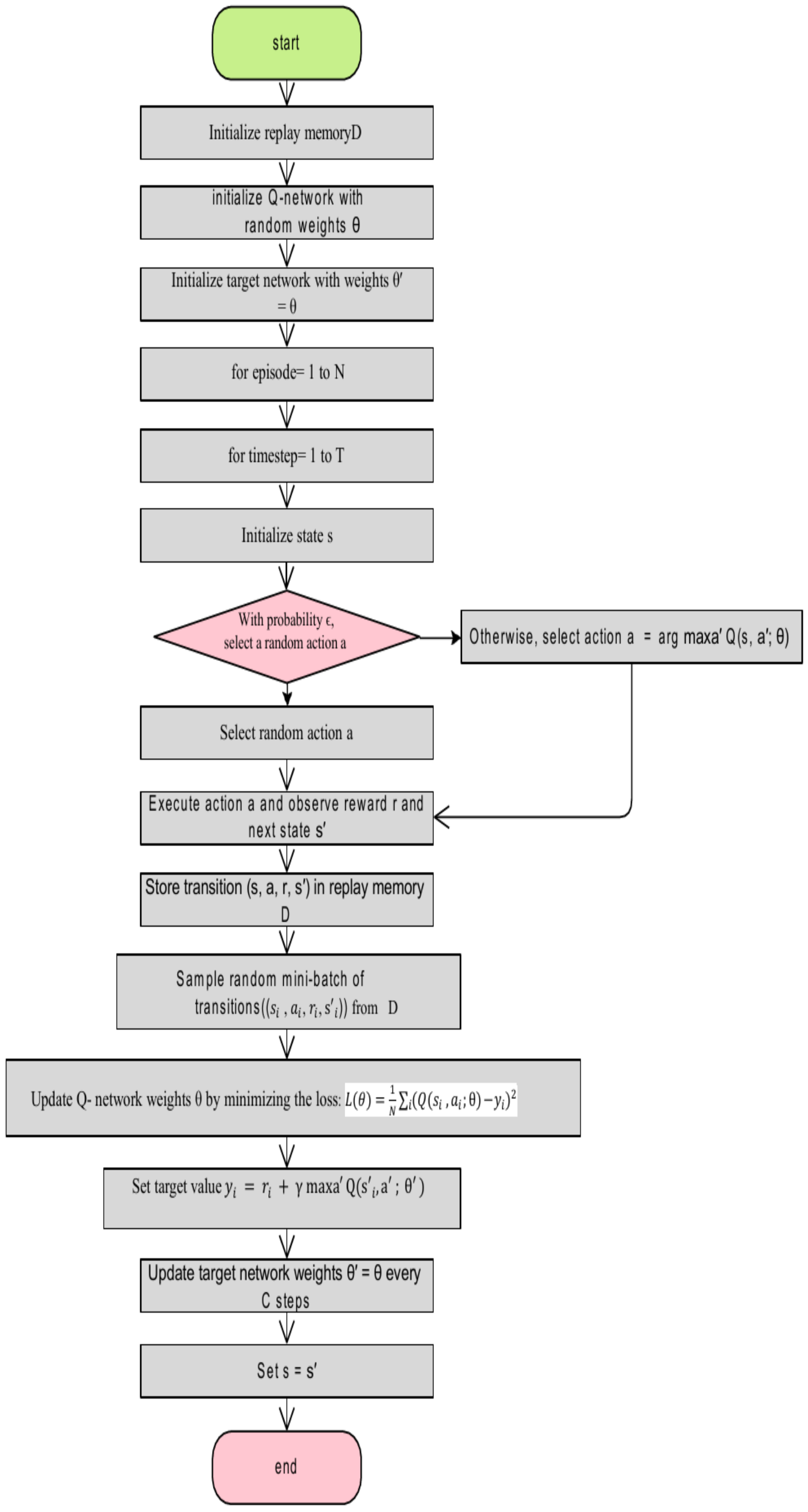

start
Initialize replay memoryD
initialize Q-network with random weights θ
Initialize target network with weights θ′ = θ
for episode= 1 to N
for timestep= 1 to T
Initialize state s
With probability ϵ, select a random action a
Otherwise, select action a = arg maxa′ Q(s, a′; θ)
Select random action a
Execute action a and observe reward r and next state s′
Store transition (s, a, r, s′) in replay memory D
Sample random mini-batch of transitions((s_i, a_i, r_i, s′_i)) from D
Update Q- network weights θ by minimizing the loss: L(θ) = 1/N Σ_i(Q(s_i, a_i; θ) − y_i)²
Set target value y_i = r_i + γ maxa′ Q(s′_i, a′ ; θ′)
Update target network weights θ′ = θ every C steps
Set s = s′
end

Figure 3. 5: *Proposed ALGORITHM/ DQN Flowchart*

To train the DQN agents, utilize the step function provided by the environment. At each step, the agent selects an action based on its current state, and the environment returns the next state, reward, and additional information. The reward function was customized based on the reward mode, with Load, Distance, and Priority as the three different modes for our agents. DQN agents learn to optimize their actions by updating the Q-values using a combination of discounted future rewards and the Q-Learning update rule.

The proposed ALGORITHM/ DQN Flowchart outlines the main steps of the DQN algorithm. It initializes the replay memory, Q-Network, and target network. It then iterates over episodes, and within each episode, it iterates over time steps. At each time step, the algorithm selects an action based on an epsilon-greedy strategy, executes the action, and stores the transition in replay memory. Then, it samples a mini-batch from the replay memory and performs a Q-Learning update to train the Q-Network. The target network was periodically updated to stabilize learning. The algorithm continues until a specified number of episodes is reached.

After running the algorithm, each DRL measures the quality Q of each possible action that corresponds to the processing; thus, the vector of the nodes is sorted from high-quality action to low; in other words, the best nodes of each DRL are those with the highest Q.

Because each DRL may have a different sorting of the nodes, it is necessary to choose the optimal node that can provide the best possible results for all three objectives. Multi-Objective Optimization of the FN service placement problem, considering a replication level, scalable service, and using pure Multi-Objective genetic algorithms. For example, MOEA/D or NSGA-II. Tested NGSA-II and

MOEAD, which are Multi-Objective Evolutionary Algorithms that aim to find the optimal node for task processing based on objectives. As shown in figure 3. 6.

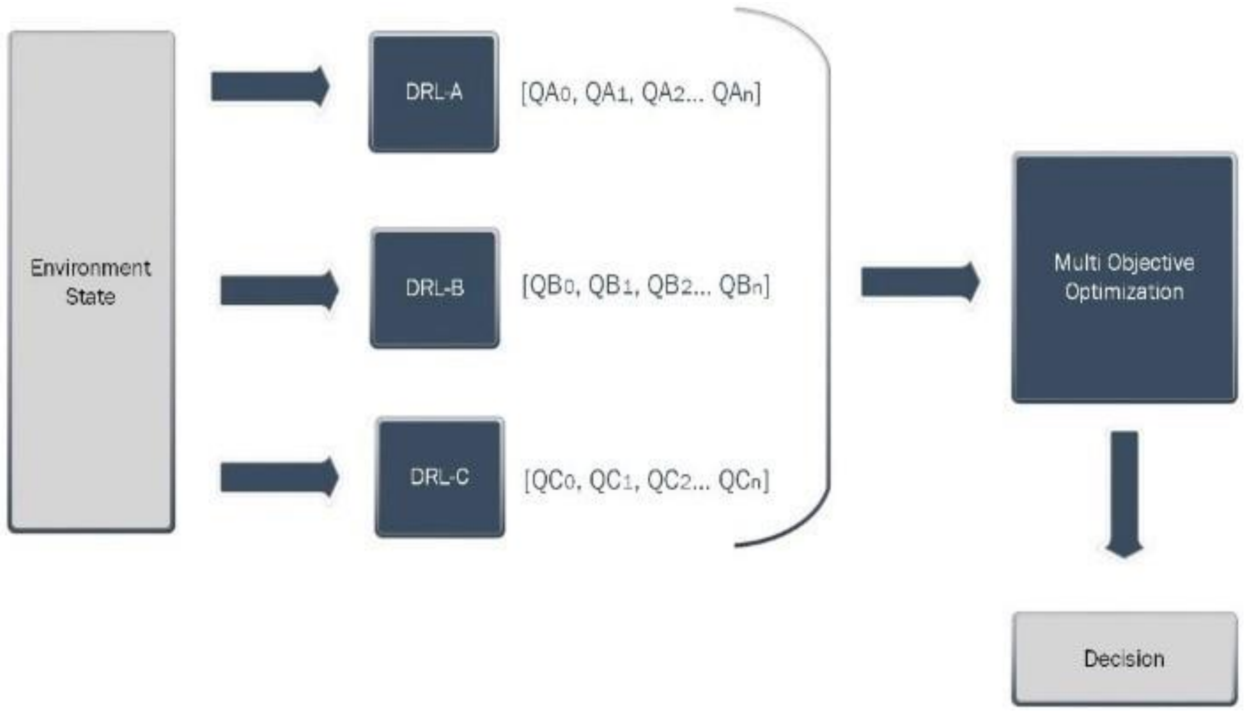


Figure 3. *6*: Proposed (DQN + MULTI-OBJECTIVE OPTIMISATION) System Environment.

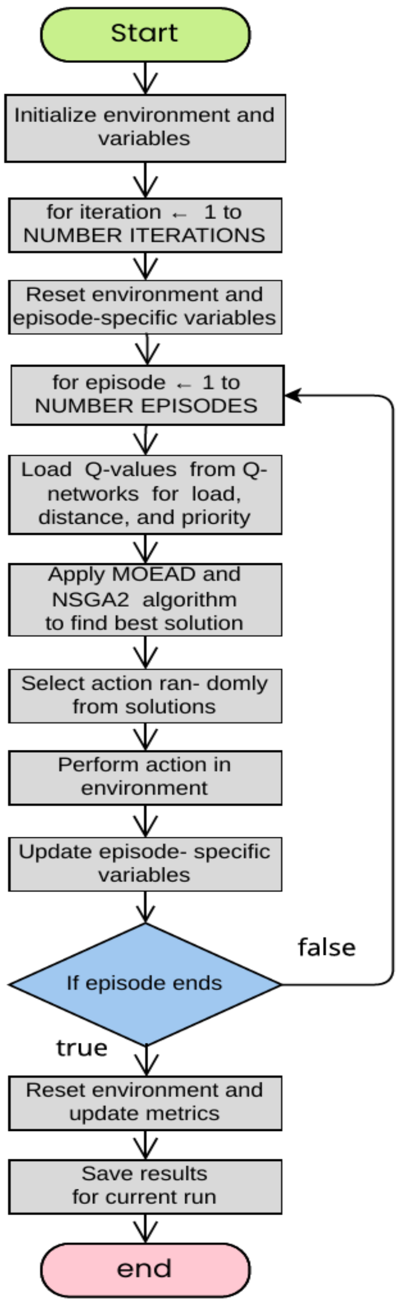

Start
Initialize environment and variables
for iteration ← 1 to NUMBER ITERATIONS
Reset environment and episode-specific variables
for episode ← 1 to NUMBER EPISODES
Load Q-values from Q-networks for load, distance, and priority
Apply MOEAD and NSGA2 algorithm to find best solution
Select action ran- domly from solutions
Perform action in environment
Update episode- specific variables
If episode ends
false
true
Reset environment and update metrics
Save results for current run
end

FIGURE 3. 7: *Proposed Multi-Objective Optimization using (DQN + MOEA/D) and (DQN + NSGA2) Flowchart.*

The main loop iterates over a fixed number of iterations; within each iteration, an episode loop runs for a fixed number of episodes. At the start of each episode, the environment was reset, and the episode-specific variables were initialized.

Inside the episode loop, the algorithm selects the best action based on the three Q-Networks using a Multi-Objective Optimization approach. The action values for each objective were obtained by passing the current state through each respective Q-Network. These values are normalized, and an instance of the Multi-Objective Optimization problem class is created. The MOEAD and NSGA2 algorithms were then used to find the best solution (action) for the given objectives. The selected action was chosen randomly from the solutions obtained by the MOEAD and NSGA2 algorithms. The combination of DQN agents with the MOEAD and NSGA2 algorithm allows the exploration of different trade-off solutions on the Pareto Front of the Multi-Objective Optimization problem, ensuring that the Intelligent Scheduling Algorithm finds solutions that align with the preferences and constraints of decision-makers as well as balance the conflicting objectives efficiently. DQN agents offer perceptions into the trade-offs between conflicting objectives (Load, Distance, and Priority), enabling more informed selection and exploration of results on the Pareto Front. Thus, the proposed Intelligent Scheduling Algorithm (DQN+ Multi-Objective Optimization) handles trade-offs between objectives by utilizing the RL capabilities of DQN agents to learn real scheduling policies and integrating them with Multi-Objective Evolutionary Algorithms.

3.6 SIMULATION EXPERIMENT

All simulation experiments are implemented in a Python environment with TensorFlow, PyTorch, Pymoo, and PQDM libraries in PyCharm, which is a powerful Python IDE. To simulate and train the Intelligent Scheduling Strategy, OpenAI Gym is utilized. A computer with a 5.0 GHz Intel Core i7 GPU and 16 GB of ARM has been used. Virtualized data using MatPlotLib in the Jupyter Notebook. The simulation environment contains 250 nodes and 1000 steps per episode for 400 tasks. Random seeds were fixed for each experiment to ensure reproducibility. The Neural Network architecture consisted of three fully connected layers with ReLU activation. The Q-Network takes the state as an input and outputs Q-values for each action. The replay buffer was used to store and sample transitions to train the Q-Network. It has a capacity of 500,000 and stores tuples of (state, next state, action, reward, and done). The buffer allows efficient sampling of mini batches for training. Simulation hyperparameters that are used in the simulation environment are listed in table 3.1, DQN hyperparameters in Table 3.2, MOEAD hyperparameters in table 3.3, and NSGA2 hyperparameters in table 3.4. Thus, the dataset is created with the environment hyperparameters that define the state, action, and reward spaces of the MODRL environment. The dataset serves as the foundation for training and evaluating the MODRL algorithm.

1. ENVIRONMENT HYPERPARAMETERS

Table 3. 1*: Environment Hyperparameter*

Parameter	**value**
Task	
- task size (min, max)	1-10
- task cpu req (min, max)	25-500
IoT Layer	
- iot layer average rate (λ)	3
Computing Node	
- node cpu freq (min, max) (cloud)	10-50(250)
- node max load (min, max) (cloud)	10-100(5000)
Communication Model	
- comm channel bandwidth (W)	1
- comm transmission power (Pi)	1
- comm channel gain (Hi)	1
- comm bandwidth (Bk)	1
- comm N0	1
Orchestrator	
- Orchestrator number of timesteps	100-1000

2. DQN HYPERPARAMETERS

Table 3. 2 *:DQN Hyperparameters*

Hyperparameter	**Value**
Discount factor (γ)	0.99
Evaluation frequency	Every 2 episodes
Batch size	256
Replay buffer capacity	500,000
Target network update frequency	Every 100 episodes
Initial epsilon	1
Epsilon decrease parameter	10,000
Minimum epsilon	0.2
Epsilon decrease start step	150,000

Stop exploring episode	5,000
Number of training episodes	7,500
Number of iterations	1
Learning rate	0.001
Number of nodes	250
Maximum timesteps per episode	1,000

3. MOEAD HYPERPARAMETERS

TABLE 3. 3: MOEA/D HYPERPARAMETERS

Parameter	**Value**
Population Size	20
Sampling	Integer Random Sampling
Crossover	SBX (Simulated Binary Crossover)
Mutation	Polynomial Mutation
Number of Neighbors	20
Prob. Neighbor Mating	0.7

4. NSGA2 HYPERPARAMETERS

TABLE 3. 4: *NSGA2 HYPERPARAMETRS*

Parameter	**Value**
Population Size	20
Sampling	IntegerRandomSampling
Crossover	SBX (Prob=1.0, Eta=3.0, Vtype=float, Repair=RoundingRepair())
Mutation	PM (Prob=1.0, Eta=3.0, Vtype=float, Repair=RoundingRepair())
Eliminate Duplicates	True

3.7 Summary

In this chapter, the system architecture, system model, system framework, and DQN + Multi-Objective Optimization algorithm has been proposed. An Intelligent Scheduling Strategy is proposed in a Fog Computing system based on a Multi-Objective Deep Reinforcement Learning Algorithm. The proposed model will tackle two main problems: task allocation and task scheduling. The MODRL algorithm is required to develop an optimal strategy to allocate the tasks generated from IoT devices to the suitable processing nodes (Fog or Cloud), as well as schedule those tasks in the allocated Fog nodes based on multiple criteria (Load, Priority, and Distance).

CHAPTER FOUR

Experimental results

4-1 Introduction

This chapter provides a comprehensive analysis and evaluation to the proposed approach for addressing the problem at hand. Describing the specific hyperparameters and configurations used in the experiments to ensure reproducibility and consistency. Furthermore, reporting and discussing the results obtained, highlighting the performance of the DQN agents and the Multi-Objective Optimization algorithm employed in terms of the defined objectives and analyzing any interesting trends or observations. Through these experiments, aiming to gain insights into the effectiveness and trade-offs of different reward functions and the impact of Multi-Objective Optimization in addressing the complexities of the problem.

In the context of evaluating and validating the proposed Intelligent Scheduling Strategies, performance, efficiency, and adaptability metrics such as Task Completion Time, QueueingDelay, Makespan, CPU Load, Storage Capacity, Latency, Computational Delay, Propagation Delay, Processing Delay, and Transmission Delay, Throughput and Network Congestion were used as benchmarks. These metrics are crucial for assessing its efficiency, effectiveness, and adaptability, as well as serving as key indicators of how well the scheduling strategy performs in Fog Computing based on MODRL

4.2 Training Process/ Training Environment

The deployment of MODRL for Fog Computing scheduling entails training a DQN to navigate the complex decision space. The scheduling is based on three objectives: Load, Distance, and Priority. The DQN's purpose is to train a policy that efficiently assigns jobs to Fog nodes, taking into account the trade-offs associated with the three objectives. Throughout the training process, the agent engages with the environment, utilizing its existing policy to make judgments and earning incentives as feedback. The rewards are obtained based on the performance in terms of load, Distance, and Priority, which encompasses the Multi-Objective characteristic of the scheduling problem. While the agent explores the decision space, the parameters of the DQN are continuously updated by methods like experience replay and target network updates.

The objective of the learning process is to identify a strategy that effectively manages the computational burden on Fog nodes, minimizes the distance between tasks and Fog nodes, and meets priority restrictions. The three plots function as visual representations, demonstrating the agent's policy development over time in order to negotiate the intricate terrain of scheduling choices. They offer valuable insights into the compromises and collaborations between the conflicting aims during the training process. The repeated improvement of the DQN's policy demonstrates the model's capacity to acquire a sophisticated and efficient scheduling approach that simultaneously optimizes Load, Distance, and Priority in Fog Computing environments, as shown in the figure 4.1:

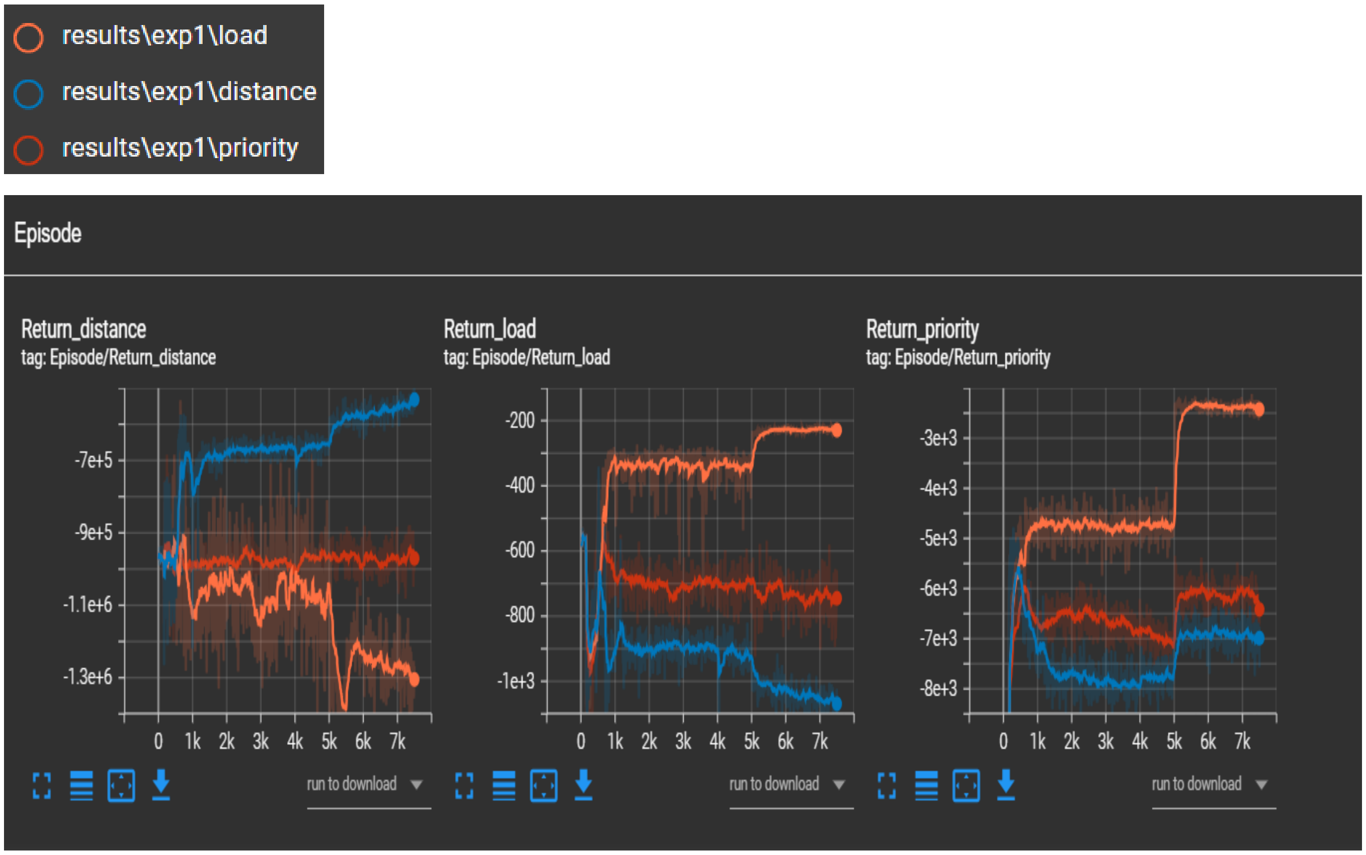


Figure 4. 1*: Episode/Return Load, Distance, and Priority*

4.3 Reward Structure

Reward calculation in the environment is based on three components: Load, Distance, and Priority. The load component penalizes high load values and favors a balanced load distribution among the nodes (the Load of a single node is computed based on the processing model introduced in section 3.3). The Distance component represents the communication cost (described by the equations in section 3.3) of sending an assigned task to the chosen node. The Priority component penalizes the average Priority of the tasks in each node queue. These objectives can include optimizing resource utilization and minimizing task execution delays, as shown in figure 4.2. Thus, the role of the reward function in the context of Deep Reinforcement Learning for scheduling is to guide the agent's decision-making process. Pareto defines the objectives of the scheduling problem, which allow shaping the agent's behavior to achieve trade-offs and specific scheduling goals, influence the agent's learning process, and enable adaptability to changing Fog environment conditions. Through reward calculation, the scheduling agent evolves and learns to make optimized, intelligent scheduling decisions. Therefore, it enhances the effectiveness and efficiency of Fog Computing systems.

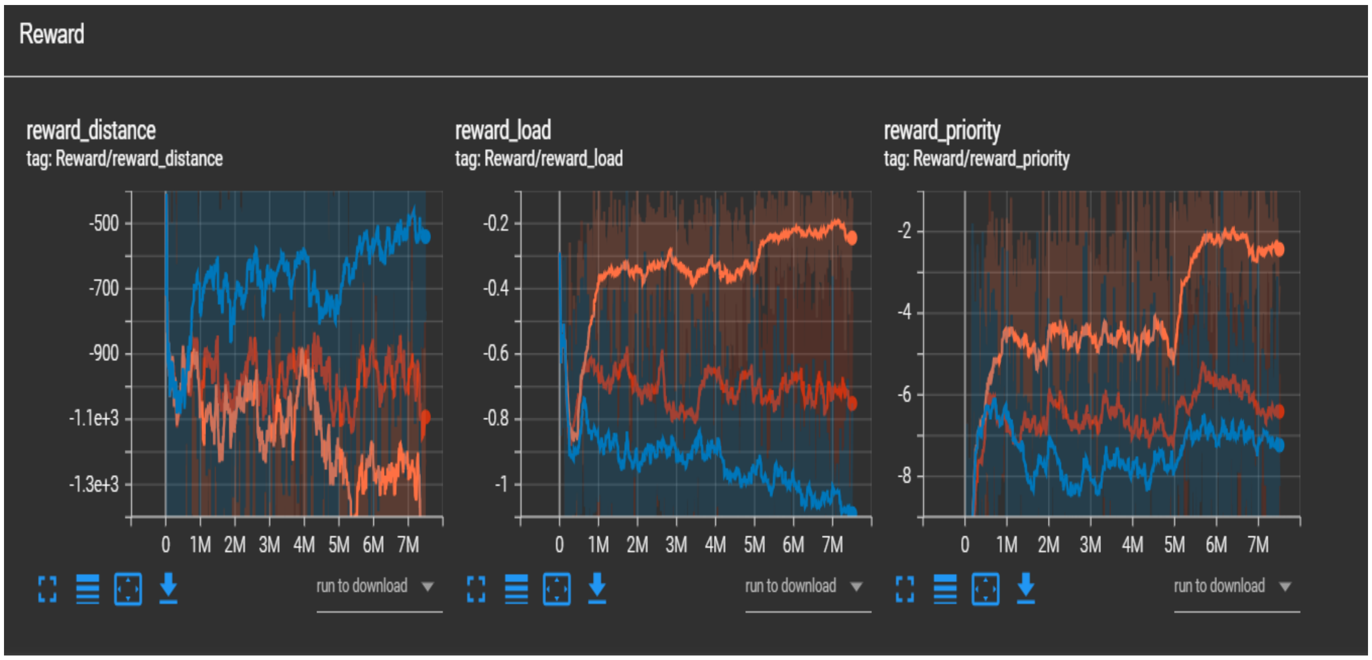


***Figure 4. 2**: Reward/Return Load, Distance, and Priority*

4.4 Exploration/Explotation

MODRL for Fog Computing uses a dynamic strategy to transition between exploration and exploitation. The strategy involves setting a parameter, epsilon (ε), to 1 during the exploration phase, allowing the agent to explore various scheduling decisions. As ε decreases, the agent shifts towards exploitation, prioritizing actions with high rewards based on learned Q-values. This adaptive strategy ensures the MODRL-based scheduler remains flexible and responsive to changes in the environment, combining the benefits of exploration with the efficiency of exploitation.

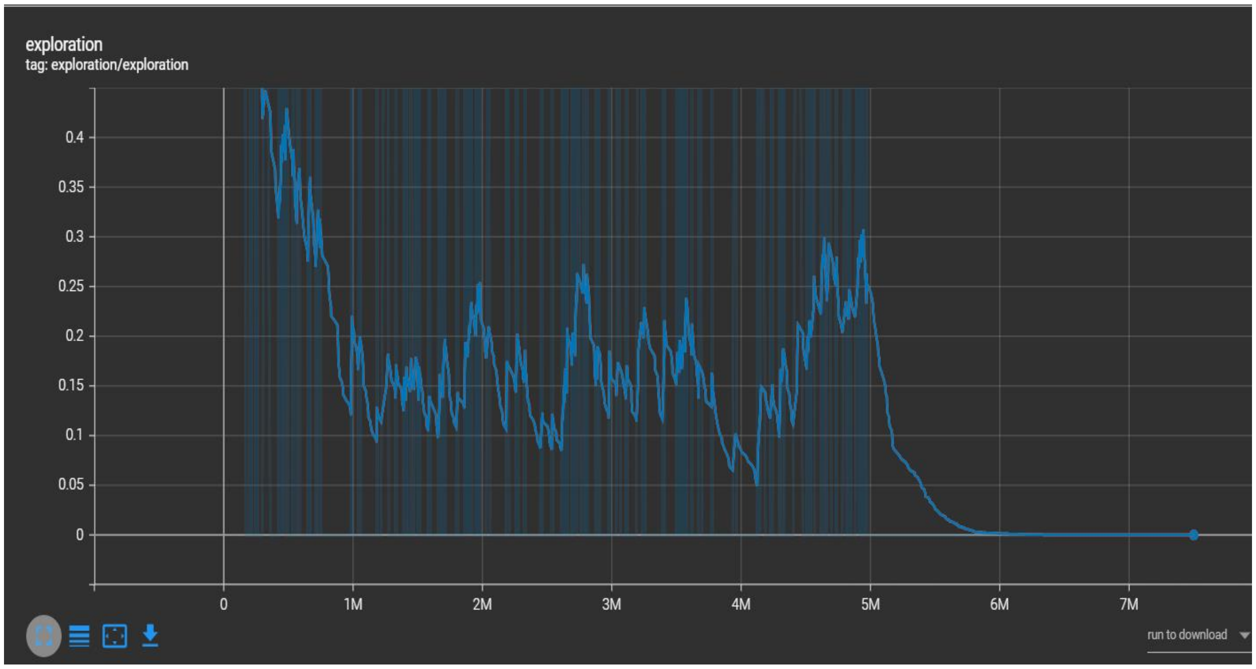


***Figure 4. 3**: Exploration/Explotation*

The consistency in plot outcomes can be attributed to the scheduling algorithm converging around a specific solution that ideally fulfills all three objectives. This convergence indicates that the algorithm has discovered a scheduling policy that efficiently distributes the workload, minimizes the distance between tasks and Fog nodes, and simultaneously meets Priority requirements.

4-5 Validating the DQN algorithm based on Load, Distance, and Priority.

Figure 4.4 shows the evaluation of DQN agents (in terms of the three learning objectives) during the training steps. The plots show three tendencies that correspond to full exploration (first 1500 steps), exploration-exploitation balancing (until 5000 steps), and finally a full exploitation phase. It can be observed from the plots that for each objective, the DQN agent that was trained specifically to optimize the objective achieved the best result, as expected. Thus, the algorithm dynamically adapts its approach to successfully manage the trade-off between exploring new potentials and exploiting identified, high-performing policies.

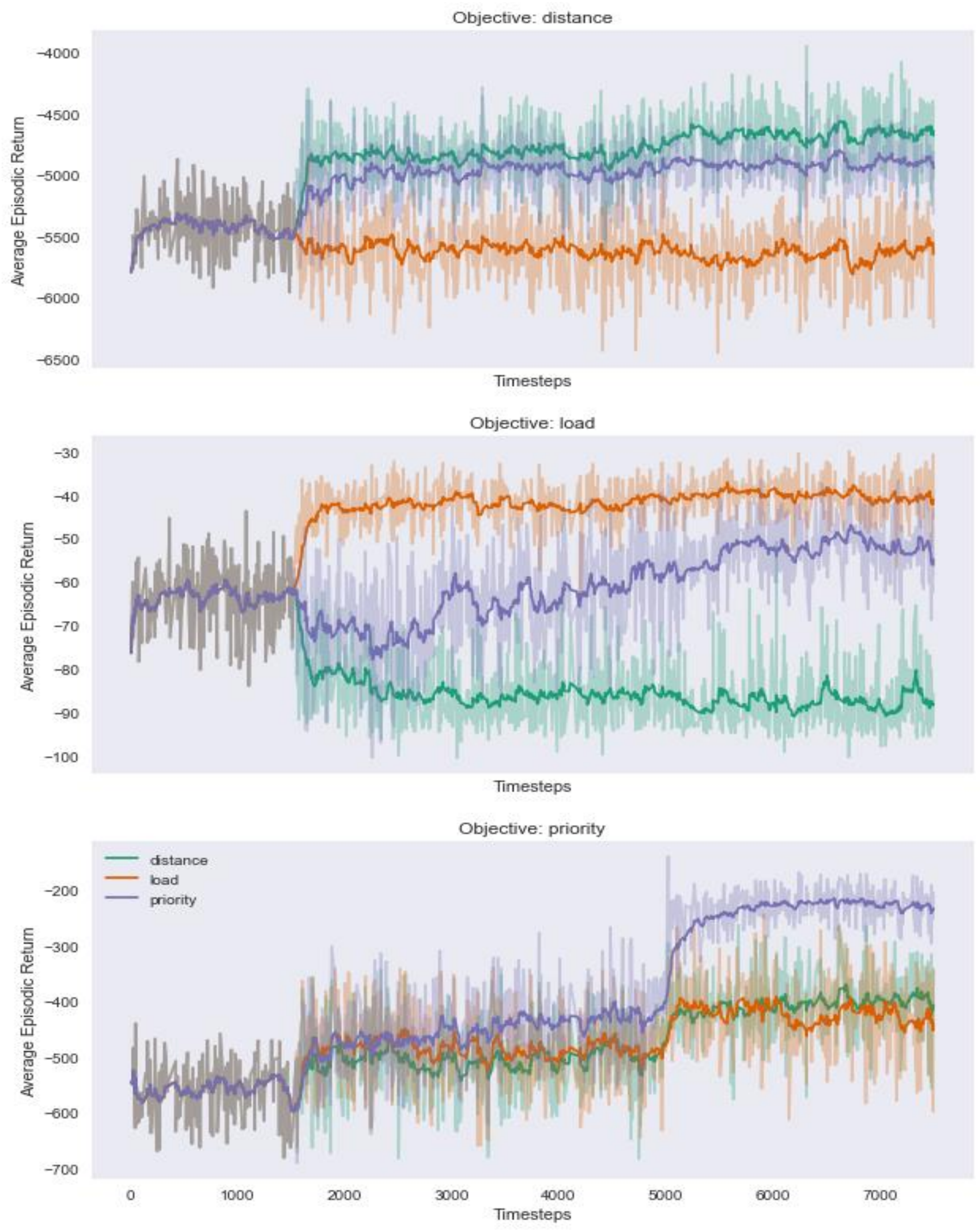


Figure 4. 4*: The average training episodic return of the three DQN algorithms (trained on three different reward functions: Load, Distance, and Priority) in terms of the three objectives: load, Distance, and Priority. (Number of nodes = 5, Number of timesteps per episode = 100)*

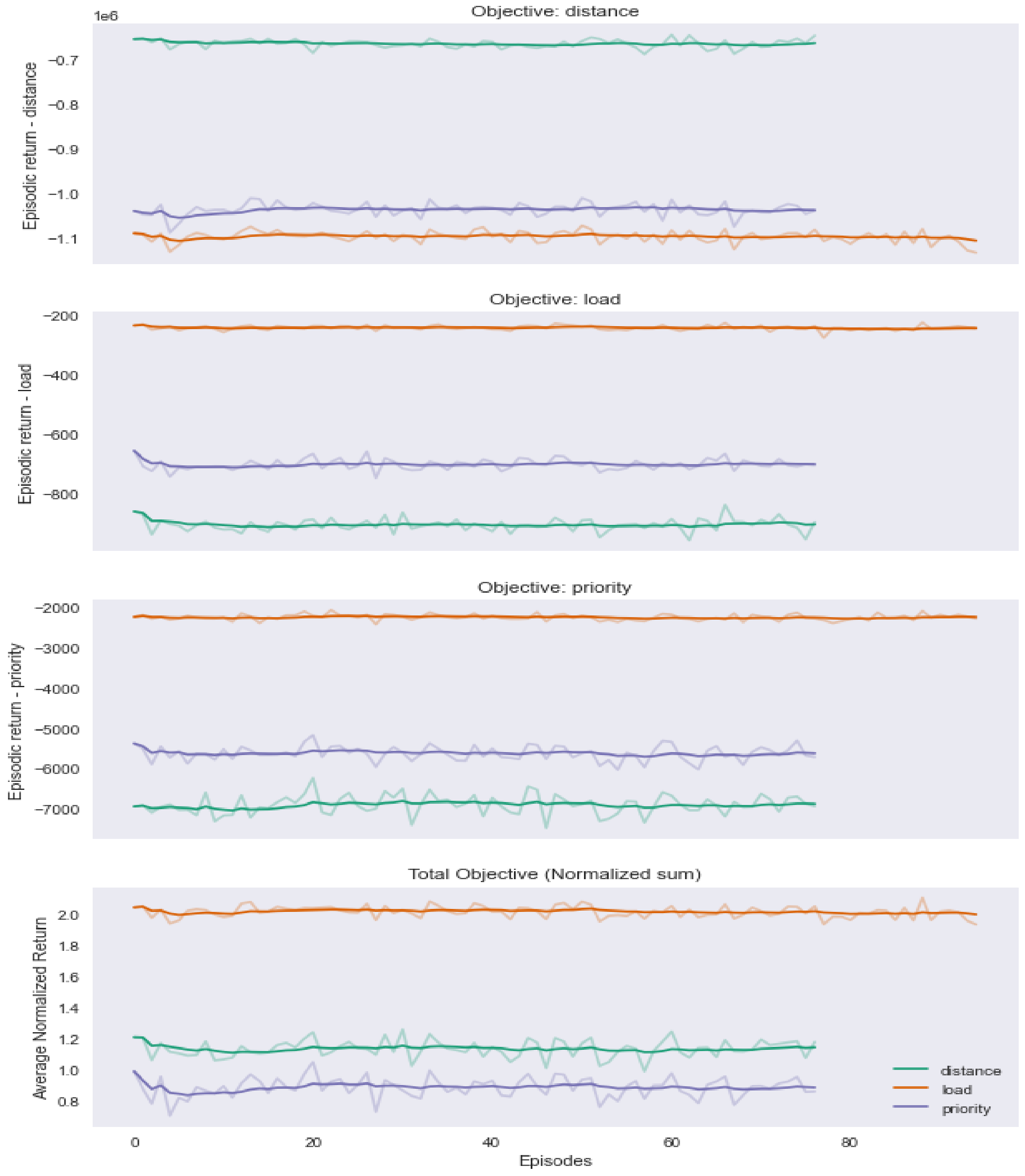


Figure 4. 5*: The average episodic return of the three DQN algorithms (trained on three different reward functions: Load, Distance, and Priority) in terms of the three objectives: Load, Distance, and Priority. The total objective is the sum of the three scores normalized between 0 and 1. (Number of nodes = 250, Number of timesteps per episode = 1000)*

Figure 4.5 shows the DQN evaluation for 250 nodes and 100 steps per episode. The DQN algorithm trained on the Load reward function achieved moderate performance in load balancing, whereas its performance in minimizing task distance and prioritizing was relatively lower. The algorithm trained on the Distance reward function excelled at minimizing the task distance but struggled with load balancing and prioritization. Finally, the DQN algorithm trained on the Priority reward function showed moderate performance in load balancing and task distance minimization but had challenges in prioritizing tasks.

The unexpected finding that the DQN trained on the Load reward function scored higher in the Priority objective compared to the DQN trained specifically for the Priority reward function raises interesting insights. One possible explanation could be that the Load reward function indirectly encourages prioritization by incentivizing the agent to allocate resources efficiently and balance the workload across nodes. As a result, the DQN trained on the Load reward function may have learned to implicitly consider task priorities when optimizing load balancing.

However, the DQN trained explicitly on the Priority reward function might have focused primarily on maximizing the Priority objective without sufficiently accounting for load balancing and task distance. This specialization on Priority might have led to sub-optimal load-balancing decisions, resulting in lower overall performance in the Priority objective compared to the DQN trained on the Load reward function.

These results highlight the trade-offs and varying performance of the three DQN algorithms. The findings emphasize the importance of carefully selecting and balancing the reward functions when training DQN algorithms for Multi-Objective Optimization tasks, as different reward functions can lead to distinct trade-offs in performance across the Load, Distance, and Priority objectives.

4-6 The multi-objective Evolutionary algorithm (MOEAD and NSGA2)

Figure 4.6 shows the average episodic return of the Multi-Objective Evolutionary Algorithms (MOEAD and NSGA2) in comparison with naive approaches (selecting random nodes and systematically selecting the closest free node).

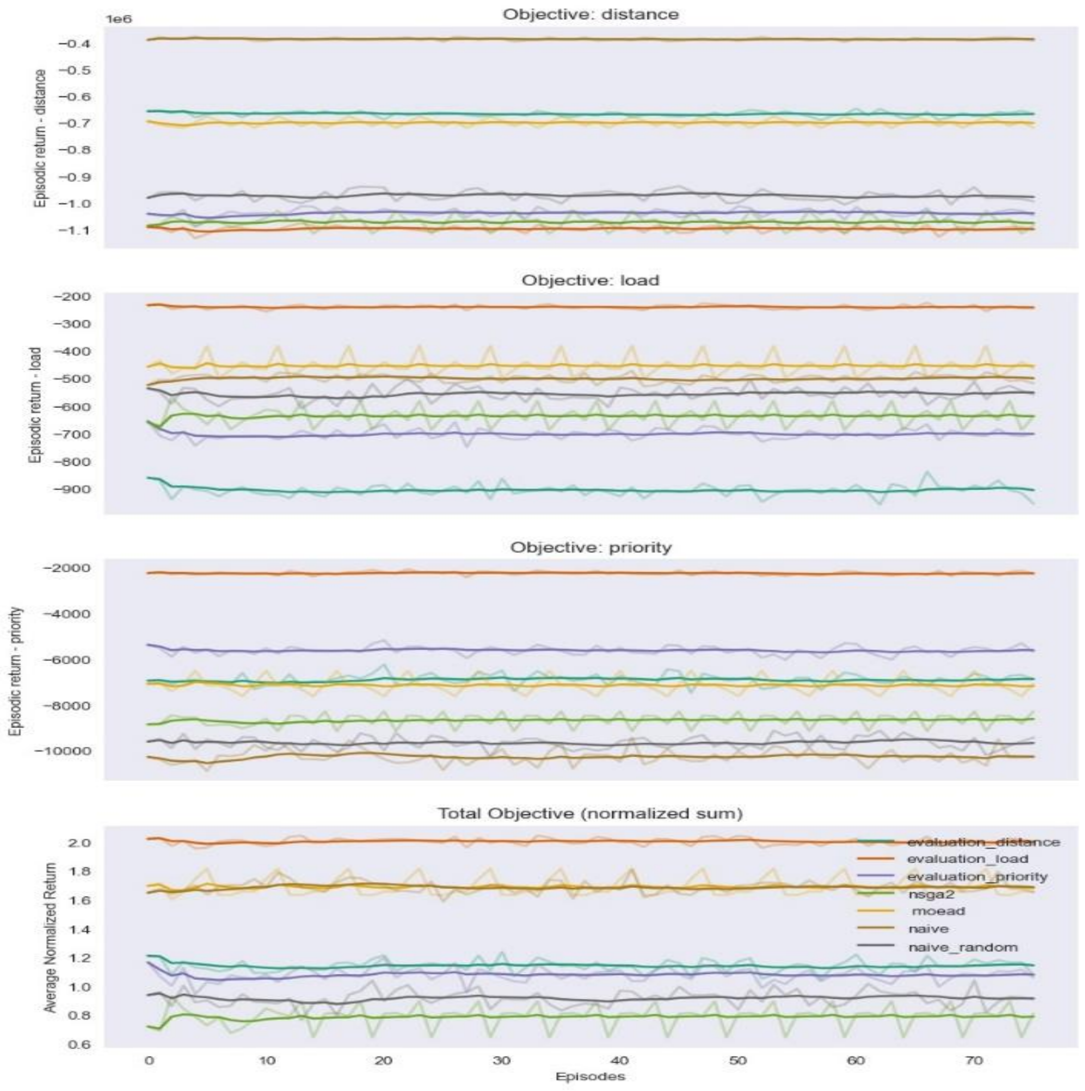


***Figure 4. 6**: The average episodic return of the Multi-Objective Evolutionary Algorithm (MOEAD and NSGA2) in comparison with Naive approaches (Selecting random nodes, and systematically selecting the closest free node). The total objective is the sum of the three scores normalized between 0 and 1. (Number of nodes = 250, Number of timesteps per episode = 1000)*

4.6 VALIDATION METRICS

In this section, the evaluation of the performance, effectiveness, and adaptability of the proposed intelligent scheduling strategy with other existing scheduling strategies has been conducted. That is based on the most important performance, efficiency, and adaptability metrics for evaluating such a schedule. To evaluate the performance, emphasis will be placed on five different metrics, which are Resource Utilization (CPU Load, Storage Capacity), Latency, Throughput, and Network Congestion. To evaluate the efficiency, emphasis will be placed on three different metrics, which are Task Completion Time, Makespan Time, and Queueing Delay. To evaluate adaptability, emphasize four different metrics, which are Communication Delay, Transmission Delay, Processing Delay, and Computational Delay. These metrics provide insights into how well the strategy performs in terms of different objectives and trade-offs. In addition, these metrics provide insights into the efficiency, performance, and adaptability of the MODRL system in handling tasks within the Fog Computing environment. Monitoring these metrics helps assess the system's efficiency in task scheduling, data processing, and network management.

In order to prove the performance, effectiveness, adaptability, and QoS of the planned intelligent scheduling strategy, a thorough validation will be conducted using all of the above metrics based on DQN+NSGA2, and DQN +MOEA/D.

A. PROPOSED ALGORITHM: DQN + NSGA2

1-Resource Utilization:

Assess how effectively Fog resources are utilized. High resource utilization implies efficient use of available storage, computing, and network resources. High Resource Utilization without delays or congestion is ideal. The two main components are:

A. ***CPU Load*****:** It is calculated by getting the current Load of the CPU as a percentage of the assigned Load from the results of processing the CPU instructions. Maintaining the CPU Load below 50% whereas having low storage usage for a large number of tasks shows the MODRL is efficient and the performance of the nodes is effective as tasks are processed faster, as shown in Figure 4.7.

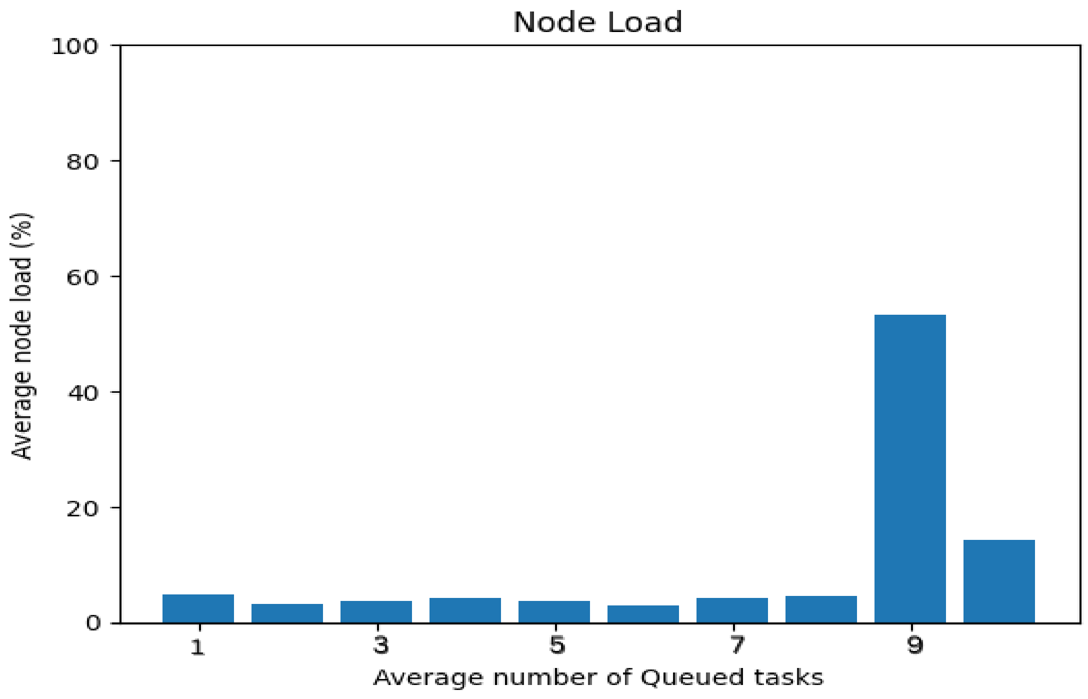


Figure 4. 7*: CPU Load*

B. ***Storage Capacity*****:** It refers to the amount of storage space available on Fog nodes or Cloud nodes for the management and storage of data. Storage Measures the remaining storage after queueingtasks. Appropriate management of storage capacity is critical for performance and optimizing the responsiveness of an Intelligent Scheduling Strategy, as shown in Figure 4.8.

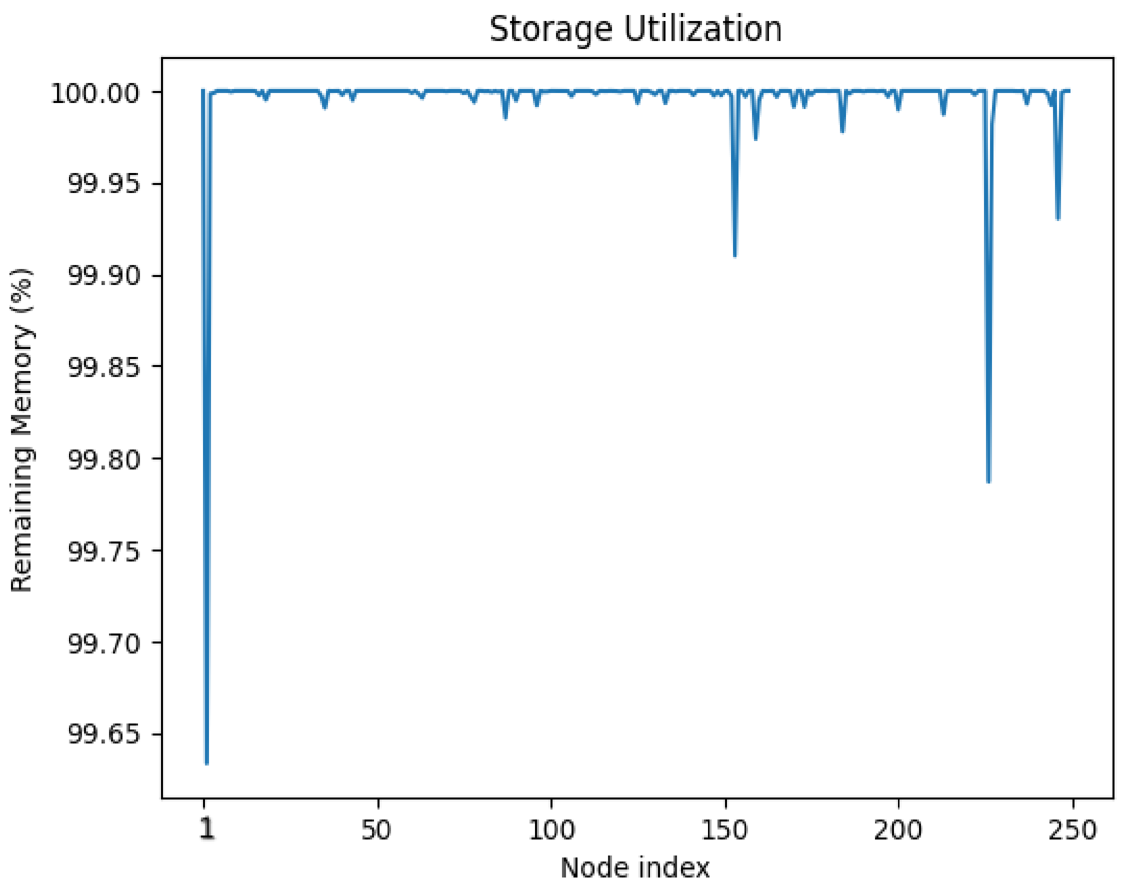


Figure 4. 8*: Storage Capacity*

2-Latency:

In Fog Computing, latency will measure the time it takes for some of the data to get to its destination over the network. Latency can be considered a measurement used for measuring delays, which is usually represented as a round-trip delay as shown in Figure 4.9. as well as Latency is often a key component of QoS requirements. Low latency is essential in Fog Computing because an Intelligent Scheduling Strategy requires rapid data processing and decision-making. The Latency was proved mathematically as follows:

$$L = \frac{TS}{R} * D \qquad (4.1)$$

Where L represents the latency, TS is the task size, R is the rate, and D is the Distance.

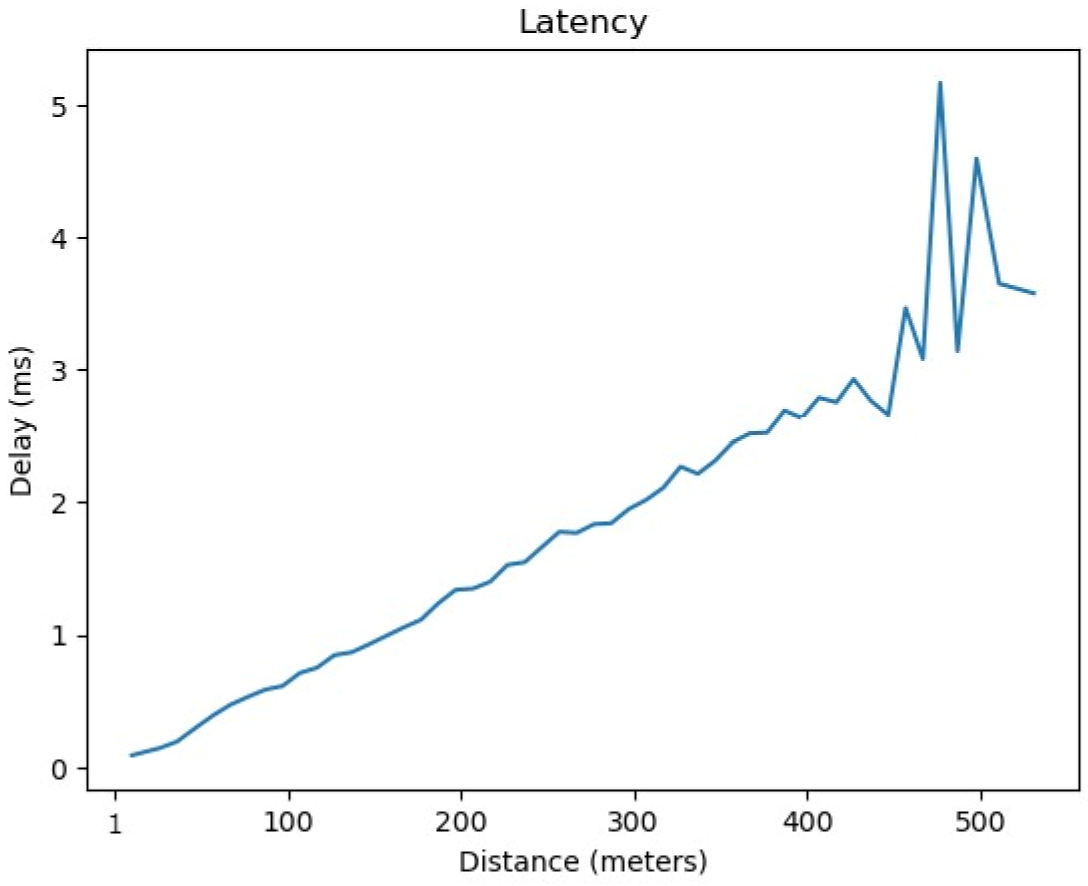


***Figure 4. 9**: Latency*

3-Task Completion Time:

Measures the time taken for tasks to be completed from their submission to the Fog system. Task Completion Times store scheduled times, with task numbers processed in Fog nodes. Figure 4.10 represents the average time for each task. Lower completion times indicate efficient scheduling. The Task Completion Time is illustrated mathematically as follows:

$$TCT = ET - RT \tag{4.2}$$

Where TCT represents the Task Completion Time, ET is the end time, and RT is the received time.

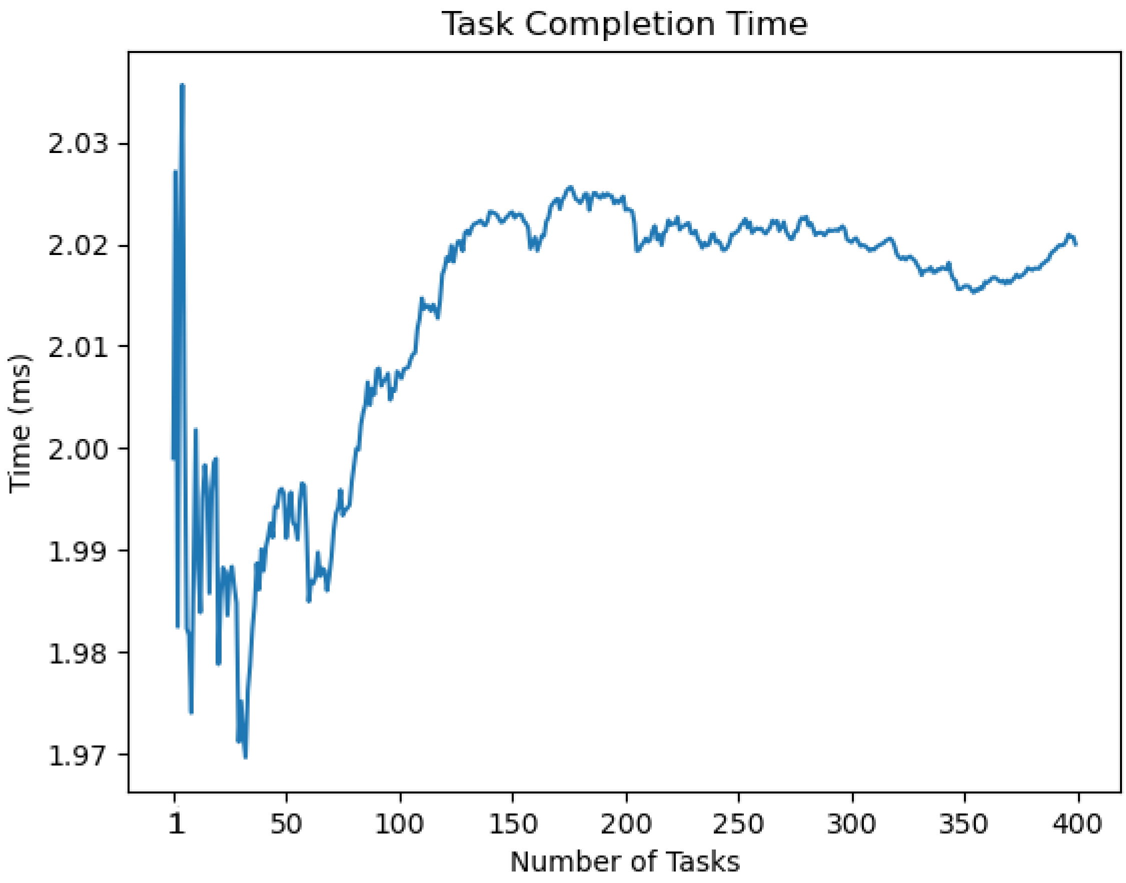


Figure 4. 10: *Task Completion Time*

4-Makespan Time:

Measures the total time taken to complete all scheduled tasks. Lower Makespan values are required. Makespan stores the total time that each node takes to compute assigned tasks. Figure 4.11 displays the average Makespan time taken by each node to process tasks during the simulation. Lower Makespan indicates efficient task completion. The Makespan time is expressed as follows:

$$MT = ET - ST \tag{4.3}$$

Where MT denotes the Makespan time, ET is the end time (empty ques), and ST is the (start of processing the ques).

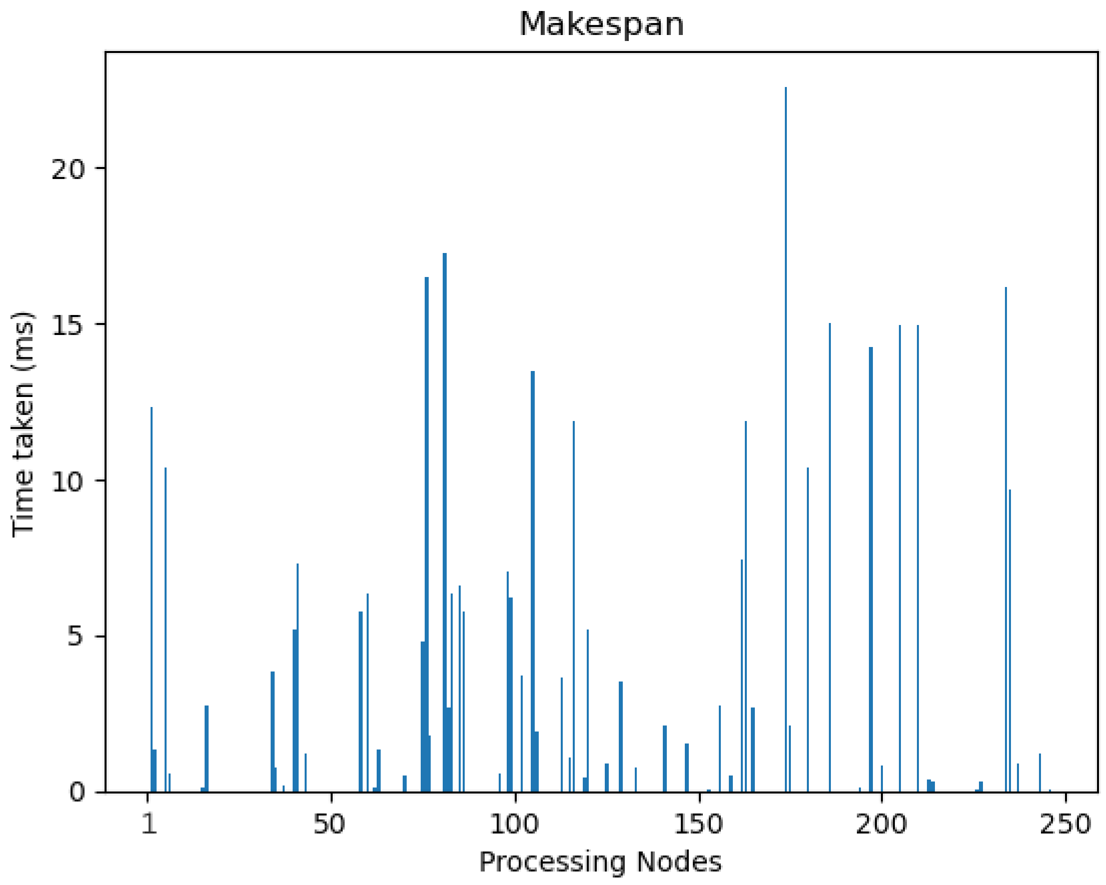


Figure 4. 11*: Makespan Time*

5-Queueing Delay:

Represents delays when data or tasks are queued before processing. The Orchestrator stores the total Queueing time for different Fog nodes. The scheduled time showcases Queueing times for different nodes over scheduled periods. Figure 4.12 shows Queueing times for different nodes over scheduled periods. Lower Queueing delays indicate more efficient task allocation. It can be measured as follows:

$$QD = \sum_0^m WT_T \qquad (4.4)$$

Where QD represents the Queueing delay, WT_T is the summation of waiting times for each task queued.

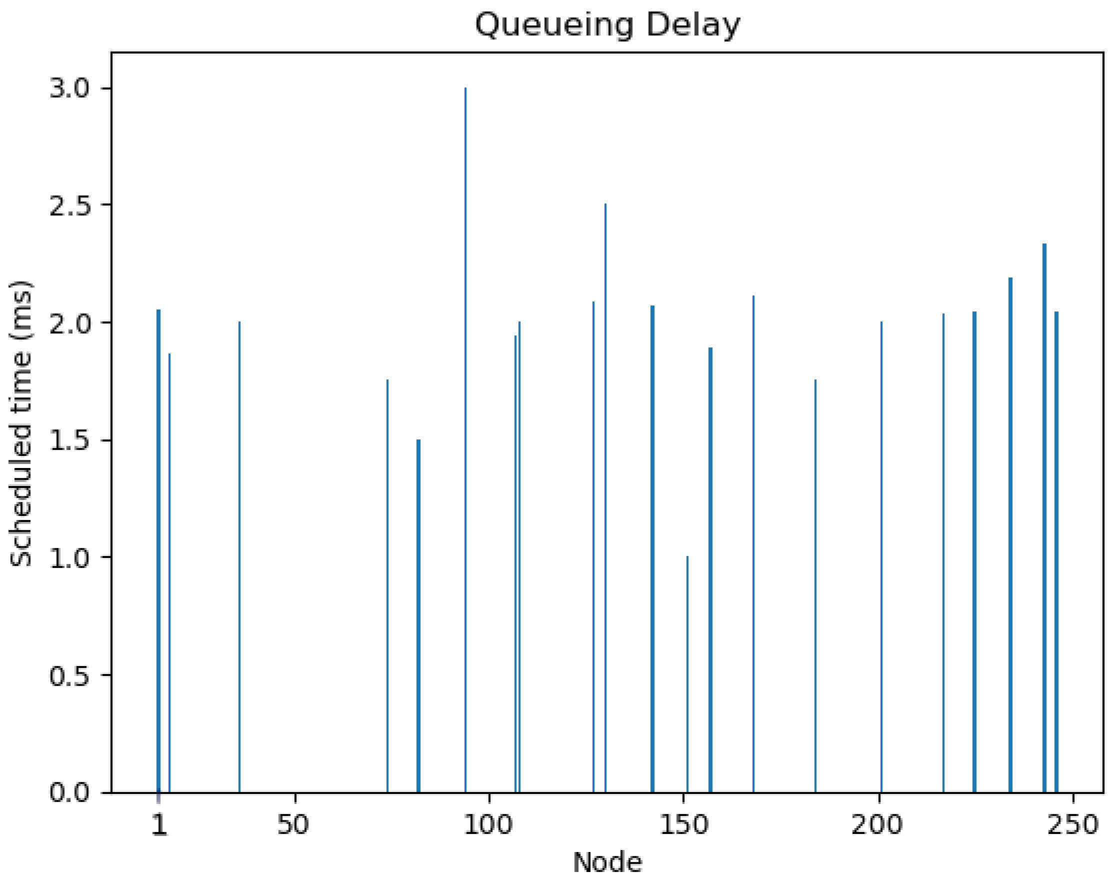


Figure 4. 12*: Queueing Delay*

6- Communication Delay:

Communication delays in FC refer to the delays experienced when information or data is transmitted between nodes or sensors within a Fog Computing system. Lower delays indicate faster data processing and allocation. Communication Delay includes:

A. ***Propagation Delay*** (data travel time): It is defined as the total time a task takes from start to completion, which includes the total lifetime of the task existing in the Orchestrator, as shown in Figure 4.13.

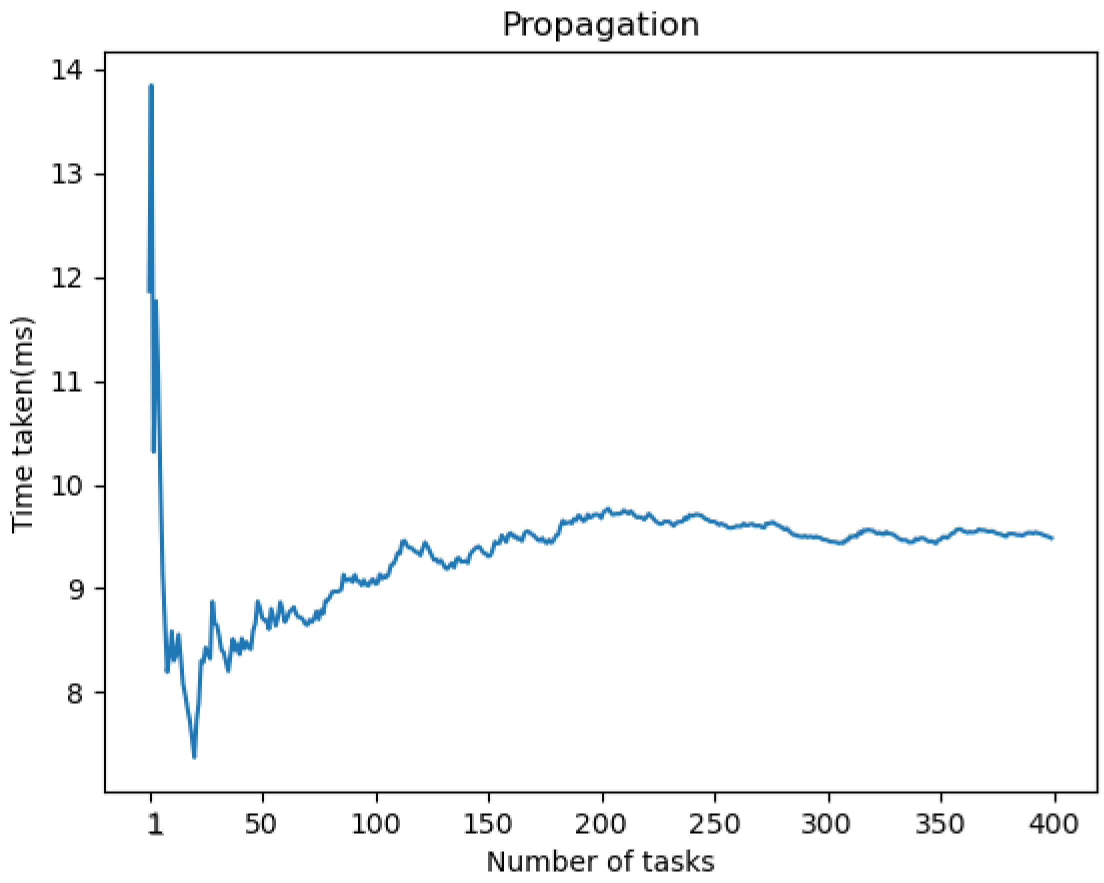


Figure 4. 13*: Propagation Delay*

B. ***Transmission Delay*** (time to send data over a link): refers to the time it takes for data to be transmitted from a source to a destination within a Fog Computing system. This delay is a serious element of end-to-end data transmission. Figure 4.14 shows the average delay for each node. The transmission delay was calculated as follows:

$$TD = TS/DR \tag{4.5}$$

Where TD signifies the Transmission Delay, TS is the task size, and DR is the data rate.

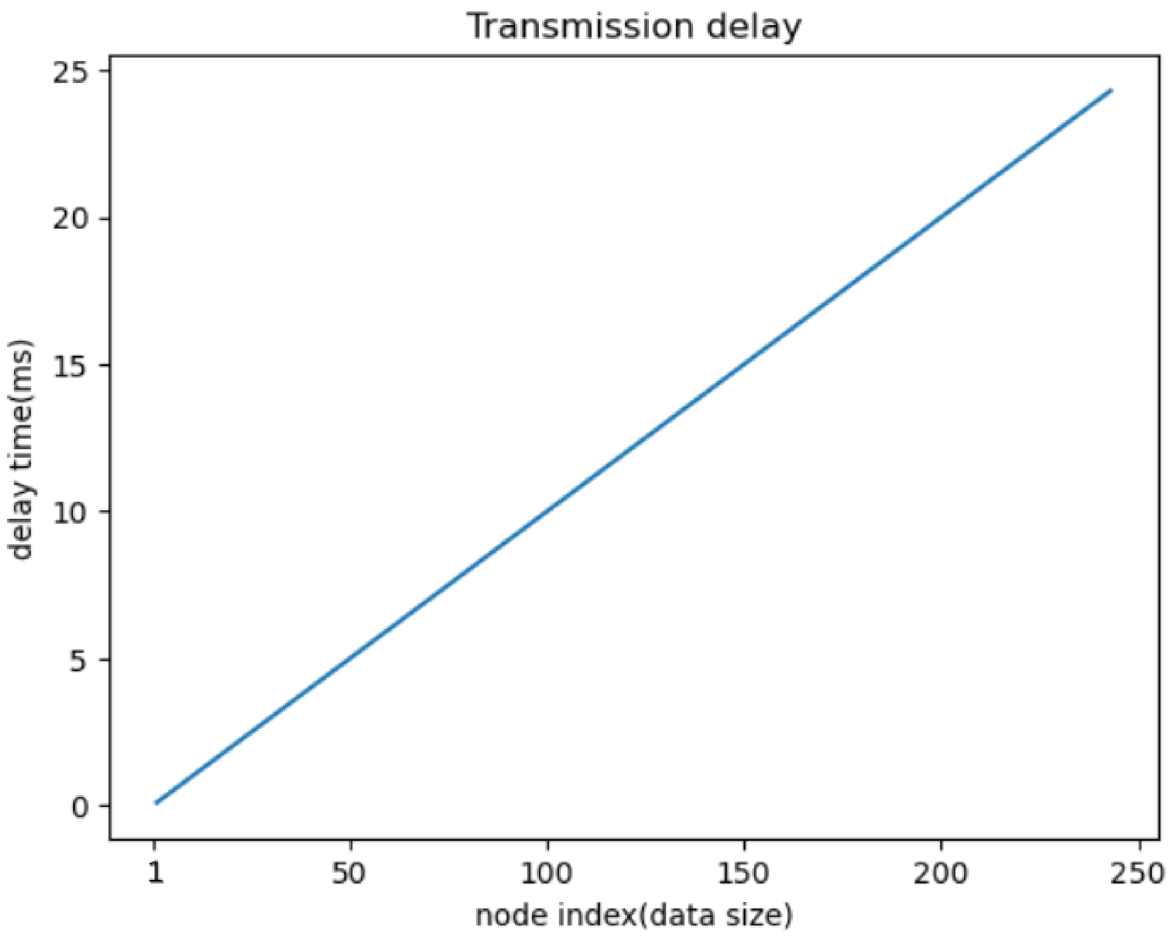


Figure 4. 14*: Transmission delay*

C. ***Processing Delay*** (time taken to process data at Fog nodes): It is the time taken to process a task in computing time, as shown in Figure 4.15. Lower delays indicate faster data processing and allocation. It can be measured as follows:

$$PD = CR/CF \tag{4.6}$$

Where PD is the Processing Delay, CR is the CPU requirements, and CF is the CPU frequency.

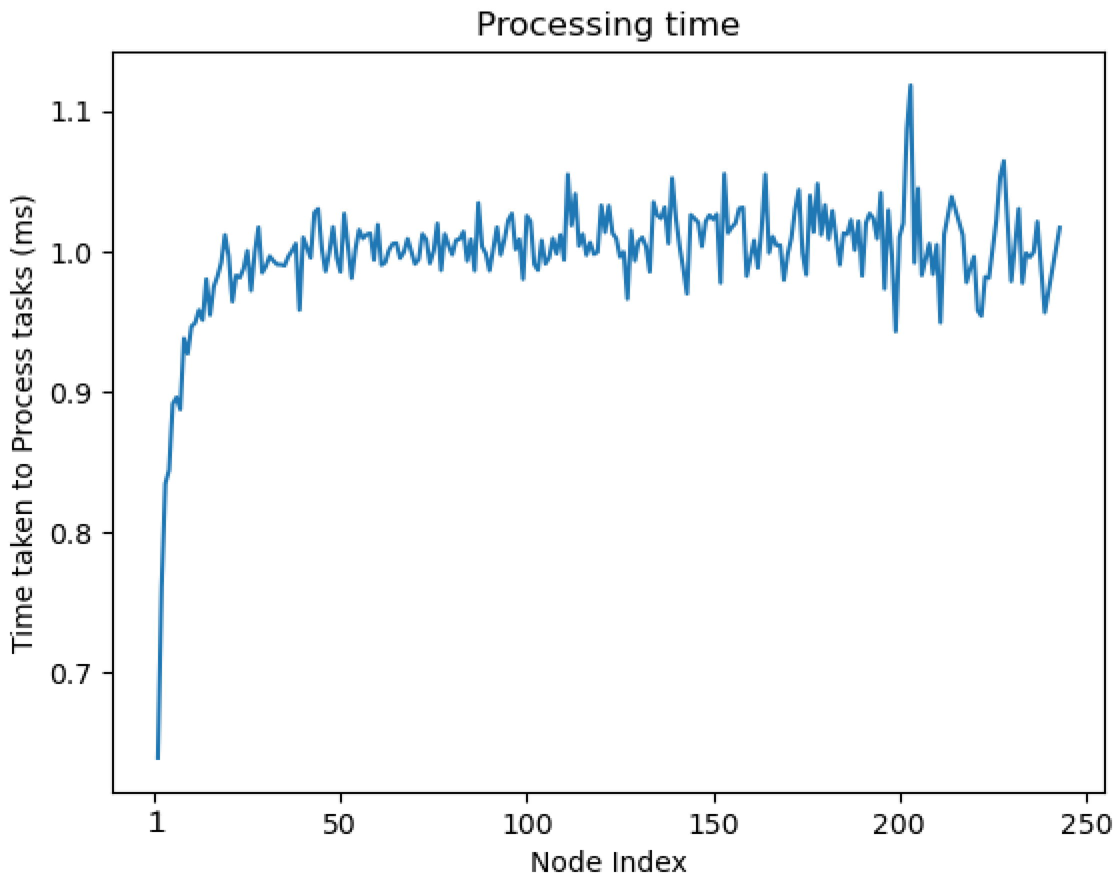


Figure 4. 15: *Processing Delay*

7-Computational Delay:

Measures the time taken to process data or perform tasks at Fog nodes. In Computation Delay, data is derived from total processing and scheduling time for tasks. Figure 4.16 displays the Computation Delay against the data size. A lower Computation Delay indicates faster task computation. Mathematically express it as follows:

$$CD = DT + SD \tag{4.7}$$

Where CD is the Computation Delay, DT is the computing time, and SD is the scheduling delay

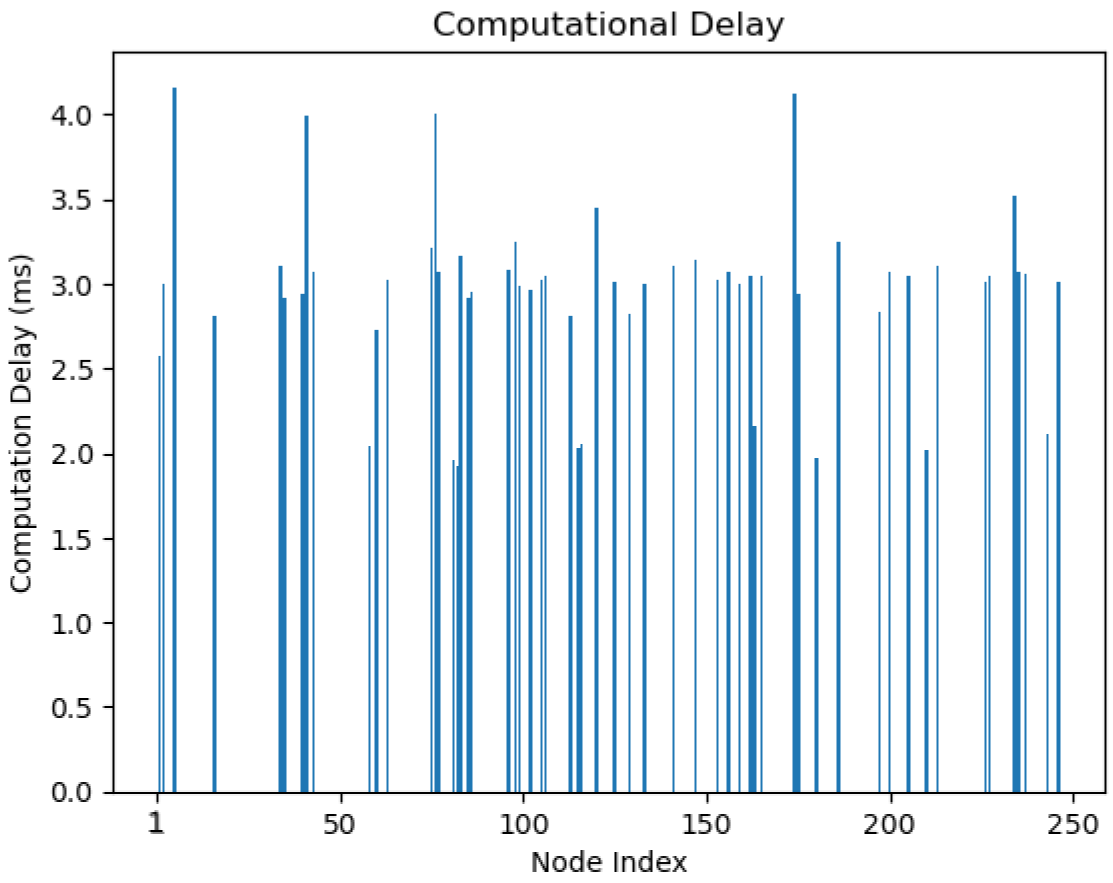


Figure 4. 16*: Computational Delay*

8- Throughput:

Measures the rate at which tasks are processed by the Fog system. Higher throughput indicates better scheduling. Throughput records the processing rate for each Fog node after handling tasks. Higher throughput indicates better scheduling and high processing power at the nodes, which improves the QoSs, as shown in the figure 4.17 (the Throughput plot shows the average rate for each node after several iterations). It expresses itself as follows:

$$TH = \frac{PT}{\sum_0^m T_n} \qquad (4.8)$$

Where TH is the Throughput's rate, PT is the number of processed tasks, and $\sum_0^m T_n$ is the total number of tasks received by the node

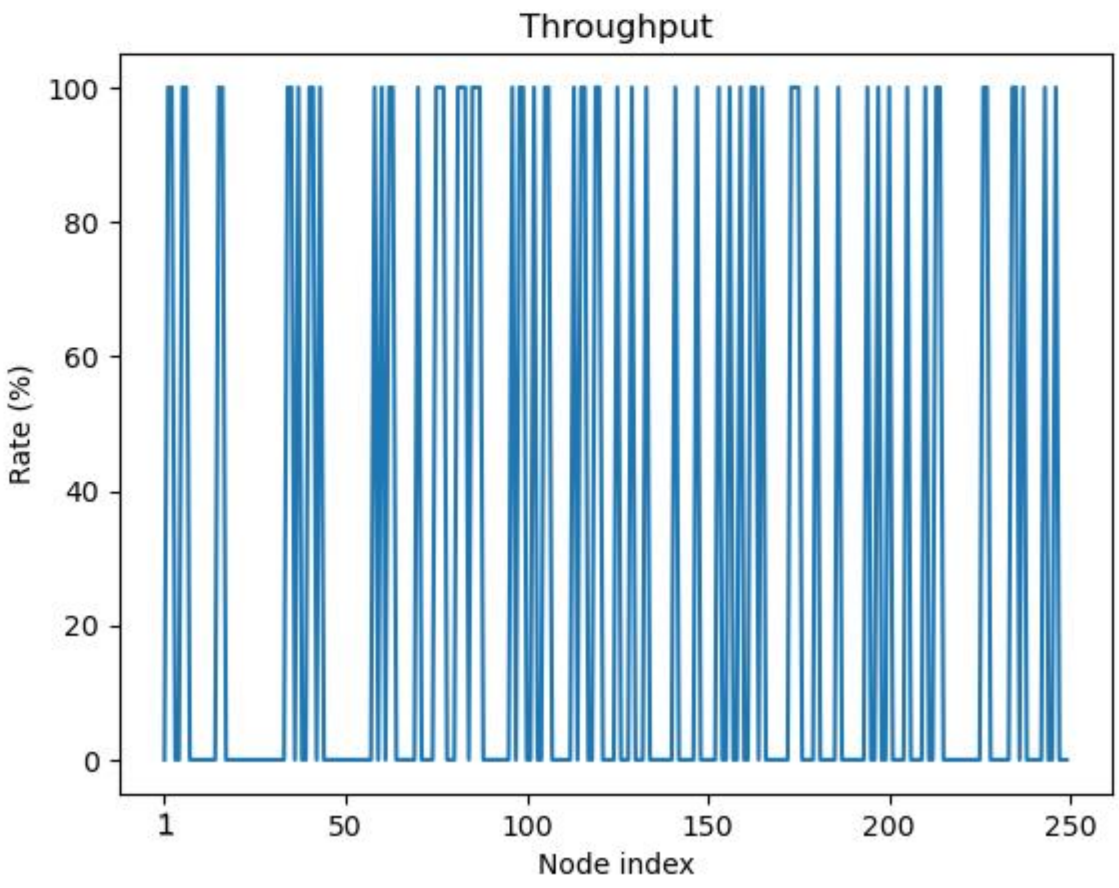


***Figure 4. 17**: Throughput*

9- Network Congestion:

Indicates high network traffic leading to increased delays. In Network Congestion, data extracted from each node's task processing iterations is used to plot network congestion, as shown in figure 4.18. Illustrates the effectiveness of MODRL in preventing network congestion. Preventing network congestion improves overall system performance. Formulate it mathematically in the following manner:

$$NC = \sum_{0}^{m} T_q \qquad (4.9)$$

Where NC is the Network Congestion, and $\sum_{0}^{m} T_q$ is the total number of tasks being queued in the Fog nodes.

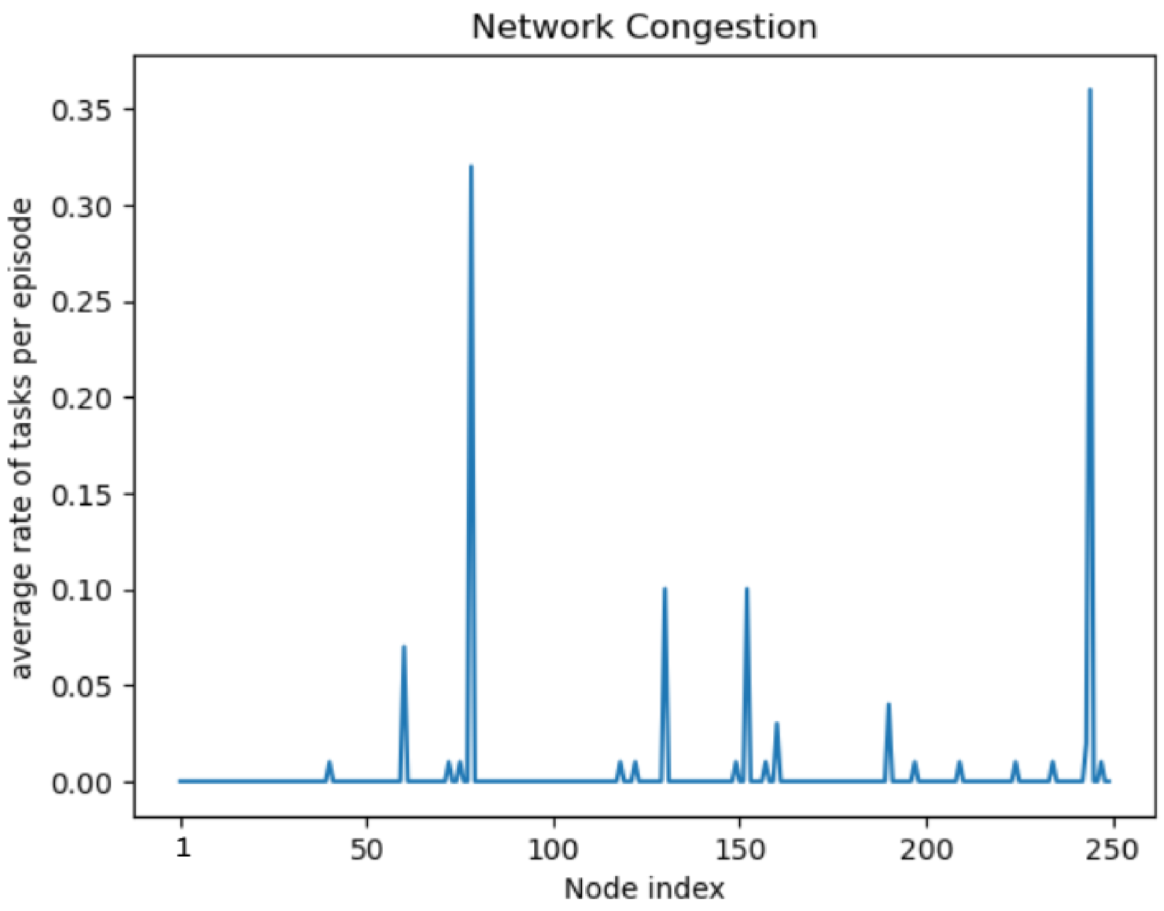


Figure 4. 18*: NetworkCongestion.*

B. PROPOSED ALGORITHM: DQN + MOEA/D

When determining the metrics for MOEA/D, it is important to notice that the underlying ideas and equations are consistent with those already developed for NSGA-II. MOEA/D, similar to NSGA-II, follows the Multi-Objective Optimization approach, with the goal of optimizing a group of conflicting objectives concurrently. The metrics include key indications such as Latency, Throughput, Task Completion Time, Makespan Time, Queueing Delay, Communication Delay, Transmission Delay, Processing Delay, and Computational Delay. In addition, the evaluation also takes into account indirect measures such as CPU Load, Storage Capacity, and Network Congestion. Due to the similar characteristics of the optimization issue and the common objective of optimizing multiple conflicting goals, the definitions and equations for these metrics remain consistent in both NSGA-II and MOEA/D. The consistency in metric definitions enables a smooth and fair comparison of the two methods, guaranteeing a thorough assessment of their individual performances in the context of Intelligent Scheduling for Fog Computing.

1- Resource Utilization

A. CPU

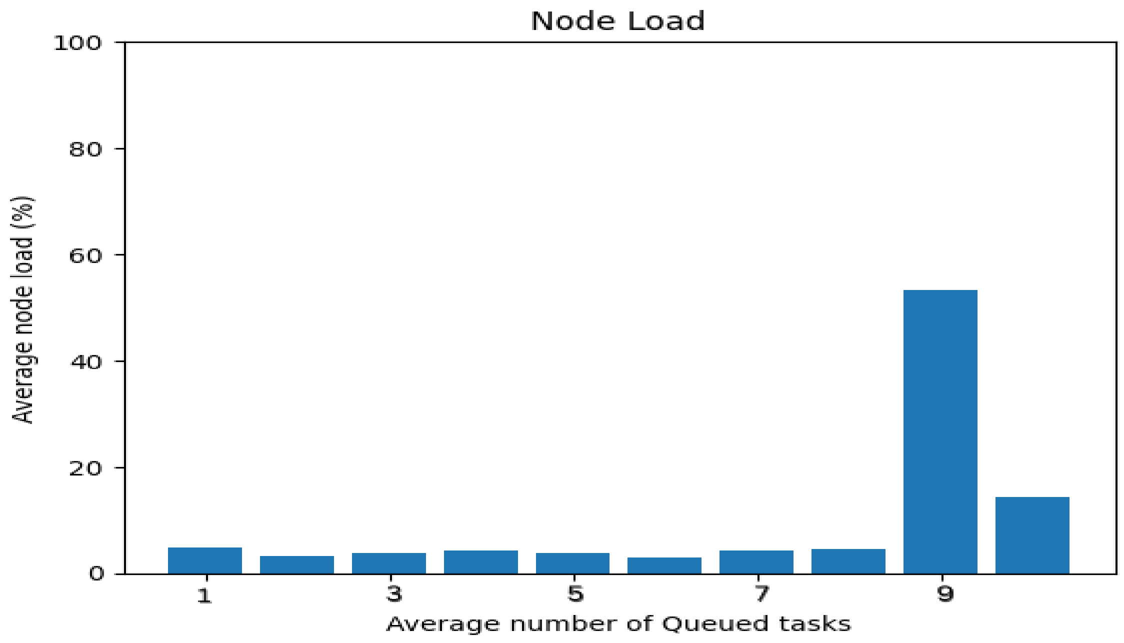


***Figure 4. 19**: CPU Load*

B. Storage Capacity

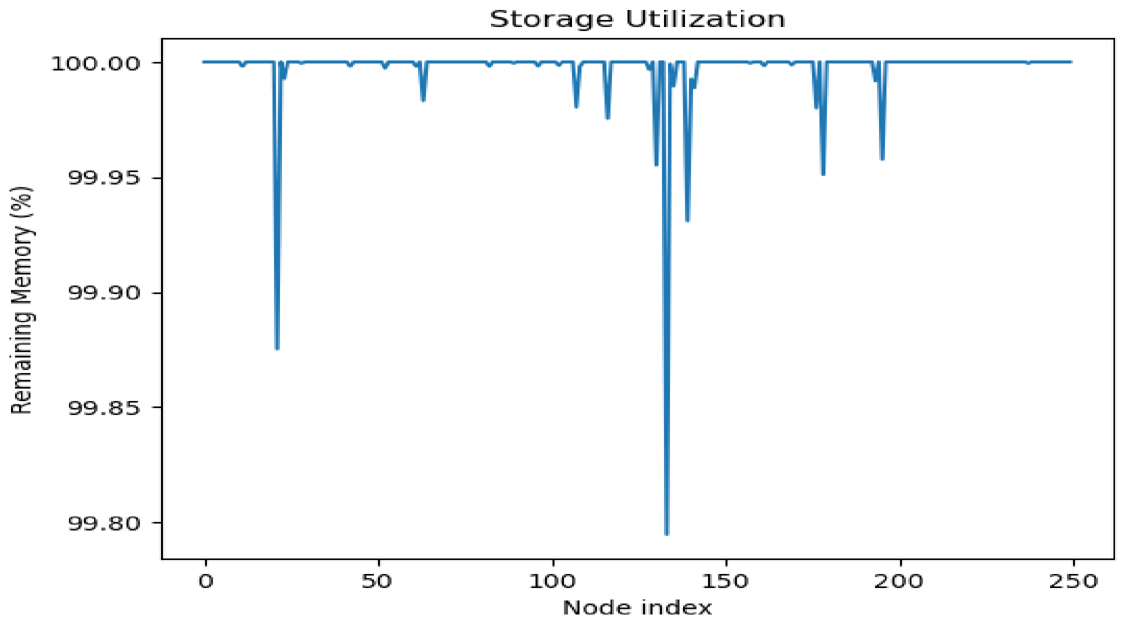


***Figure 4. 20**: Storage Capacity*

2-Latency

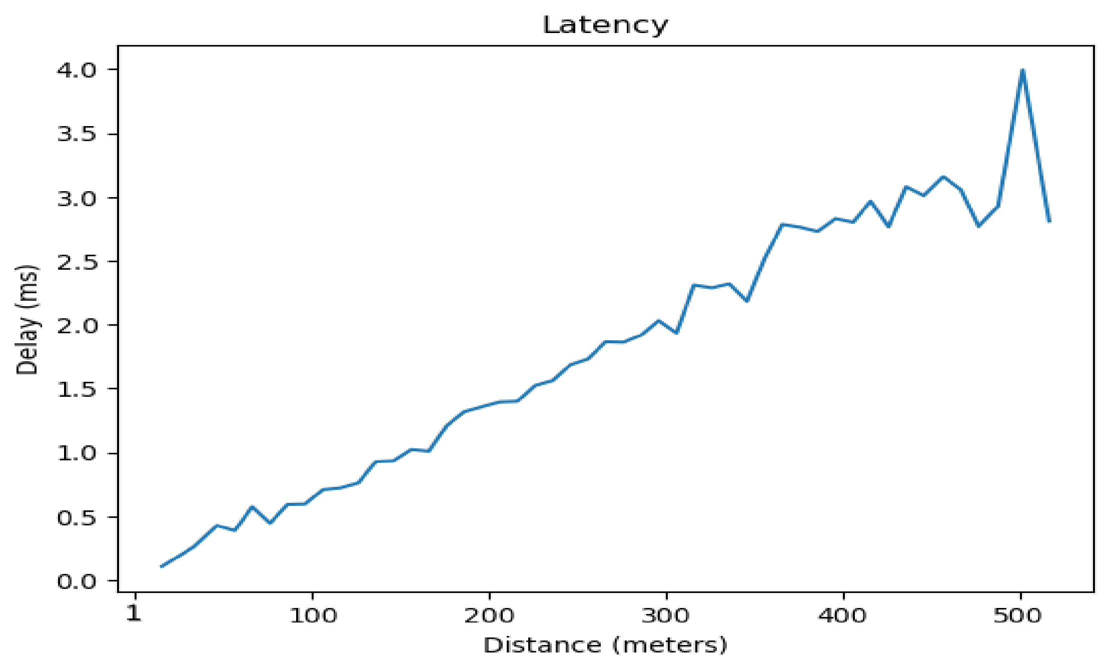


***Figure 4. 21**: Latency*

3-Task Completion Time

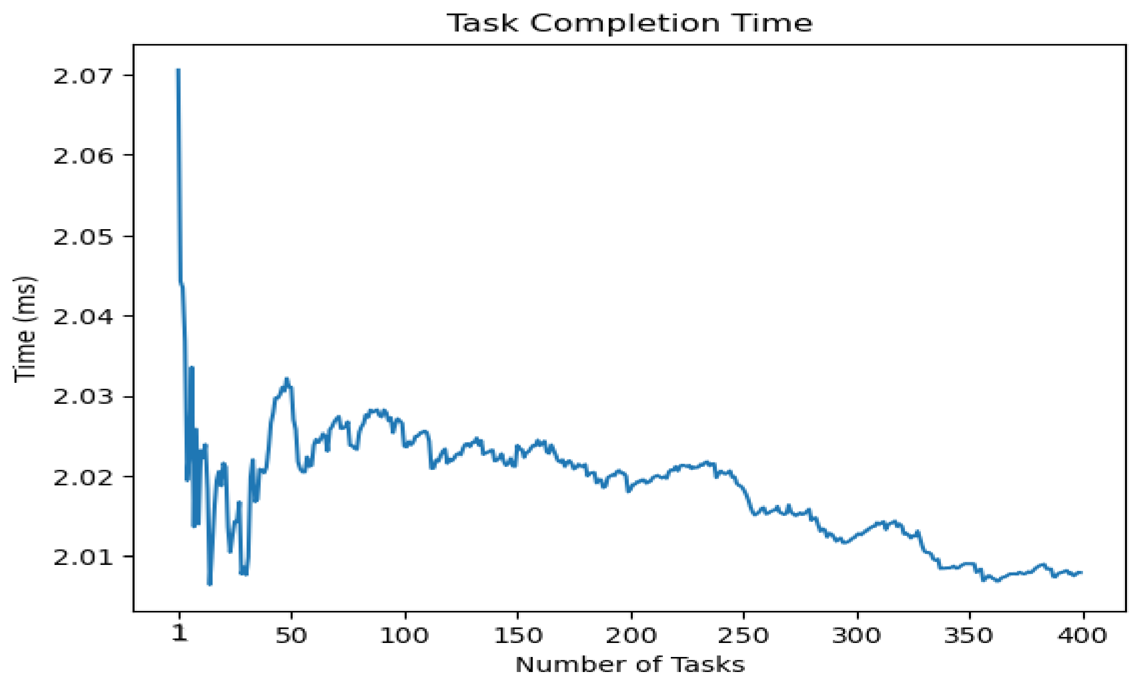


***Figure 4. 22**: Task Completion Time*

4-Makespan Time

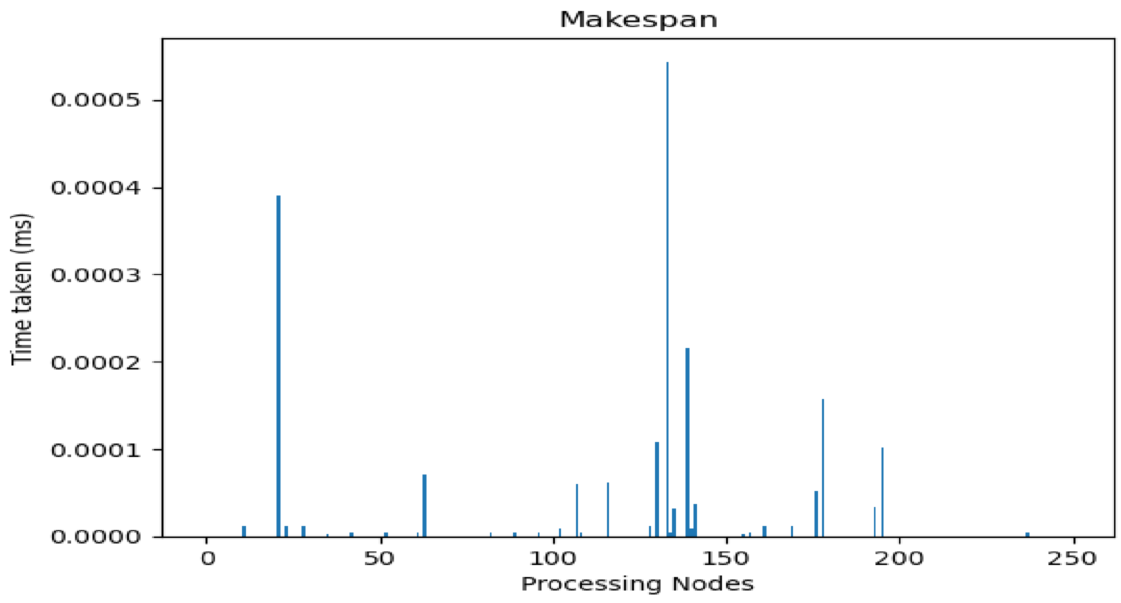


***Figure 4. 23**: Makespan Time*

5-Queueing delay

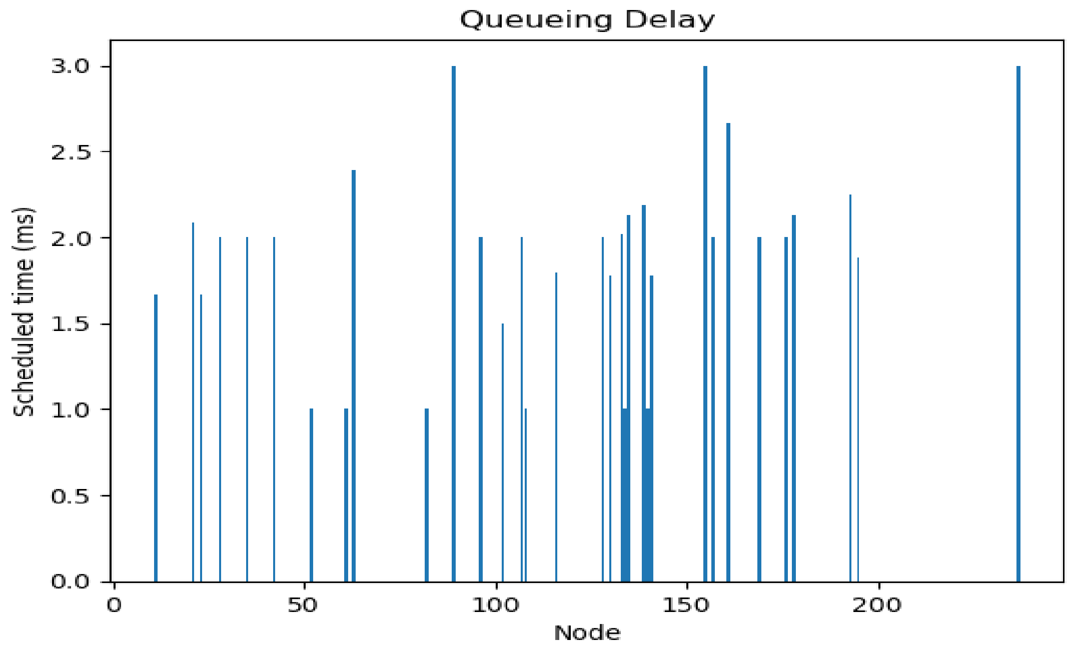


***Figure 4. 24**: Queueing Delay*

6-Communication delay

A. Propagation Delay

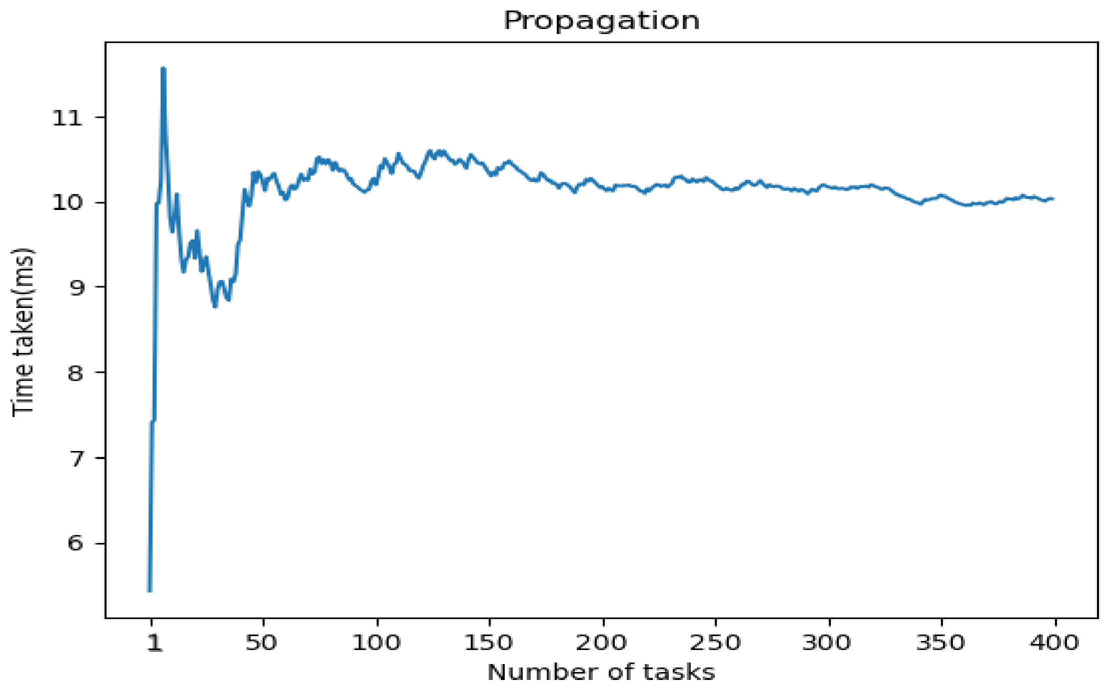


***Figure 4. 25**: Propagation Delay*

B. Transmission Delay

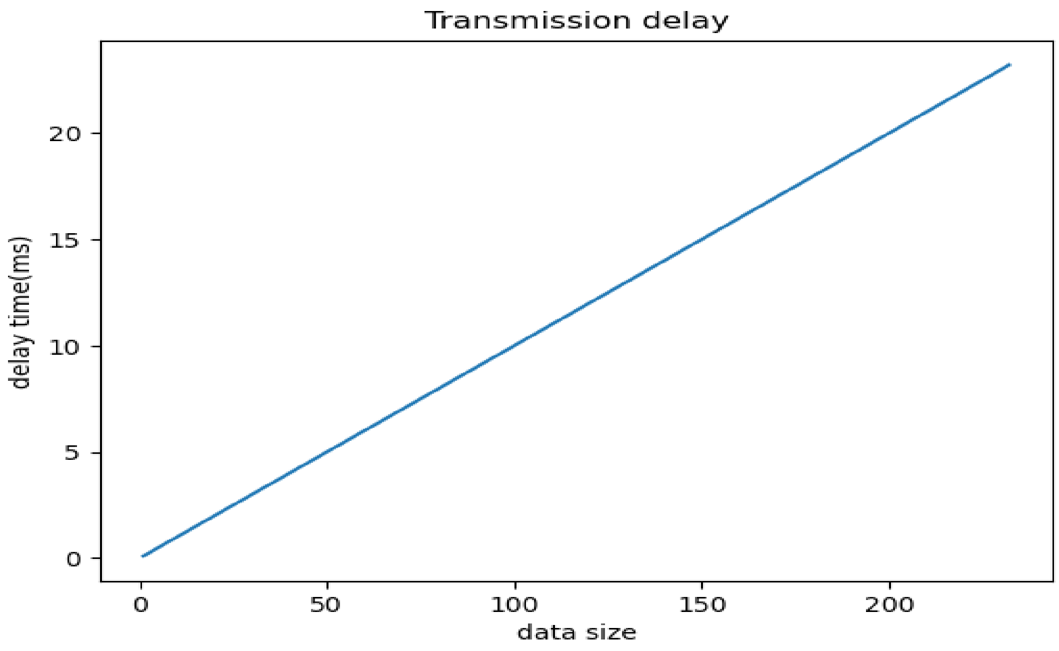


***Figure 4. 26**: Transmission Delay*

C.Processing Delay

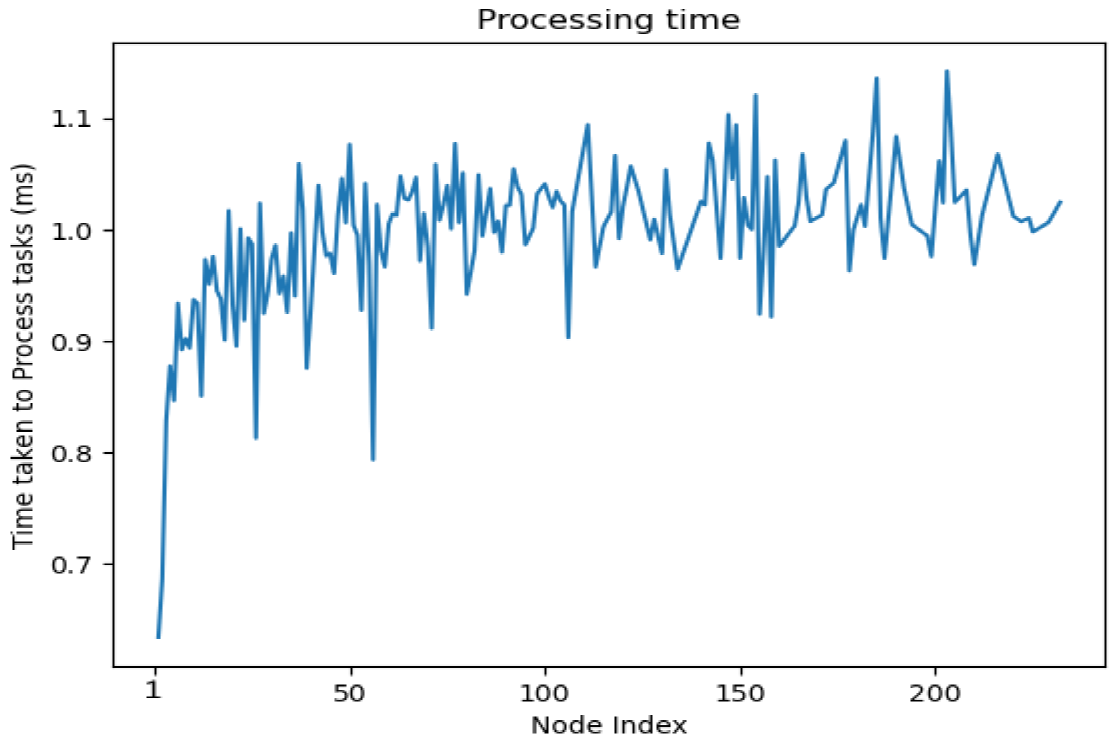


Figure 4. 27*: Processing Delay*

7-Computational Delay

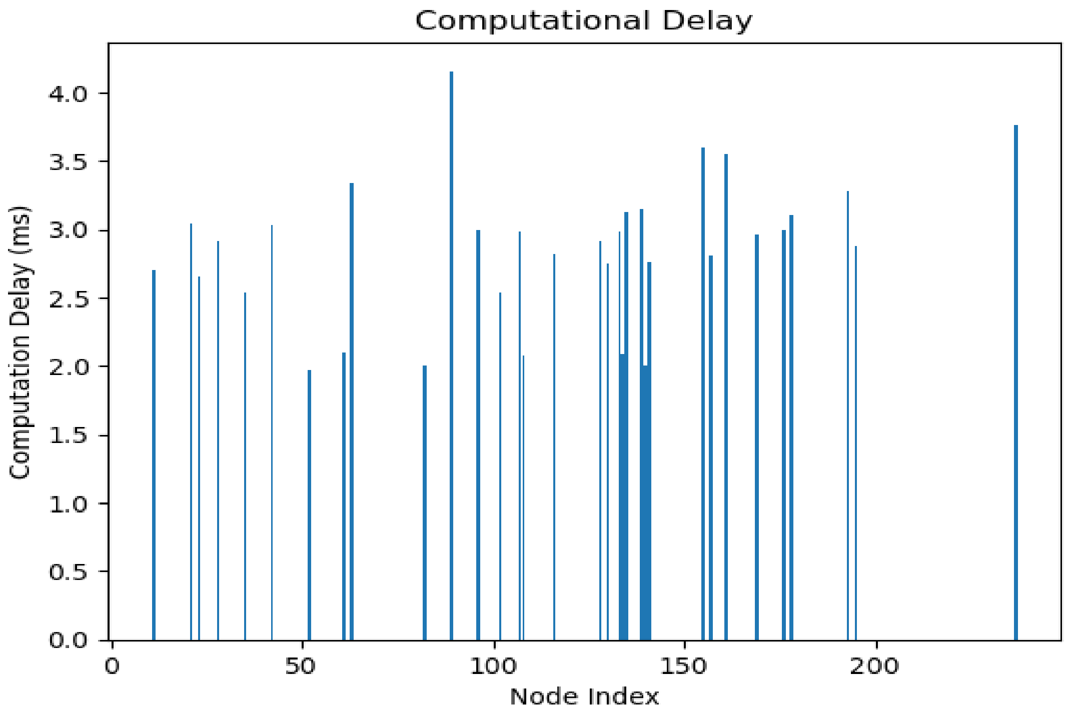


Figure 4. 28*: Computational Delay*

8-Throughput

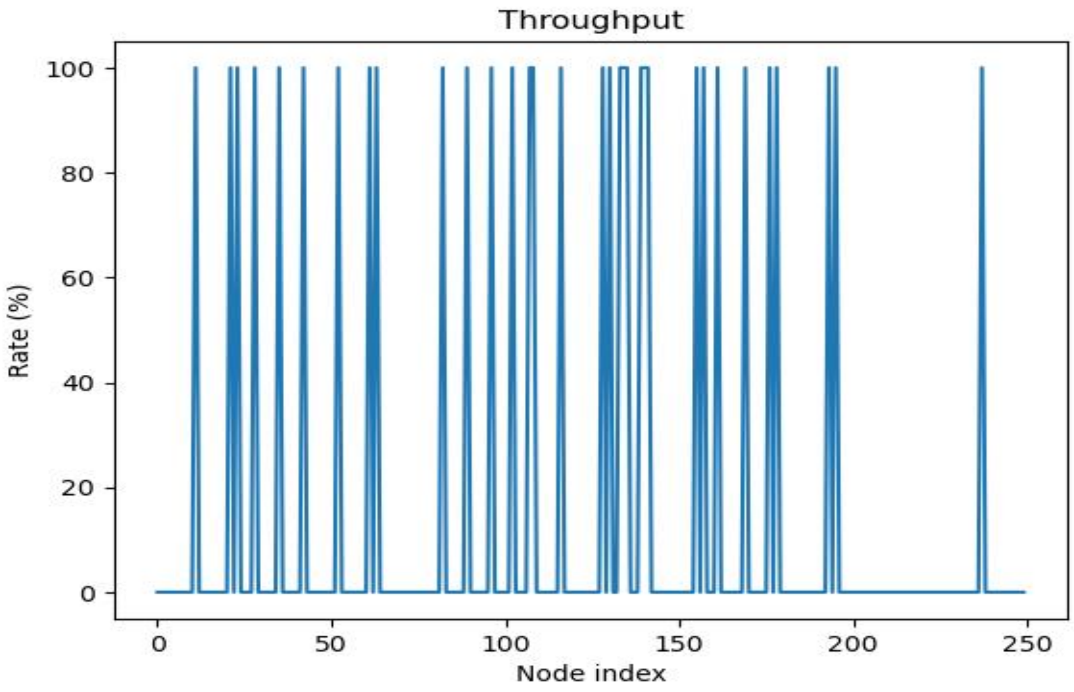


Figure 4. 29*: Throughput*

9-Network Congestion

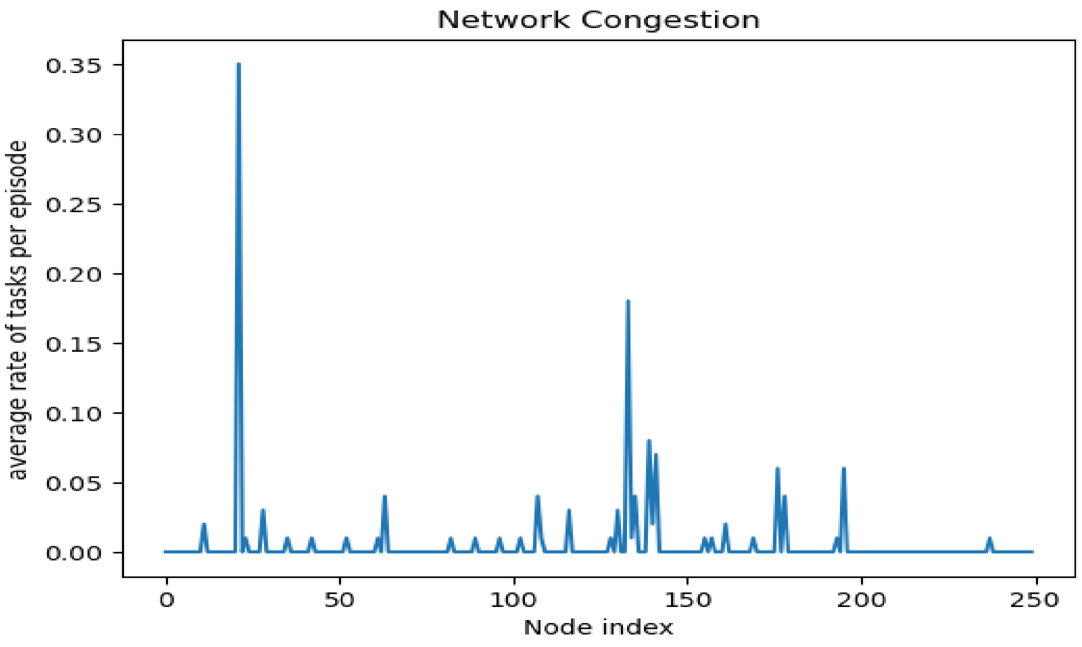


Figure 4. 30*: Network Congestion*

The divergence in results between NSGA-II and MOEA/D can be attributed to differences in their optimization mechanisms, strategies, and the specific characteristics of the optimization problem. These differences include the search mechanism, population initialization, problem characteristics, parameter settings, problem complexity, stochastic nature, and algorithm sensitivity. NSGA-II uses a Non-Dominated Sorting method, while MOEA/D Decomposes the Multi-Objective problem into sub-problems. The initialization of the population and diversity maintenance mechanisms also contribute to the exploration of different areas of the key space. The performance of evolutionary algorithms is sensitive to parameter settings, as well the difficulty of the optimization problem can favor one algorithm over the other. To understand the specific reasons for these differences, a thorough analysis of the Pareto Fronts and convergence behavior can provide valuable insights.

C- RESULTS COMPARISON

In this study, a comparative analysis was conducted between the proposed Intelligent Scheduling Strategy, based on MODRL algorithm, and the most relevant scheduling strategy presented in table 2.1. With a focus on key performance indicators. The validation includes three critical aspects: efficiency, performance, and adaptability. In terms of efficiency, which involves Task Completion Time, Makespan, and Queueing delay, the proposed MODRL-based strategy displayed more effectiveness by yielding the lowest values in these metrics, compared to the approach in table 2.1. Regarding performance, which considers Computation Delay, Storage Capacity, CPU Load, Throughput, and Network Congestion. The scheduling strategy excelled with a lower Computation Delay, a CPU Load maintained below 50%, a higher amount of storage space available, higher Throughput, and achieving low Network Congestion, whereas the approach outlined in table 2.1 lacked a simulation. Thus, the proposed Intelligent Scheduling Strategy showed it had higher performance. Lastly, in terms of adaptability, which comprises Communication Delay (Transmission Delay, Propagation Delay and Processing Delay), the proposed MODRL algorithm demonstrated advantages by achieving lower Communication Delay and effectively justifying Network Congestion, an aspect not addressed in existing algorithm in table 2.1. Thus, this enhanced adaptability. Table 4.1 details the comparison of the proposed Intelligent Scheduling and existing scheduling algorithms. From this comparison, it can be concluded that the proposed (DQN+MULTI-OBJECTIVE OPTIMIZATION) algorithm achieves better performance and is more effective and adaptive compared to the most relevant existing methods in table 4.1.

When considering scheduling in Fog Computing using DRL, multiple metrics are used to evaluate the QoS offered by the system. QoS-related metrics encompass Latency, Throughput, Task Completion Time, Makespan Time, and several delay components such as Queueing Delay, Communication Delay, Transmission Delay, Processing Delay, and Computational Delay. Latency is a crucial statistic for measuring QoS, since it represents the time, it takes for data to travel over the network and finish processing. Throughput, which refers to the speed at which tasks are successfully completed, is an additional crucial aspect that affects the QoS. The Task Completion Time and Makespan time offer valuable insights into the efficiency of the scheduling system, with lower values suggesting quicker and more efficient job execution. Queueing Delay is the amount of time that jobs wait in a queue before being processed, and it directly affects how quickly the system responds. Delays, such as those in Communication, Transmission, Processing, and Computing, have an impact on the overall QoS by affecting the speed and reliability of data transfer and task execution. An efficiently optimized scheduling method, led by DRL, aims to collectively minimize these metrics. This ensures a Fog Computing environment that is responsive, efficient, and dependable while also meeting the QoS requirements of various applications and users. Resource Utilization does not directly measure QoS, but it does have an indirect impact on QoS by affecting the efficiency and performance of the system. Optimal allocation of tasks enhances the dependability and agility of the Fog Computing ecosystem.

The reason behind the proposed Intelligent Scheduling Strategy's success is the prominent role of the MODRL algorithm, which is dedicated to the minimization of task execution times and the effective optimization of objectives within the Fog Orchestrator. As well as reduced waiting times for tasks.

***Table 4. 1**: Comparison of proposed algorithm and most relevant existing works.*

Ref.	**Described Algorithm**	**Core Contribution**	**MO-DRL**	**PERFORMANCE(Average)**					**EFFICIENCY (Average ms)**			**ADAPTABILITY (Average ms)**			
				CPU Load %	**Storage %**	**L.(ms)**	**TH %**	**NC (rate)**	**TCT.**	**MT.**	**QD.**	**Propagation delay**	**TD.**	**PD.**	**CD.**
Gazori et al.	DRL	Task scheduling	×	--	--	7500	--	--	19000	250	2500	--	--	--	7500
Wang et al.	RL	Task scheduling	×	--	--	--	--	--	20000	2000	--	--	--	--	--
Wu *et al.*	DRL	Task scheduling	×	--	--	--	--	--	--	--	--	--	--	--	--
Qi, Zhuo et al.	DRL	Task scheduling	×	--	--	--	--	--	--	--	--	--	--	--	--
Sheng *et al.*	DRL	Task scheduling	×	--	--	--	--	--	--	--	--	--	--	--	--
Jin *et al.*	DRL	Task scheduling	×	--	--	--	--	--	--	--	--	--	--	--	--
Proposed (DQN+NSGA2)	DRL+Multi objective optimization	Intelligent scheduling strategy (task scheduling and task allocation	√	**10**	**99.9**	**3.5**	**100**	**0.10**	**2.02**	**10**	**2**	**9.5**	**25**	**1**	**3**
Proposed (DQN+MOEA/D)	DRL+Multi objective optimization	Intelligent scheduling strategy (task scheduling and task allocation	√	**10**	**99.9**	**4.0**	**100**	**0.07**	**2.02**	**10**	**2**	**10.5**	**25**	**1**	**2.5**

4.8 SUMMARY

The proposed Intelligent Scheduling technique aims to optimize task scheduling and task allocation in the quickest time possible, with a focus on maximizing resource use. The integration of Multi-Objective Deep Reinforcement Learning is achieved by combining Deep Q-Network with Multi-Objective Optimization approaches. The suggested technique outperforms conventional methods by prioritizing crucial validation parameters, including Task Completion Time, Queueing Delay, Makespan, CPU Load, Storage Capacity, Latency, Computational Delay, Propagation Delay, Processing Delay, and Transmission Delay, Throughput and Network Congestion. By utilizing the learning capabilities of DQN and the efficiency improvements from Multi-Objective Optimization, the system adjusts dynamically to various and changing Fog Computing environments. The distinct combination of these elements establishes the suggested algorithm as a superior performer, showcasing improved effectiveness in reducing delay, optimizing CPU usage, and maximizing data processing capacity. The combination of Deep Reinforcement Learning and Multi-Objective Optimization paradigms in this technique represents notable progress in Intelligent Scheduling algorithms for Fog Computing environments.

CHAPTER FIVE - CONCLUSIONS AND FUTURE WORKS

With the development of Cloud Computing technology, FC has gradually become a significant middle layer in Cloud Computing, which has less time delay, enhanced interactivity, and stronger processing capacity terminal equipment. In addition, Fog Computing offers an edge-centric and distributed approach to enhance and complement overall computing capabilities.

5.1 Conclusion

This study aims to resolve the resource management problem in Fog Computing environments. The study intends to improve scheduling by concurrently considering many goals, which will increase the overall performance, efficiency, and adaptability of the Fog Computing system. The task allocation and task scheduling problem in FC is an important problem in Fog Computing, and its calculation method directly affects the results and efficiency of task execution in a Fog environment. This study proposes a MODRL-based Deep Q Network and Multi-Objective Optimization (NSGA2 + MOEA/D) to tackle task allocation and task scheduling problems in FC, which is implemented in a Fog Orchestrator that can choose the optimal node by considering three objectives (Load, Priority, and Distance).

The experimental results prove that the key findings of the proposed DQN+ Multi-Objective Optimization algorithm are effective based on eight validation metrics: Task Completion Time, Makespan Time, Queueing Delay, Latency,Network Congestion, Throughput, CPU Load, and Storage Capacity with an average value of 2.02ms, 10ms, 2ms, , 3.5ms, 0.10ms, %100, %10,%99, respectively. As well as adaptive in terms of four performance metrics: Propagation Delay, Transmission Delay, Processing Delay, and Computational Delay with an average value of 9.5ms, 25ms, 1ms, 3ms, respectively, in solving the MODRL of FC for task scheduling and task allocation. Therefore, the intelligent scheduling strategy in FG is essential for the following:

1- Minimizing Latency, by intelligently allocating tasks and processing locations, bringing computation closer to the edge for real-time applications.
2- Reducing Makespan, contributing to faster completion times for the entire workload.
3- Reduces Communication Delay, by making informed determinations regarding the optimal location for job processing, whether it is locally or in the Cloud. This aids in minimizing the necessity for lengthy data transfers, hence improving the overall responsiveness of the system.
4- Maximize Throughput, confirming that the system can handle a higher workload effectively.
5- Maximazing the Resource Utilization, ensuring reliable and efficient operation in a dynamic environment, and optimizing the use of resources.

5.2 Future Works

For the future, the algorithm will be adjusted to enhance the mechanisms for re-scheduling tasks facing long queueing times in the Orchestrator, which may involve the following:

1- Feedback-based re-scheduling, A re-scheduling mechanism continuously learns from past decisions, analyzing factors like task completion times and resource utilization.
2- Predictive re-scheduling, is a method that uses forecasting and predictive analytics to anticipate future changes in the Fog Computing environment.

REFERENCES

Abdel-Basset, M. *et al.* (2021a) ‘Multi-Objective Task Scheduling Approach for Fog Computing’, *IEEE Access*, 9, pp. 126988–127009. Available at: https://doi.org/10.1109/ACCESS.2021.3111130.

Abdel-Basset, M. *et al.* (2021b) ‘Multi-Objective Task Scheduling Approach for Fog Computing’, *IEEE Access*, 9, pp. 126988–127009. Available at: https://doi.org/10.1109/ACCESS.2021.3111130.

Alizadeh, M.R. *et al.* (2020a) ‘Task scheduling approaches in fog computing: A systematic review’, *International Journal of Communication Systems*, 33(16), pp. 1–36. Available at: https://doi.org/10.1002/dac.4583.

Alizadeh, M.R. *et al.* (2020b) ‘Task scheduling approaches in fog computing: A systematic review’, *International Journal of Communication Systems*, 33(16). Available at: https://doi.org/10.1002/dac.4583.

Alsmirat, M., Institute of Electrical and Electronics Engineers. French Section and Institute of Electrical and Electronics Engineers *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC) : Paris, France. April 20-23, 2020*.

Atlam, H.F., Walters, R.J. and Wills, G.B. (2018) ‘Fog computing and the internet of things: A review’, *Big Data and Cognitive Computing*. MDPI, pp. 1–18. Available at: https://doi.org/10.3390/bdcc2020010.

Chen, W. *et al.* (2021) ‘A novel multiobjective evolutionary algorithm based on decomposition and multi-reference points strategy’. Available at: http://arxiv.org/abs/2110.14124.

Deb, K. *et al.* (2002a) *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*.
Deb, K. *et al.* (2002b) *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*.

Deb, Kalyan and Deb, Kalyanmoy (2014) *Multiobjective Optimization Using Evolutionary Algorithms Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction*. Available at: http://www.iitk.ac.in/kangal/deb.htm.

Dizdarevic, J. *et al.* (2018) ‘Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration’. Available at: https://doi.org/10.1145/3292674.

Dlamini, S. and Ventura, N. (2019) ‘Resource management in fog computing: Review’, in *icABCD 2019 - 2nd International Conference on Advances in Big Data, Computing and Data Communication Systems*. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/ICABCD.2019.8851016.

Fahimullah, M., Ahvar, S. and Trocan, M. (2022) *A Review of Resource Management in Fog Computing: Machine Learning Perspective*.

Gazori, P., Rahbari, D. and Nickray, M. (2020a) ‘Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach’, *Future Generation Computer Systems*, 110, pp. 1098–1115. Available at: https://doi.org/10.1016/j.future.2019.09.060.

Gazori, P., Rahbari, D. and Nickray, M. (2020b) ‘Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach’, *Future Generation Computer Systems*, 110, pp. 1098–1115. Available at: https://doi.org/10.1016/j.future.2019.09.060.

Ghobaei-Arani, M., Souri, A. and Rahmanian, A.A. (2020a) ‘Resource Management Approaches in Fog Computing: a Comprehensive Review’, *Journal of Grid Computing*. Springer. Available at: https://doi.org/10.1007/s10723-019-09491-1.

Ghobaei-Arani, M., Souri, A. and Rahmanian, A.A. (2020b) ‘Resource Management Approaches in Fog Computing: a Comprehensive Review’, *Journal of Grid Computing*. Springer. Available at: https://doi.org/10.1007/s10723-019-09491-1.

Guerrero, C., Lera, I. and Juiz, C. (2019) ‘Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures’, *Future Generation Computer Systems*, 97, pp. 131–144. Available at: https://doi.org/10.1016/j.future.2019.02.056.

Hazra, A. *et al.* (2023) ‘Fog computing for next-generation Internet of Things: Fundamental, state-of-the-art and research challenges’, *Computer Science Review*. Elsevier Ireland Ltd. Available at: https://doi.org/10.1016/j.cosrev.2023.100549.

Henderson, P. *et al.* (2018) *Deep Reinforcement Learning that Matters*. Available at: www.aaai.org.

Islam, M.S.U., Kumar, A. and Hu, Y.C. (2021) ‘Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions’, *Journal of Network and Computer Applications*. Academic Press. Available at: https://doi.org/10.1016/j.jnca.2021.103008.

Jang, B. *et al.* (2019) ‘Q-Learning Algorithms: A Comprehensive Classification and Applications’, *IEEE Access*, 7, pp. 133653–133667. Available at: https://doi.org/10.1109/ACCESS.2019.2941229.

Jin, C. *et al.* (2023) ‘Reinforcement Learning-Based Intelligent Task Scheduling for Large-Scale IoT Systems’, *Wireless Communications and Mobile Computing*, 2023. Available at: https://doi.org/10.1155/2023/3660882.

Kaur, M. and Kumar, V. (2018) ‘Parallel non-dominated sorting genetic algorithm-II-based image encryption technique’, *Imaging Science Journal*, 66(8), pp. 453–462. Available at: https://doi.org/10.1080/13682199.2018.1505327.

Kaur, N., Kumar, A. and Kumar, R. (2021) ‘A systematic review on task scheduling in Fog computing: Taxonomy, tools, challenges, and future directions’, *Concurrency and Computation: Practice and Experience*, 33(21). Available at: https://doi.org/10.1002/cpe.6432.

Laghari, A.A., Jumani, A.K. and Laghari, R.A. (2021) ‘Review and State of Art of Fog Computing’, *Archives of Computational Methods in Engineering*, 28(5), pp. 3631–3643. Available at: https://doi.org/10.1007/s11831-020-09517-y.

Lakhan, A. *et al.* (2022) ‘Efficient deep-reinforcement learning aware resource allocation in SDN-enabled fog paradigm’, *Automated Software Engineering*, 29(1). Available at: https://doi.org/10.1007/s10515-021-00318-6.

Lazaridis, A. (2020) *Deep Reinforcement Learning: A State-of-the-Art Walkthrough, Journal of Artificial Intelligence Research*.

Li, K., 2021. Decomposition multi-objective evolutionary optimization: From state-of-the-art to future opportunities. arXiv preprint arXiv:2108.09588.

Liu, C., Xu, X. and Hu, D. (2015) ‘Multiobjective reinforcement learning: A comprehensive overview’, *IEEE Transactions on Systems, Man, and*

Cybernetics: Systems, 45(3), pp. 385–398. Available at: https://doi.org/10.1109/TSMC.2014.2358639.

Liu, Y. *et al.* (2019) 'Deep Reinforcement Learning for Offloading and Resource Allocation in Vehicle Edge Computing and Networks', *IEEE Transactions on Vehicular Technology*, 68(11), pp. 11158–11168. Available at: https://doi.org/10.1109/TVT.2019.2935450.

Von Lücken, C., Barán, B. and Brizuela, C. (2014) 'A survey on multi-objective evolutionary algorithms for many-objective problems', *Computational Optimization and Applications*, 58(3), pp. 707–756. Available at: https://doi.org/10.1007/s10589-014-9644-1.

Mao, H. *et al.* (2016) 'Resource management with deep reinforcement learning', in *HotNets 2016 - Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. Association for Computing Machinery, Inc, pp. 50–56. Available at: https://doi.org/10.1145/3005745.3005750.

Martinez, I., Hafid, A.S. and Jarray, A., 2020. Design, resource management, and evaluation of fog computing systems*: a survey. IEEE Internet of Things Journal, 8(4), pp.2494-2516.*

Matrouk, K. and Alatoun, K. (2021) 'Scheduling Algorithms in Fog Computing: A Survey', *International Journal of Networked and Distributed Computing*, 9(1), pp. 59–74. Available at: https://doi.org/10.2991/IJNDC.K.210111.001.

Mohamad Shirajuddin, T., Muhammad, N.S. and Abdullah, J. (2023) 'Optimization problems in water distribution systems using Non-dominated Sorting Genetic Algorithm II: An overview', *Ain Shams Engineering Journal*, 14(4). Available at: https://doi.org/10.1016/j.asej.2022.101932.

Mossalam, H. *et al.* (2016) 'Multi-Objective Deep Reinforcement Learning'. Available at: http://arxiv.org/abs/1610.02707.

Mseddi, A. *et al.* (2019) 'Intelligent Resource Allocation in Dynamic Fog Computing Environments', in *Proceeding of the 2019 IEEE 8th International Conference on Cloud Networking, CloudNet 2019*. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/CloudNet47604.2019.9064110.

Mukherjee, M., Shu, L. and Wang, D. (2018) 'Survey of fog computing: Fundamental, network applications, and research challenges', *IEEE*

Communications Surveys and Tutorials, 20(3), pp. 1826–1857. Available at: https://doi.org/10.1109/COMST.2018.2814571.

Naha, R.K. *et al.* (2018) 'Fog computing: Survey of trends, architectures, requirements, and research directions', *IEEE Access*, 6, pp. 47980–48009. Available at: https://doi.org/10.1109/ACCESS.2018.2866491.

Nassar, A.T. and Yilmaz, Y. (2018) 'Reinforcement Learning-based Resource Allocation in Fog RAN for IoT with Heterogeneous Latency Requirements'. Available at: http://arxiv.org/abs/1806.04582.

Ni, J. *et al.* (2018) 'Securing Fog Computing for Internet of Things Applications: Challenges and Solutions', *IEEE Communications Surveys and Tutorials*, 20(1), pp. 601–628. Available at: https://doi.org/10.1109/COMST.2017.2762345.

Niranjan, S.K. *et al.* Institute of Electrical and Electronics Engineers, Institute of Electrical and Electronics Engineers. Bangalore Section, and IEEE Computational Intelligence Society. Bangalore Chapter, ''A survey: Integration of IoT and fog computing,'' in Proc. 2nd Int. Conf. Green Comput. Internet Things (ICGCIoT), Karnataka, India, Aug. 2018, pp. 235–239.

Özdemir, S., Attea, B.A. and Khalil, Ö.A. (2013) 'Multi-objective evolutionary algorithm based on decomposition for energy efficient coverage in wireless sensor networks', *Wireless Personal Communications*, 71(1), pp. 195–215. Available at: https://doi.org/10.1007/s11277-012-0811-3.

Plaat, A. (2022) 'Deep Reinforcement Learning, a textbook'. Available at: https://doi.org/10.1007/978-981-19-0638-1.

Qi, F.A.N., Zhuo, L. and Xin, C. (2020) 'Deep Reinforcement Learning Based Task Scheduling in Edge Computing Networks', in *2020 IEEE/CIC International Conference on Communications in China, ICCC 2020*. Institute of Electrical and Electronics Engineers Inc., pp. 835–840. Available at: https://doi.org/10.1109/ICCC49849.2020.9238937.

Qiao, J. *et al.* (2019) 'A decomposition-based multiobjective evolutionary algorithm with angle-based adaptive penalty', *Applied Soft Computing Journal*, 74, pp. 190–205. Available at: https://doi.org/10.1016/j.asoc.2018.10.028.

Rahman, G.M.S., Dang, T. and Ahmed, M. (2021) 'Deep reinforcement learning based computation offloading and resource allocation for low-latency fog radio access networks', *Intelligent and Converged Networks*, 1(3), pp. 243–257. Available at: https://doi.org/10.23919/icn.2020.0020.

Rani, A., Prakash, V. and Darbari, M. (2022) 'Fog Computing Paradigm with Internet of Things to Solve Challenges of Cloud with IoT', in *Communications in Computer and Information Science*. Springer Science and Business Media Deutschland GmbH, pp. 72–84. Available at: https://doi.org/10.1007/978-3-031-23724-9_7.

Roderick, M., MacGlashan, J. and Tellex, S. (2017) 'Implementing the Deep Q-Network'. Available at: http://arxiv.org/abs/1711.07478.

Roheed Khaliqyar, A.Professor., Amir Kror Shahidzay, A.Prof. and Aslamza, Assistant.P.S. (2023) 'An Approach from Internet of Things to Cloud of Things using Fog Computing', *International Journal of Multidisciplinary Research and Analysis*, 06(04). Available at: https://doi.org/10.47191/ijmra/v6-i4-53.

Sabireen, H. and Neelanarayanan, V. (2021) 'A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges', *ICT Express*, 7(2), pp. 162–176. Available at: https://doi.org/10.1016/j.icte.2021.05.004.

Sellami, B. *et al.* (2020) 'Deep Reinforcement Learning for Energy-Efficient Task Scheduling in SDN-based IoT Network', in *2020 IEEE 19th International Symposium on Network Computing and Applications, NCA 2020*. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/NCA51143.2020.9306739.

Sharma, S. and Kumar, V. (2022) 'A Comprehensive Review on Multi-objective Optimization Techniques: Past, Present and Future', *Archives of Computational Methods in Engineering*. Springer Science and Business Media B.V., pp. 5605–5633. Available at: https://doi.org/10.1007/s11831-022-09778-9.

Sheng, S. *et al.* (2021) 'Deep reinforcement learning-based task scheduling in iot edge computing', *Sensors*, 21(5), pp. 1–19. Available at: https://doi.org/10.3390/s21051666.

Sherbrooke, C.C. Discrete compound Poisson processes and tables of the geometric Poisson distribution,'' Nav. Res. Logistics Quart., vol. 15, no. 2, pp. 189–203, Jun. 1968, doi: 10.1002/nav.3800150206.

Tan, F., Yan, P. and Guan, X. (2017a) 'Deep Reinforcement Learning: From Q-Learning to Deep Q-Learning', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag, pp. 475–483. Available at: https://doi.org/10.1007/978-3-319-70093-9_50.

Tan, F., Yan, P. and Guan, X. (2017b) 'Deep Reinforcement Learning: From Q-Learning to Deep Q-Learning', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag, pp. 475–483. Available at: https://doi.org/10.1007/978-3-319-70093-9_50.

Thi Nguyen, T. et al. 2020. A multi-objective deep reinforcement learning framework. Engineering Applications of Artificial Intelligence, 96, p.103915.

Tran-Dang, H. *et al.* (2022) 'Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues', *Journal of Communications and Networks*, 24(1), pp. 83–98. Available at: https://doi.org/10.23919/jcn.2021.000041.

Verdú, S. (1998) *Fifty Years of Shannon Theory, IEEE Trans. Inf. Theory, vol. 44, no. 6, pp. 2057–2078, Oct. 1998, doi: 10.1109/18.720531*

Verma, S., Pant, M. and Snasel, V. (2021) 'A Comprehensive Review on NSGA-II for Multi-Objective Combinatorial Optimization Problems', *IEEE Access*, 9, pp. 57757–57791. Available at: https://doi.org/10.1109/ACCESS.2021.3070634.

Wang, N. *et al.* (2023) 'A Review of Deep Reinforcement Learning Methods and Military Application Research', *Mathematical Problems in Engineering*, 2023, pp. 1–16. Available at: https://doi.org/10.1155/2023/7678382.

Wang, Y., Dong, S. and Fan, W. (2023) 'Task Scheduling Mechanism Based on Reinforcement Learning in Cloud Computing', *Mathematics*, 11(15). Available at: https://doi.org/10.3390/math11153364.

Goudarzi, M. *et al.* (2023) 'Deep Reinforcement Learning-based scheduling for optimizing system load and response time in edge and fog computing environments', *Future Generation Computer Systems*, 152, pp. 55–69. Available at: https://doi.org/10.1016/j.future.2023.10.012.

Wei, Y. *et al.* (2018) ‘DRL-Scheduling: An intelligent QoS-Aware job scheduling framework for applications in clouds’, *IEEE Access*, 6, pp. 55112–55125. Available at: https://doi.org/10.1109/ACCESS.2018.2872674.

Wu, J. *et al.* (2021) ‘Deep Reinforcement Learning for Scheduling in an Edge Computing-Based Industrial Internet of Things’, *Wireless Communications and Mobile Computing*, 2021. Available at: https://doi.org/10.1155/2021/8017334.

Zhang, Q. and Li, H. (2007) ‘MOEA/D: A multiobjective evolutionary algorithm based on decomposition’, *IEEE Transactions on Evolutionary Computation*, 11(6), pp. 712–731. Available at: https://doi.org/10.1109/TEVC.2007.892759.

Zheng, T. *et al.* (2022) ‘Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing’, *Journal of Cloud Computing*, 11(1). Available at: https://doi.org/10.1186/s13677-021-00276-0.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 October 2023, accepted 19 November 2023, date of publication 27 November 2023,
date of current version 30 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3337034

RESEARCH ARTICLE

An Intelligent Scheduling Strategy in Fog Computing System Based on Multi-Objective Deep Reinforcement Learning Algorithm

MEDIA ALI IBRAHIM AND SHAVAN ASKAR, (Senior Member, IEEE)
Department of Information System Engineering, Erbil Polytechnic University, Erbil 44001, Iraq

Corresponding author: Media Ali Ibrahim (media.ibrahim@epu.edu.iq)

ABSTRACT Fog computing (FC) has recently emerged as a promising new paradigm that provides resource-intensive Internet of Things (IoT) applications with low-latency services at the network edge. However, the limited capacity of computing resources in fog colonies poses great challenges for scheduling and allocating application tasks. In this paper, we propose an intelligent scheduling strategy algorithm in an FC system based on multi-objective deep reinforcement learning (MODRL) to select nodes for task processing (fog nodes or cloud nodes) based on three objectives: node current load, node distance, and task priority. The proposed model addresses two main problems: task allocation and task scheduling. We employ three deep reinforcement learning (DRL) agents based on a deep Q network (DQN), one for each objective. However, this is a more challenging scenario because there is a trade-off among these objectives, and eventually, each algorithm may select different processing nodes according to its own objective, which brings us to a Pareto front problem. To solve this problem, we propose using multi-objective optimization, a multi-objective evolutionary algorithm based on decomposition (MOEA/D), and a non-dominated sorting genetic algorithm (NSGA2), which are multi-objective optimization algorithms that can choose the optimal node by considering three objectives. The simulation results show that our proposed intelligent scheduling strategy could achieve better outcomes for the various employed performance, efficiency, and adaptability metrics: Task Completion Time, Makespan, Queueing Delay, Propagation Delay, Transmission Delay, Processing Delay, Computational Delay, Latency, CPU Load, and Storage Utilization, with an average value of 2.02 ms, 10 ms, 2 ms, 9.9 ms, 25 ms, 1.0 ms, 3.5 ms, 10 %, and 99 %, respectively, compared with the existing related research studies.

INDEX TERMS Fog computing (FC), Internet of Things (IoT), multi-objective deep reinforcement learning (MODRL), deep Q network (DQN), scheduling and allocating tasks.

پوختە

بەم دواییانە (Fog computing)، وەک پارادایمیکێکی نوێ سەریهەڵداوە، کە خزمەتگوزارییەکانی بۆ ئەپلیکەیشنەکان لەسەردەمی ئینتەرنێتی (IoT) بە وچانێکی کەم بۆ تۆڕەکان ئامادە دەکات، بەڵام توانای سنورداری سەرچاوەکانی کومپیوتینگ (computing) لە کۆلۆنیەکانی فۆگ (Fog colonies) کێشەی گەورە بۆ خشتەی کارەکان وتەرخانکردنی ئەرکەکانی بەکارهێنان دروستدەکات. لەم کارەدا ئالگۆریتمێکی ستراتیژی خشتەکردنی زیرەک لە سیستەمی (FC) لەسەر بنەمای فێربوونی بەهێزکردنی قووڵی فرە-ئامانج (MODRL) بۆ دیاریکردنی بەیەکبەستەکان (nodes) بۆ پرۆسەکردنی ئەرک پێشنیار دەکرێت.

(Fog nodes or Cloud nodes) بەندە لەسەر سێ ئامانج: دۆخی ئێستای بەیەك بەستەکان، دووری بەیەك بەستەکان وپێشنیارکردنی ئەولەویەتی ئەرک (Load, Priority, Distance).

MODRL میتۆدۆلۆژیایەکی پێشکەوتووە، کە بیرۆکەکانی باشترکردنی فرە-ئامانج و فێربوونی بەهێزکردنی قووڵ بۆ بڕیاردان لە چارەسەرکردنی ئەو بارودۆخە ئاڵۆزانەی، کە چەندین ئامانجی ناکۆکی تێدایە تێکەڵ دەکات. ئەم تەکنیکە بە نرخە بەتایبەتی لەو بارودۆخانەدا، کە پێویستیان بە زیادکردنی ژمارەیەکی زۆر لە پێوەرەکان هەیە. لەهەمان کاتدا تەنانەت ئەگەر بە تەواوی هێڵیش نەبن وپێویستە مامەڵەکان ڕەچاو بکرێن. ئەم مۆدێلەی پێشنیارکراوە دوو کێشەی سەرەکی چارەسەر دەکات: 'تەرخانکردنی ئەرک وخشتەی ئەرک. سێ بەکارهێنانی فێربوونی بەهێزکردنی قووڵ (DRL) بریکارەکان لەسەر بنەمای تۆڕێکی قووڵی Q-Network (DQN)، یەک بۆ هەر ئامانجێک ئەمە جۆرێکی تایبەتە لە پێکهاتەی تۆڕی (ANN)، کە بۆ بەهێزکردنی فێربوون (RL) بەکاردێت. ئەلگۆریتمی DQN تۆڕی (NN) بەکاردێنێت، کە بە گشتی تۆڕێکی ئاڵۆزە (CNN)، بۆ خەمڵاندنی کرداری Q. ئەمەش ڕێگە بە مۆدێلەکە دەدات، کە بەشێوەیەکی کاریگەر بوارە ئاڵۆزەکان پرۆسە بکات. لەگەڵ ئەوەشدا ئەمە سیناریۆیەکی سەختترە، چونکە ئاڵوگۆڕێک لەنێوان ئەم ئامانجانەدا هەیە، لە کۆتاییدا هەر ئەلگۆریتمێک لەوانەیە بەیەکبەستنی پرۆسەکردنی جیاواز بەپێی ئامانجەکەی هەڵبژێرێت، کە بۆ کێشەی پێشەوەی (Pareto) دەمانبات. بۆ چارەسەرکردنی ئەم کێشەیە پێشنیار بکە بە بەکارهێنانی باشکردنی فرە-ئامانج، ئەلگۆریتمی فرە-ئامانجی پەرەسەندو لەسەر بنەمای لابردنی فرە ئامانج (MOEA/D) وئەلگۆریتمی بۆماوەیی پۆلێن نەکراوە (NSGA2)، کە ئەلگۆریتمی باشکردنی فرە-ئامانجن، کە دەتوانن بە ڕەچاوکردنی سێ ئامانج (Load, Priority, Distance) باشترین بەیەك بەست هەڵبژێرن.

لیکۆڵینەوە وتاقیکردنەوەکان بە بەکارهێنانیPython evironment لەگەڵ TensorFlow, Pymoo, PQDM, Pytorch, لە PyChram، کە پایتۆنێکی بەهێزی IDE، بۆ هاوشیوە وڕاهێنانی ستراتیژییەتی خشتەکردنی زیرەک. هەروەها Virtualized data دەکرێت بە بەکارهێنانی MatPlotLib لە Jupyter notebok ئاماژە بەوە بکات، کە ستراتیژیی خشتەکردنی دەستکردی زیرەک دەتوانێت بۆ ئەنجامدانی جۆرەها ئەدای بەکارهێنراو ئەنجامی باشتر بەدەست بهێنێت، کارایی و پێوانەکانی توانای خۆگونجاندن: کاتی تەواوکردنی ئەرک MAKESPAN، دواکەوتنی ڕیزبەند، دواکەوتنی بڵاوبوونەوە، دواکەوتنی گواستنەوە، دواکەوتنی کرداری پرۆسیسکردن، دواکەوتنی کۆمپیوتینگ، دواکەوتن، بارکردنی سی پی یو (CPU Load)، بەکارهێنانی هەڵگرتن (Storage Capacity)، تێکڕای بەهای 2.02ms, 10ms, 2ms, 9.9ms, 25ms, 1.0ms, 3.5ms, %10, 99 % بەراورد بە توێژینەوە پەیوەندیدارەکانی ئێستا.